Translating **AWN networks** to the **mCRL2 model-checker**

En route to formal routing protocol development

Djurre van der Wal

UNIVERSITY OF TWENTE.

DATA 61 CSIRO

UNIVERSITY
OF TWENTE.

MASTER THESIS, UNIVERSITY OF TWENTE, NETHERLANDS

HTTP://HOEFNER-ONLINE.DE/IFM18/

This research was done under the supervision of Dr. Peter Höfner and Dr. Rob van Glabbeek with the financial support of Data61 (CSIRO) in Australia and the Twente Mobility Fund within a total of 7 months (excluding 2 months for preliminary research).

First release, June 2018

Word of thanks

As may be inferred from the photographs used in this report, I did the bulk of the work for my master thesis in Sydney, Australia. The amazing offer to go to the other side of the world for an extended period of time was made by Peter when he gave a course on verifying network protocols at the University of Twente, and I am very grateful to him for the opportunity that this gave me.

In Australia, working on my project with Peter and Rob felt more like cooperation than supervision, which was a wonderful experience. I loved our long discussions, including the times when they accelerated with enthusiasm beyond my comprehension – this was the sort of moment in which I could see a glimpse of 'their world' towards which my work has only been a modest contribution.

I made several friends in Australia; mostly among colleagues, but there were also those I met only briefly during my many small trips into Sydney. Everyone was very hospitable, even when political differences resulted in long, passionate conversations, and I wish them all the best.

Finally, a word of thanks to university staff, family, long-time friends, and house mates back home. All have been extremely accommodating, which made the entire enterprise as smooth as I could have hoped for.

Thanks to you all.

Djurre

Contents

Introduction

Nowadays, there is little overlap between the formal analysis of systems and the development of routing protocols. Typically, it is only a long development time and a large number of contributors that give us confidence in a particular routing protocol [1]. If issues with a protocol are eventually discovered when the protocol has already become a standard and has been implemented on a large scale, the consequences are considerable.

The Border Gateway Protocol (BGP), for example, is one of the protocols used to establish routes between internet service providers. It was discovered that routers in this protocol could cycle rapidly through a list of possible routes for a particular destination, meaning that the network does not converge *and* becomes less efficient. The protocol was therefore extended in 1998 with a mechanism known as *route flap damping* [2]. There are indications, however, that this mechanism has negative side-effects of its own [3], and that the increased performance of modern routers makes the originally undesirable behavior of BGP preferable.

Another example is the Ad hoc On-Demand Distance Vector (AODV) protocol, a popular protocol designed for WMNs (Wireless Mesh Networks) and MANETs (Mobile Ad-hoc Networks) [4]. This protocol does not have certain properties that are often taken for granted, such as loop freedom [5] or packet delivery [6].

Formal analysis of routing protocols increases the likelihood that problems such as the ones mentioned above to be discovered during development, which would enable proper structural solutions. Formal analysis of routing protocols would also require those protocols to be formally *specified* – this itself would be an improvement to protocol development, since ambiguous or incomplete segments are far from uncommon in informal protocol specifications. The specification of the AODV protocol, for example, was found to be open to 5184 different interpretations [7]!

However, routing protocol developers often prioritize code writing over theoretical work, and they also may not have the knowledge required to use model-checking tools to analyze their product automatically. It would therefore be useful to develop tools that make the formal specification and analysis of routing protocols more accessible from their point of view.

This report describes a 7 month project with this goal. One of the products of this project is a development environment in Eclipse[1] for a formal language for wireless network protocols, AWN. AWN (Algebra for Wireless Network protocols) is a process algebra that has been developed as a contribution to the formalization of wireless network protocols. It was proposed in 2012 [6] for the purpose of specifying WMN and MANET protocols unambiguously and evaluating and validating them exhaustively. The ultimate goal of AWN is to reduce development time of (modifications of) WMN and MANET protocols and to increase their reliability and performance.

AWN's first use was demonstrated by modeling several interpretations of the AODV protocol [8], revealing, for instance, that not all of these interpretations guarantee loop freedom [5] or packet delivery [6]. One of the ambitions for AWN is that it will be possible to automatically model and verify AWN specifications with existing model checking toolsets via translations of AWN to input that these toolsets accept.

AWN distinguishes itself from existing formalisms by disregarding packet loss as a possibility – that is, a message transmitted in AWN is received by all intended recipients within range. This abstraction allows the verification of the property of a network that a packet inserted by a client into the network is eventually delivered to the destination. AWN also features a conditional unicast operator and imperative 'mid-process' variable assignments.

In 2016, T-AWN was proposed [9], a *timed* process algebra reusing the existing AWN syntax. It was used to continue the investigation of the AODV protocol, and it was shown that *without* time abstraction the unambiguous interpretations of the AODV protocol always fail the loop freedom property.

For this project, a plain-text input language for AWN was developed, including data expressions. This made it possible to implement a development environment for AWN in Eclipse, providing syntax highlighting, code completion, refactoring facilities, and more. The environment also serves as a front end for a translation to mCRL2, a generic process algebra with an accompanying modeling and verification toolset [10]. This translation is implemented as a module in a translation *framework* – the idea is that more translations can easily be added in the future.

Another prospect for the environment is to use it for *code generation*; this way, routing protocol developers could reap double benefits from using the software, and be more inclined to divert attention to formal analysis. Furthermore, properties of protocols cannot yet be specified in the Eclipse environment, and the analysis of properties must therefore still be done by hand. The implementation can also only be used to analyze given topologies. Working on these ideas and shortcomings are only some of the future work that remains.

The project was also used as the foundation of a paper published at the International Conference of integrated Formal Methods [11]. It can be found online[2].

[1] `https://www.eclipse.org/`

[2] `http://hoefner-online.de/ifm18/`

1. Motivation

The early stage of the project focused on two important questions:

- Which model-checker should be used to analyze AWN specifications?
- Which approach should be taken for developing the accompanying software?

This chapter gives the arguments for the decisions that were made, namely using mCRL2 as the model-checker and applying model-driven engineering techniques to the software design.

1 Why mCRL2?

In the first stage of the research mCRL2 was chosen as the target model checking toolset for the translation from AWN. This was done by choosing several criteria that the target toolset should satisfy. These criteria were used to make a selection of model checking toolsets found in internet databases [12] [13] [14]. In general, this selection process has suffered from a time constraint that was self-imposed to prevent the comparison of existing toolsets from being taken further than what is beneficial to this graduation project. Searching for unambiguous information about a certain model checking toolset was therefore stopped after approximately 1.5 hours. As a consequence, information might have been missed, and a toolset might have been dismissed unjustly.

Furthermore, there are multiple model checking toolsets that are based on their own particular algebra or logic. A comparison of these toolsets would require much more in-depth study of and experience with these toolsets, which would consume valuable project time. It is therefore possible that relevant information was misunderstood or incorrectly considered to be ambiguous.

Finally, it should be noted that the databases where model checking toolsets were found cannot be expected to contain *all* model checking toolsets, and it is an assumption made by the author that the toolsets that were found are at least representative of the state-of-the-art functionality.

Toolset	Failed criteria
ECW	4
IVy	6
Mobility Workbench	4
PEP	3, 6
Spot 2.0	2
TAPAs	4
TLA+	6
UCLID 3	4
UPPAAL	(An AWN-to-UPPAAL translation is already in development.)
ZING	2

Table 1.1: Toolsets that were rejected and why.

The chosen criteria for the model checking toolset to be targeted by the translation from AWN are these:

1. A toolset should have clear support for generic model checking, and not be specifically aimed at, for instance, code verification or probabilistic model checking.
2. A toolset should have accessible, complete, and up-to-date formal semantics. This is required to prove the validity of the translation later in the research.
3. A toolset should have sufficient online documentation, including examples, tutorials, and possibly active user forums. This makes the toolset easier to learn, which facilitates more advanced use of that toolset as a back end for AWN.
4. A toolset must have been updated since 2012 (in the previous 5 years). This criteria aims to exclude tools that are no longer being developed and improved without having to rely on that information being publicly available (which it may not be), and to exclude tools with very slow release cycles.
5. A toolset should have a text-based, sufficiently expressive modeling language, so that the result of a translation is easily readable by users and manual adjustments can be made if desired. This implies the presence of a framework for modeling concurrent systems, for example, but also more basic features, such as data types, functions, arithmetical operators, and so on.
6. A toolset should have a sufficiently expressive property language, so that a maximum number of different types of properties of a protocol can be analyzed. Toolsets with more expressive property languages are preferred over those with less expressive ones.

Criteria 1 has been used as an initial filter to reduce the number of model checking toolsets. Table 1.1 lists the toolsets that failed one or more of the other criteria. From the toolsets that remained, listed in Table 1.2, mCRL2 was chosen because of prior experience with mCRL2 of the author as well as the similarity between AWN and mCRL2 (they are both process algebra languages). Additionally, there exists a LTSmin [15] back-end for mCRL2, so translating to mCRL2 yields two supporting tools ‘for the price of one’.

Toolset	Modelling language(s)	Reference
ARC	AltaRica	[16]
CADP	LOTOS, FSP, LOTOS NT	[17]
FDR	CSPM	[18]
LTSmin	Promela, μCRL, mCRL2	[15]
mCRL2	mCRL2	[19]
nuXmv	SMV	[20]
SPIN	Promela	[21]

Table 1.2: Selected toolsets.

2 Why MDE?

The translation from AWN to mCRL2 has been implemented using *model-driven engineering techniques*. This means that rather than object-focused, the implementation is model-focused, where 'models' are essentially groups of related classes.

Model-driven engineering (MDE) has been chosen because they provide tools to construct new models from existing models via *transformations*. This means that if an AWN specification is stored in an input model and an mCRL2 specification can be stored in an mCRL2 model, a transformation can be created that goes from the former to the latter – the transformation being the implementation of the AWN-to-mCRL2 translation, of course. In short, the basic MDE architecture precisely matches the structure needed by a translation framework for AWN.

Models conform to the structure defined by *metamodels*, which conform to a *metametamodel*. Frameworks can be built around such a metametamodel so that software can be generated that accepts models conforming to a customized metamodel as input. Transformation languages such as ATL and QVTO are such frameworks. Xtext is a framework that can be used to generate the core of an Eclipse plug-in, which means that it is easy to create an editor with the features expected by average users.

Another advantage of MDE is that it automatically sets a high standard for the degree at which the different aspects of an implementation are separated, similar to *aspect-oriented techniques* [22]. Other approaches in software engineering are often more prone to situations in which multiple concerns are addressed in one place, reducing code maintainability.

The disadvantage of the MDE approach is that it requires developers to know 1 general-purpose language (Java or another language supported by MDE tools) and at least 4 DSLs (there exist different DSLs for the same purpose in MDE) instead of only 1 language:

- One for specifying the AWN metamodel, and potentially the mCRL2 metamodel;
- One for specifying the grammar of the AWN input language;
- One for specifying the transformation from AWN to mCRL2;
- One for specifying how an mCRL2 model is converted to text.

2. Formal translation

1 AWN semantics

This first section will give an overview of the semantics of AWN. The corresponding inference rules can be found in Tables 2.1 to 2.5. The labels that identify the inference rules show a number in between brackets which refers to the table in [6] from which they were copied. For a full description of the semantics of AWN, consult that document.

The semantics of AWN are divided into four 'levels':

- The *sequential process level*. The semantics of this level describe how the decision flow of a single process of a protocol is specified, including guards, variable assignments, and local broadcasts.
- The *parallel process level* combines multiple sequential processes into a single parallel process so that they can run on the same node. Combined sequential processes can only communicate in one direction.
- The *node level* gives parallel processes their address and the set of addresses of nodes that are within their transmission range. It also adds network behavior such as connecting and disconnecting to parallel processes.
- The *network level* determines which behavior is ultimately allowed to occur: a node that tries to transmit a message, for example, will only succeed if a recipient actively cooperates.

1.1 Sequential process level

The decision flow of a protocol is determined by a composition of sequential processes. Sequential processes have a signature $X(\mathtt{var}_1, \ldots, \mathtt{var}_n)$ consisting of a name X and a number of parameters $\mathtt{var}_i$, and their behavior is specified via a sequential process expression SP with the following

grammar:

$$\begin{aligned}\text{SP} ::= \ & [\phi]\ \text{SP} \quad | \quad [\![\texttt{var} := exp]\!]\ \text{SP} \\ | \ & \textbf{broadcast}(msg).\text{SP} \quad | \quad \textbf{groupcast}(dests, msg).\text{SP} \\ | \ & \textbf{unicast}(dest, msg).\text{SP} \blacktriangleright \text{SP} \quad | \quad \textbf{send}(msg).\text{SP} \\ | \ & \textbf{deliver}(data).\text{SP} \quad | \quad \textbf{receive}(\texttt{msg}).\text{SP} \\ | \ & \text{X}(exp_1, \ldots, exp_n)\end{aligned}$$

Clearly, sequential processes have many possibilities to determine the behavior of a protocol: using guards $[\phi]$ SP, they can elect to perform certain actions only under particular circumstances; variable assignment $[\![\texttt{var} := exp]\!]$ SP allows the contents of the internal data structure of the sequential process to be changed; they can perform a **broadcast**, **groupcast**, **unicast**, **send**, **deliver**, or **receive** action, which allow processes to exchange messages in specific ways; they can let another sequential process $\text{X}(exp_1, \ldots, exp_n)$ determine their subsequent behavior; and they can choose non-deterministically between multiple of these possibilities via the $+$ operator.

The different types of message exchanges have distinct purposes: **broadcast** is used to send a message to *all* nodes in the network that are within range; **groupcast** does the same as long as nodes are in a specified subset; **unicast** allows a message to be sent to a specified node, continuing with its first branch p if that node is within range or continuing with a $\neg$**unicast** action and its second branch q if the transmission failed; and **send** passes a message to another sequential process running on the same node (see Section 1.2).

The **receive** action performs the complementary action: it intercepts a message (regardless of whether it originated from a **broadcast**, **groupcast**, **unicast**, or **send** action) and stores it in a specified variable. Messages (or rather, the relevant data that they contain) can exit the network via the **deliver** action.

Inference rules BROADCAST (T1) to GUARD (T1) in Table 2.1 give the semantics of sequential process expressions ξ, p in AWN. In these expressions, ξ is a variable-to-value mapping and p the current sequential process expression. Note that inference rules that contain expressions of the form $\xi(exp)$ are only defined if exp is bound by ξ!

Interpretation of guards

Guards in AWN use the syntax $[\phi]\ p$. The rule in the original semantics of AWN that allows guards to be constructed makes use of the notation $\xi \xrightarrow{\phi} \zeta$ [6]. Informally, this syntax is defined to mean that the variable-to-value mapping ζ extends variable-to-value mapping ξ with new mappings for variables unmapped in ξ in such a way that ϕ under ζ evaluates to *true*. This allows AWN to set variables in guards using a type of *pattern matching*.

For example, let `ip` and `data` be unmapped when the following expression is executed:

$$\textbf{receive}(\texttt{msg}).[\texttt{msg} = \texttt{Message}(\texttt{ip}, \texttt{data})]\ p$$

The execution will reach p if and only if a message is received that conforms to the structure $\texttt{Message}(ip, data)$, after which p is executed with $\xi(\texttt{ip}) := ip$ and $\xi(\texttt{data}) := data$.

$$\frac{}{\xi, \textbf{broadcast}(ms).\mathrm{p} \xrightarrow{\textbf{broadcast}(\xi(ms))} \xi, \mathrm{p}} \text{ BROADCAST (T1)}$$

$$\frac{}{\xi, \textbf{groupcast}(dests, ms).\mathrm{p} \xrightarrow{\textbf{groupcast}(\xi(dests),\xi(ms))} \xi, \mathrm{p}} \text{ GROUPCAST (T1)}$$

$$\frac{}{\xi, \textbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q} \xrightarrow{\textbf{unicast}(\xi(dest),\xi(ms))} \xi, \mathrm{p}} \text{ UNICAST (T1-1)}$$

$$\frac{}{\xi, \textbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q} \xrightarrow{\neg\textbf{unicast}(\xi(dest),\xi(ms))} \xi, \mathrm{q}} \text{ UNICAST (T1-2)}$$

$$\frac{}{\xi, \textbf{send}(ms).\mathrm{p} \xrightarrow{\textbf{send}(\xi(ms))} \xi, \mathrm{p}} \text{ SEND (T1)}$$

$$\frac{}{\xi, \textbf{deliver}(data).\mathrm{p} \xrightarrow{\textbf{deliver}(\xi(data))} \xi, \mathrm{p}} \text{ DELIVER (T1)}$$

$$\forall m \in \mathrm{MSG} \; \frac{}{\xi, \textbf{receive}(\mathtt{msg}).\mathrm{p} \xrightarrow{\textbf{receive}(m)} \xi[\mathtt{msg} := m], \mathrm{p}} \text{ RECEIVE (T1)}$$

$$\frac{}{\xi, [\![\mathtt{var} := exp]\!]\mathrm{p} \xrightarrow{\tau} \xi[\mathtt{var} := \xi(exp)], \mathrm{p}} \text{ ASSIGNMENT (T1)}$$

$$\forall a \in \mathrm{Act} \; \frac{\emptyset[\mathtt{var}_i := \xi(exp_i)]_{i=1}^{n}, \mathrm{p} \xrightarrow{a} \zeta, \mathrm{p}' \qquad \mathrm{X}(\mathtt{var}_1, \cdots, \mathtt{var}_n) \stackrel{\text{def}}{=} \mathrm{p}}{\xi, \mathrm{X}(exp_1, \cdots, exp_n) \xrightarrow{a} \zeta, \mathrm{p}'} \text{ RECURSION (T1)}$$

$$\forall a \in \mathrm{Act} \; \frac{\xi, \mathrm{p} \xrightarrow{a} \zeta, \mathrm{p}'}{\xi, \mathrm{p} + \mathrm{q} \xrightarrow{a} \zeta, \mathrm{p}'} \text{ CHOICE (T1-1)}$$

$$\forall a \in \mathrm{Act} \; \frac{\xi, \mathrm{q} \xrightarrow{a} \zeta, \mathrm{q}'}{\xi, \mathrm{p} + \mathrm{q} \xrightarrow{a} \zeta, \mathrm{q}'} \text{ CHOICE (T1-2)}$$

$$\zeta(\phi) = \mathit{true} \wedge \{\mathtt{q}_1, \cdots, \mathtt{q}_n\} = \mathrm{FV}(\phi) \setminus \mathrm{DOM}(\xi) \; \frac{\zeta = \xi[\mathtt{q}_1 := e_1, \cdots, \mathtt{q}_n := e_n]}{\xi, [\phi]\mathrm{p} \xrightarrow{\tau} \zeta, \mathrm{p}} \text{ GUARD (T1)}$$

Table 2.1: AWN inference rules (1/5)

For the correctness proof of the translation, a formal interpretation of $\xi \xrightarrow{\phi} \zeta$ is needed. To this end, GUARD (T1) uses the following side condition:

$$\zeta(\phi) = \mathit{true} \wedge \{\mathtt{q}_1, \cdots, \mathtt{q}_n\} = \text{FV}(\phi) \setminus \text{DOM}(\xi)$$

where $\text{FV}(\phi)$ are the free variables in ϕ and $\text{DOM}(\xi) = \{\ \mathtt{x} \mid \mathtt{x} := \mathtt{y} \in \xi\ \}$.

$$\forall a \neq \mathbf{receive}(m)\ \frac{\mathrm{P} \xrightarrow{a} \mathrm{P}'}{\mathrm{P} \langle\langle\ \mathrm{Q} \xrightarrow{a} \mathrm{P}' \langle\langle\ \mathrm{Q}}\ \text{PARALLEL (T2-1)}$$

$$\forall a \neq \mathbf{send}(m)\ \frac{\mathrm{Q} \xrightarrow{a} \mathrm{Q}'}{\mathrm{P} \langle\langle\ \mathrm{Q} \xrightarrow{a} \mathrm{P} \langle\langle\ \mathrm{Q}'}\ \text{PARALLEL (T2-2)}$$

$$\forall m \in \text{MSG}\ \frac{\mathrm{P} \xrightarrow{\mathbf{receive}(m)} \mathrm{P}' \qquad \mathrm{Q} \xrightarrow{\mathbf{send}(m)} \mathrm{Q}'}{\mathrm{P} \langle\langle\ \mathrm{Q} \xrightarrow{\tau} \mathrm{P}' \langle\langle\ \mathrm{Q}'}\ \text{PARALLEL (T2-3)}$$

Table 2.2: AWN inference rules (2/5)

1.2 Parallel process level

Sequential processes are combined into a single parallel process according to the following grammar:

$$\text{PP} ::= \xi, \text{SP} \quad | \quad \text{PP} \langle\langle\ \text{PP}$$

The sequential processes form a pipeline for messages: processes can use the **send** action to send a message to the process to their left, where the **receive** action can be used to receive it. Only the rightmost sequential process can use **receive** to listen for messages from other nodes in the network. See inference rules PARALLEL (T2-1) to PARALLEL (T2-3) in Table 2.2 for the semantics.

1.3 Node level

The rules for the node level, BROADCAST (T3) to DISCONNECT (T3-3), can be found in Table 2.3. These rules add network behavior to parallel processes: they allow them to connect, to disconnect, to deliver data objects, and they rewrite actions such as $\mathbf{broadcast}(m)$ and $\mathbf{unicast}(d,m)$ in terms of the current state of the network: $\mathbf{broadcast}(m)$ will be converted to $R : \mathbf{starcast}(m)$, where R is the set of addresses of all nodes within range, whereas $\mathbf{unicast}(d,m)$ is converted to the action $\{d\} : \mathbf{starcast}(m)$ because only the node with address d should be a recipient.

The inference rules for the node level make use of the $ip : \text{PP} : R$ notation to consistently access the part of the network state relevant to a node. In this notation PP is a syntactic parallel process expression, ip is the address of the node that runs PP (as a semantic value), and R is the set of addresses of nodes that are within range of that node (also as a semantic value).

$$\frac{\mathrm{P} \xrightarrow{\textbf{broadcast}(m)} \mathrm{P}'}{ip : \mathrm{P} : R \xrightarrow{R:\textbf{*cast}(m)} ip : \mathrm{P}' : R}\ \textsc{Broadcast (T3)} \qquad \frac{\mathrm{P} \xrightarrow{\textbf{groupcast}(D,m)} \mathrm{P}'}{ip : \mathrm{P} : R \xrightarrow{R\cap D:\textbf{*cast}(m)} ip : \mathrm{P}' : R}\ \textsc{Groupcast (T3)}$$

$$\frac{\mathrm{P} \xrightarrow{\textbf{unicast}(dip,m)} \mathrm{P}' \qquad dip \in R}{ip : \mathrm{P} : R \xrightarrow{\{dip\}:\textbf{*cast}(m)} ip : \mathrm{P}' : R}\ \textsc{Unicast (T3-1)} \qquad \frac{\mathrm{P} \xrightarrow{\neg\textbf{unicast}(dip,m)} \mathrm{P}' \qquad dip \notin R}{ip : \mathrm{P} : R \xrightarrow{\tau} ip : \mathrm{P}' : R}\ \textsc{Unicast (T3-2)}$$

$$\frac{\mathrm{P} \xrightarrow{\textbf{deliver}(d)} \mathrm{P}'}{ip : \mathrm{P} : R \xrightarrow{ip:\textbf{deliver}(d)} ip : \mathrm{P}' : R}\ \textsc{Deliver (T3)}$$

$$\frac{\mathrm{P} \xrightarrow{\tau} \mathrm{P}'}{ip : \mathrm{P} : R \xrightarrow{\tau} ip : \mathrm{P}' : R}\ \textsc{Internal (T3)}$$

$$\frac{\mathrm{P} \xrightarrow{\textbf{receive}(m)} \mathrm{P}'}{ip : \mathrm{P} : R \xrightarrow{\{ip\}\neg\emptyset:\textbf{arrive}(m)} ip : \mathrm{P}' : R}\ \textsc{Arrive (T3-1)} \qquad \frac{}{ip : \mathrm{P} : R \xrightarrow{\emptyset\neg\{ip\}:\textbf{arrive}(m)} ip : \mathrm{P} : R}\ \textsc{Arrive (T3-2)}$$

$$\frac{}{ip : \mathrm{P} : R \xrightarrow{\textbf{connect}(ip,ip')} ip : \mathrm{P} : R \cup \{ip'\}}\ \textsc{Connect (T3-1)}$$

$$\frac{}{ip : \mathrm{P} : R \xrightarrow{\textbf{connect}(ip',ip)} ip : \mathrm{P} : R \cup \{ip'\}}\ \textsc{Connect (T3-2)}$$

$$\frac{ip \notin \{ip', ip''\}}{ip : \mathrm{P} : R \xrightarrow{\textbf{connect}(ip',ip'')} ip : \mathrm{P} : R}\ \textsc{Connect (T3-3)}$$

$$\frac{}{ip : \mathrm{P} : R \xrightarrow{\textbf{disconnect}(ip,ip')} ip : \mathrm{P} : R \setminus \{ip'\}}\ \textsc{Disconnect (T3-1)}$$

$$\frac{}{ip : \mathrm{P} : R \xrightarrow{\textbf{disconnect}(ip',ip)} ip : \mathrm{P} : R \setminus \{ip'\}}\ \textsc{Disconnect (T3-2)}$$

$$\frac{ip \notin \{ip', ip''\}}{ip : \mathrm{P} : R \xrightarrow{\textbf{disconnect}(ip',ip'')} ip : \mathrm{P} : R}\ \textsc{Disconnect (T3-3)}$$

Table 2.3: AWN inference rules (3/5)

1.4 Network level

A partial network is constructed through parallel composition of nodes (via the $||$ operator):

$$\mathrm{M} ::= \mathit{ip} : \mathrm{PP} : R \quad | \quad \mathrm{M} \,||\, \mathrm{M}$$

Parallel nodes can *exchange messages* according to rules CAST (T4-1) to CAST (T4-4) in Table 2.4. These exchanges occur through synchronization of $R : \mathbf{starcast}(m)$ and $H\neg K : \mathbf{arrive}(m)$ actions (where $\neg$ is simply a syntactic separator of the sets H and K). Of these two actions, $H\neg K : \mathbf{arrive}(m)$ is where a message m arrives at the nodes in the set H and where m is ignored by the nodes in the set K (because they are out of range). It is necessary for synchronization that $H \subseteq R$ and that $K \cap R = \emptyset$ because R in $R : \mathbf{starcast}(m)$ is the set of addresses of the intended recipients within range.

Encapsulation ($[\_]$) allows a partial network M to be converted into a complete network $[M]$. Encapsulation also restricts the set of actions exposed by a network to τ, $\mathbf{connect}(\mathit{ip}, \mathit{ip'})$ and $\mathbf{disconnect}(\mathit{ip}, \mathit{ip'})$, $\mathit{ip} : \mathbf{newpkt}(\mathit{data}, d)$, and $\mathit{ip} : \mathbf{deliver}(\mathit{data})$.

These actions have the following meanings. As usual in process algebra, τ is an action that is hidden from the environment of the network. Actions $\mathbf{connect}(\mathit{ip}, \mathit{ip'})$ and $\mathbf{disconnect}(\mathit{ip}, \mathit{ip'})$ represent events within the network where a node with address $\mathit{ip'}$ respectively enters or leaves the range of a node with address ip (note that all nodes must agree on this topology change). Action $\mathit{ip} : \mathbf{newpkt}(\mathit{data}, d)$ is the event where a data object d to be delivered at the node with address d is injected into the network (wrapped into a message $\texttt{newpkt}(\mathit{data}, d)$) at the node with address ip, whereas action $\mathit{ip} : \mathbf{deliver}(\mathit{data})$ is the actual delivery of a data object at the node with address ip. The accompanying inference rules are DELIVER (T4-1) to NEWPKT (T4) in Table 2.5.

$$H \subseteq R \wedge K \cap R = \emptyset \;\; \frac{\mathrm{M} \xrightarrow{R:\mathbf{*cast}(m)} \mathrm{M}' \qquad \mathrm{N} \xrightarrow{H\neg K:\mathbf{arrive}(m)} \mathrm{N}'}{\mathrm{M} \,||\, \mathrm{N} \xrightarrow{R:\mathbf{*cast}(m)} \mathrm{M}' \,||\, \mathrm{N}'} \;\; \text{CAST (T4-1)}$$

$$H \subseteq R \wedge K \cap R = \emptyset \;\; \frac{\mathrm{M} \xrightarrow{H\neg K:\mathbf{arrive}(m)} \mathrm{M}' \qquad \mathrm{N} \xrightarrow{R:\mathbf{*cast}(m)} \mathrm{N}'}{\mathrm{M} \,||\, \mathrm{N} \xrightarrow{R:\mathbf{*cast}(m)} \mathrm{M}' \,||\, \mathrm{N}'} \;\; \text{CAST (T4-2)}$$

$$\frac{\mathrm{M} \xrightarrow{H\neg K:\mathbf{arrive}(m)} \mathrm{M}' \qquad \mathrm{N} \xrightarrow{H'\neg K':\mathbf{arrive}(m)} \mathrm{N}'}{\mathrm{M} \,||\, \mathrm{N} \xrightarrow{(H\cup H')\neg(K\cup K'):\mathbf{arrive}(m)} \mathrm{M}' \,||\, \mathrm{N}'} \;\; \text{CAST (T4-3)}$$

$$\frac{\mathrm{M} \xrightarrow{R:\mathbf{*cast}(m)} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{\tau} [\,\mathrm{M}'\,]} \;\; \text{CAST (T4-4)}$$

Table 2.4: AWN inference rules (4/5)

$$\frac{\mathrm{M} \xrightarrow{ip:\mathbf{deliver}(d)} \mathrm{M}'}{\mathrm{M} \,||\, \mathrm{N} \xrightarrow{ip:\mathbf{deliver}(d)} \mathrm{M}' \,||\, \mathrm{N}} \text{ DELIVER (T4-1)}$$

$$\frac{\mathrm{N} \xrightarrow{ip:\mathbf{deliver}(d)} \mathrm{N}'}{\mathrm{M} \,||\, \mathrm{N} \xrightarrow{ip:\mathbf{deliver}(d)} \mathrm{M} \,||\, \mathrm{N}'} \text{ DELIVER (T4-2)}$$

$$\frac{\mathrm{M} \xrightarrow{ip:\mathbf{deliver}(d)} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{ip:\mathbf{deliver}(d)} [\,\mathrm{M}'\,]} \text{ DELIVER (T4-3)}$$

$$\frac{\mathrm{M} \xrightarrow{\tau} \mathrm{M}'}{\mathrm{M} \,||\, \mathrm{N} \xrightarrow{\tau} \mathrm{M}' \,||\, \mathrm{N}} \text{ INTERNAL (T4-1)}$$

$$\frac{\mathrm{N} \xrightarrow{\tau} \mathrm{N}'}{\mathrm{M} \,||\, \mathrm{N} \xrightarrow{\tau} \mathrm{M} \,||\, \mathrm{N}'} \text{ INTERNAL (T4-2)}$$

$$\frac{\mathrm{M} \xrightarrow{\tau} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{\tau} [\,\mathrm{M}'\,]} \text{ INTERNAL (T4-3)}$$

$$\frac{\mathrm{M} \xrightarrow{\mathbf{connect}(ip,ip')} \mathrm{M}' \qquad \mathrm{N} \xrightarrow{\mathbf{connect}(ip,ip')} \mathrm{N}'}{\mathrm{M} \,||\, \mathrm{N} \xrightarrow{\mathbf{connect}(ip,ip')} \mathrm{M}' \,||\, \mathrm{N}'} \text{ CONNECT (T4-1)}$$

$$\frac{\mathrm{M} \xrightarrow{\mathbf{connect}(ip,ip')} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{\mathbf{connect}(ip,ip')} [\,\mathrm{M}'\,]} \text{ CONNECT (T4-2)}$$

$$\frac{\mathrm{M} \xrightarrow{\mathbf{disconnect}(ip,ip')} \mathrm{M}' \qquad \mathrm{N} \xrightarrow{\mathbf{disconnect}(ip,ip')} \mathrm{N}'}{\mathrm{M} \,||\, \mathrm{N} \xrightarrow{\mathbf{disconnect}(ip,ip')} \mathrm{M}' \,||\, \mathrm{N}'} \text{ DISCONNECT (T4-1)}$$

$$\frac{\mathrm{M} \xrightarrow{\mathbf{disconnect}(ip,ip')} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{\mathbf{disconnect}(ip,ip')} [\,\mathrm{M}'\,]} \text{ DISCONNECT (T4-2)}$$

$$\frac{\mathrm{M} \xrightarrow{\{ip\}\neg K:\mathbf{arrive}(\mathtt{newpkt}(d,dip))} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{ip:\mathbf{newpkt}(d,dip)} [\,\mathrm{M}'\,]} \text{ NEWPKT (T4)}$$

Table 2.5: AWN inference rules (5/5)

1.5 AWN examples

Listings 2.1 to 2.4 contain simple examples to illustrate how AWN might be used. The example of Listing 2.1 defines a radio tower (`RadioTower`, line 1) that broadcasts messages to radios (`Radio`, line 4) within range. Each message contains a value 1 higher than the value of the preceding message, and radios store the most recently received message. A specific network related to this example might be defined as in lines 6 through 9: the radio tower has address 1, there are 3 radios with addresses 2 to 4, and only radios with addresses 2 and 4 are within range of the radio tower.

```
1  RadioTower(msgVal: Integer) =
2    broadcast(new Message(msgVal)) . RadioTower(msgVal + 1);
3
4  Radio(lastMsg: Message) = receive(msg) . Radio(msg);
5
6  RadioNetwork = [ 1 : RadioTower(1) : { 2, 4 } ||
7                   2 : Radio(null) : { 1 } ||
8                   3 : Radio(null) : ∅ ||
9                   4 : Radio(null) : { 2 } ];
```

Listing 2.1: AWN radio tower example.

```
1  ChatClient(lastMsg: Message, sentMsg: Boolean) =
2      receive(msg) . ChatClient(msg, sentMsg)
3    + unicast(1, Message("Hello everyone!")) .
4        ChatClient(lastMsg, true) ▶ ChatClient(lastMsg, false);
5
6  ChatServer() = receive(msg) . broadcast(msg) . ChatServer();
```

Listing 2.2: AWN 'chat client' example.

The second example introduces a 'chat client' (`ChatClient`, line 1) that listens for messages similar to the `Radio` process. However, the server (`ChatServer`, line 6) only forwards messages that it receives from clients within range. These messages are non-deterministically sent by clients to a node within range that has address 1, which is assumed to be the chat server. Clients always send the same message.

Listing 2.3 defines a simple flooding protocol. Each node of a network runs the same process (`Flood`), line 1). This process injects its message into the network at a non-deterministic time (similar to `ChatClient` from the previous example, but now nodes send their message a maximum of one time). The process also listens for incoming messages: for each node address, it stores the contents of the first message received from that node.

When analyzed, the example of Listing 2.3 reveals a problem with the protocol: the network can deadlock when some nodes A and B have just received a message (at the end of line 2) that is not in their `store` (line 4) and when either A or B needs to forward its message to node B in order for it to complete its **broadcast** action. The other node is essentially 'not ready to receive', also described as not being *input-enabled*.

```
1  Flood(msgSent: Boolean, store: IP ↦ Text) =
2      receive(msg) . [msg = Message(ip', text)] (
3        [ip' ↦ text ∈ store] . Flood(msgSent, store)
4      + [ip' ↦ text ∉ store] .
5          ⟦ store := store[ip' ↦ text] ⟧
6            broadcast(msg) . Flood(msgSent, store))
7    + [¬ msgSent] broadcast(new Message("Hello everyone!")) .
8        Flood(true, store);
```

Listing 2.3: AWN flooding example.

```
1  Queue(queue: Sequence(Message)) =
2      receive(msg) . Queue(queue→append(msg))
3    + [queue→size() > 0] (
4        receive(msg) . Queue(queue→append(msg))
5      + send(queue→at(0)) . Queue(queue→remove(0))
6    );
7
8  FloodWithQueue = Flood(false, ∅) << Queue(∅);
9
10 FloodNetwork = [ 1 : FloodWithQueue() : { 2, 3, 4 } ||
11                  2 : FloodWithQueue() : { 1, 3 } ||
12                  3 : FloodWithQueue() : { 1, 2 } ||
13                  4 : FloodWithQueue() : { 1 } ];
```

Listing 2.4: AWN queued flooding example.

The example of Listing 2.4 uses the typical approach to make a protocol specification input-enabled: in addition to the original `Flood` process, each node is also running the `Queue` process (line 8). Because the `Queue` process is input-enabled (note the seemingly superfluous **receive** on line 4, which makes it possible to receive a message in the process state just after the guard of line 3 has been evaluated), the `Flood` process can take more time to handle incoming messages without causing deadlocks.

2 mCRL2 semantics

For the translation from AWN to mCRL2, the features of mCRL2 related to time will not be used. This section will therefore give the *untimed* fragment of the semantics of mCRL2 (a similar approach as the one used in [23]). Note that only an overview is provided here; refer to the book in which mCRL2 is documented [24] for the full semantics.

2.1 Grammar

Similar to sequential processes in AWN, mCRL2 processes have a signature $X(\mathtt{var}_1 : D_1, \cdots, \mathtt{var}_n : D_n)$ where X is a name and $\mathtt{var}_i$ are process parameters of data type D_i. All mCRL2 processes are contained within the set PD. The behavior of mCRL2 processes is specified by expressions conforming to the following grammar:

$$\begin{array}{rcl} \mathrm{p} ::= \delta & | & \omega \mid \mathrm{p}.\mathrm{p} \mid \mathrm{p}+\mathrm{p} \mid c \to \mathrm{p} \mid c \to \mathrm{p} \diamond \mathrm{p} \mid \sum_{\mathtt{x}:\mathrm{D}} \mathrm{p} \\ & | & \mathrm{p} \,||\, \mathrm{p} \mid \Gamma_C(\mathrm{p}) \mid \nabla_V(\mathrm{p}) \mid \partial_B(\mathrm{p}) \mid \rho_R(\mathrm{p}) \mid \tau_I(\mathrm{p}) \\ & | & \mathrm{X}(\mathtt{d}_1, \cdots, \mathtt{d}_n) \\ \omega ::= \tau & | & \mathbf{a}(\mathtt{d}_1, \cdots, \mathtt{d}_n) \mid \omega|\omega \end{array}$$

In this grammar, δ represents a deadlocked process, meaning that it cannot make any transitions. To do an action ω, one only has to write it; however, mCRL2 actually uses *multi-actions*, which are actions that potentially consist of multiple action labels $\mathbf{a}$ that can each carry their own data. The action τ is the empty multi-action with the property that $\omega|\tau = \tau|\omega = \omega$.

Processes can be chained, or choices can be made between them, either by checking a condition c or non-deterministically – the expression $\sum_{\mathtt{x}:\mathrm{D}} \mathrm{p}$, in particular, is used to express a non-deterministic choice between summands for every possible value of $\mathtt{x}$, a variable of data type D. Expressions p and q can be put in parallel by writing p || q. Finally Γ_C, ∇_V, ∂_B, ρ_R, and τ_I are operators that can influence the behavior of a process in various ways – their descriptions can be found in 2.3.

2.2 Inference rules

Tables 2.6 and 2.7 list rules in Plotkin style [25] for the structural operational semantics of mCRL2 process expressions. In these rules, several notations are used that require short explanations:

- The ✓-predicate is used to indicate the mCRL2 termination state – a process 'goes here' when it has no more actions to do;
- The syntax $\mathtt{d} : \mathrm{D}$ means that variable $\mathtt{d}$ is of type D;
- M_D refers to the set of all possible semantic values of type D. Furthermore, if $e \in M_\mathrm{D}$ then there exists t_e, a syntactic symbol for e such that $[\![t_e]\!] = e$ (see Definition 15.2.17 in [24]);
- $\underline{\omega}$ denotes a multi-action from which all data has been removed;
- $\omega_{\{\}}$ is the set of all single actions that multi-action ω contains.

$$\frac{}{\alpha \xrightarrow{[\![\alpha]\!]} \checkmark}\ \textsc{Axiom}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \checkmark}{\mathrm{p}+\mathrm{q} \xrightarrow{\omega} \checkmark}\ \textsc{Choice 1} \qquad [\![c]\!] = \mathit{true}\ \frac{\mathrm{p} \xrightarrow{\omega} \checkmark}{c \to \mathrm{p} \xrightarrow{\omega} \checkmark}\ \textsc{Guard 1}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \mathrm{p}'}{\mathrm{p}+\mathrm{q} \xrightarrow{\omega} \mathrm{p}'}\ \textsc{Choice 2} \qquad [\![c]\!] = \mathit{true}\ \frac{\mathrm{p} \xrightarrow{\omega} \mathrm{p}'}{c \to \mathrm{p} \xrightarrow{\omega} \mathrm{p}'}\ \textsc{Guard 2}$$

$$\frac{\mathrm{q} \xrightarrow{\omega} \checkmark}{\mathrm{p}+\mathrm{q} \xrightarrow{\omega} \checkmark}\ \textsc{Choice 3} \qquad [\![c]\!] = \mathit{true}\ \frac{\mathrm{p} \xrightarrow{\omega} \checkmark}{c \to \mathrm{p} \diamond \mathrm{q} \xrightarrow{\omega} \checkmark}\ \textsc{Guard 3}$$

$$\frac{\mathrm{q} \xrightarrow{\omega} \mathrm{q}'}{\mathrm{p}+\mathrm{q} \xrightarrow{\omega} \mathrm{q}'}\ \textsc{Choice 4} \qquad [\![c]\!] = \mathit{true}\ \frac{\mathrm{p} \xrightarrow{\omega} \mathrm{p}'}{c \to \mathrm{p} \diamond \mathrm{q} \xrightarrow{\omega} \mathrm{p}'}\ \textsc{Guard 4}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \checkmark}{\mathrm{p}.\mathrm{q} \xrightarrow{\omega} \mathrm{q}}\ \textsc{Seq 1} \qquad [\![c]\!] = \mathit{false}\ \frac{\mathrm{q} \xrightarrow{\omega} \checkmark}{c \to \mathrm{p} \diamond \mathrm{q} \xrightarrow{\omega} \checkmark}\ \textsc{Guard 5}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \mathrm{p}'}{\mathrm{p}.\mathrm{q} \xrightarrow{\omega} \mathrm{p}'.\mathrm{q}}\ \textsc{Seq 2} \qquad [\![c]\!] = \mathit{false}\ \frac{\mathrm{q} \xrightarrow{\omega} \mathrm{q}'}{c \to \mathrm{p} \diamond \mathrm{q} \xrightarrow{\omega} \mathrm{q}'}\ \textsc{Guard 6}$$

$$\mathrm{P}(\mathtt{d}_1 : \mathrm{D}_1, \cdots, \mathtt{d}_n : \mathrm{D}_n) \stackrel{\text{def}}{=} \mathrm{q}\ \frac{\mathrm{q}[\mathtt{d}_1 := t_1, \cdots, \mathtt{d}_n := t_n] \xrightarrow{\omega} \checkmark}{\mathrm{P}(t_1, \cdots, t_n) \xrightarrow{\omega} \checkmark}\ \textsc{Recursion 1}$$

$$\mathrm{P}(\mathtt{d}_1 : \mathrm{D}_1, \cdots, \mathtt{d}_n : \mathrm{D}_n) \stackrel{\text{def}}{=} \mathrm{q}\ \frac{\mathrm{q}[\mathtt{d}_1 := t_1, \cdots, \mathtt{d}_n := t_n] \xrightarrow{\omega} \mathrm{q}'}{\mathrm{P}(t_1, \cdots, t_n) \xrightarrow{\omega} \mathrm{q}'}\ \textsc{Recursion 2}$$

Table 2.6: mCRL2 inference rules (1/2)

$$e \in M_{\mathrm{D}} \ \frac{\mathrm{p}[\mathtt{d} := t_e] \xrightarrow{\omega} \checkmark}{\sum_{\mathtt{d}:\mathrm{D}} \mathrm{p} \xrightarrow{\omega} \checkmark}\ \textsc{Sum 1}$$

$$e \in M_{\mathrm{D}} \ \frac{\mathrm{p}[\mathtt{d} := t_e] \xrightarrow{\omega} \mathrm{p}'}{\sum_{\mathtt{d}:\mathrm{D}} \mathrm{p} \xrightarrow{\omega} \mathrm{p}'}\ \textsc{Sum 2}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \checkmark}{\mathrm{p} \mid\mid \mathrm{q} \xrightarrow{\omega} \mathrm{q}}\ \textsc{Par 1}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \mathrm{p}'}{\mathrm{p} \mid\mid \mathrm{q} \xrightarrow{\omega} \mathrm{p}' \mid\mid \mathrm{q}}\ \textsc{Par 2}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \mathrm{p}' \qquad \mathrm{q} \xrightarrow{\overline{\omega}} \mathrm{q}'}{\mathrm{p} \mid\mid \mathrm{q} \xrightarrow{\omega|\overline{\omega}} \mathrm{p}' \mid\mid \mathrm{q}'}\ \textsc{Par 3}$$

$$\frac{\mathrm{q} \xrightarrow{\omega} \checkmark}{\mathrm{p} \mid\mid \mathrm{q} \xrightarrow{\omega} \mathrm{p}}\ \textsc{Par 4}$$

$$\frac{\mathrm{q} \xrightarrow{\omega} \mathrm{q}'}{\mathrm{p} \mid\mid \mathrm{q} \xrightarrow{\omega} \mathrm{p} \mid\mid \mathrm{q}'}\ \textsc{Par 5}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \checkmark \qquad \mathrm{q} \xrightarrow{\overline{\omega}} \checkmark}{\mathrm{p} \mid\mid \mathrm{q} \xrightarrow{\omega|\overline{\omega}} \checkmark}\ \textsc{Par 6}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \mathrm{p}' \qquad \mathrm{q} \xrightarrow{\overline{\omega}} \checkmark}{\mathrm{p} \mid\mid \mathrm{q} \xrightarrow{\omega|\overline{\omega}} \mathrm{p}'}\ \textsc{Par 7}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \checkmark \qquad \mathrm{q} \xrightarrow{\overline{\omega}} \mathrm{q}'}{\mathrm{p} \mid\mid \mathrm{q} \xrightarrow{\omega|\overline{\omega}} \mathrm{q}'}\ \textsc{Par 8}$$

$$\underline{\omega} \in V \cup \{\tau\} \ \frac{\mathrm{p} \xrightarrow{\omega} \checkmark}{\nabla_V(\mathrm{p}) \xrightarrow{\omega} \checkmark}\ \textsc{Allow 1}$$

$$\underline{\omega} \in V \cup \{\tau\} \ \frac{\mathrm{p} \xrightarrow{\omega} \mathrm{p}'}{\nabla_V(\mathrm{p}) \xrightarrow{\omega} \nabla_V(\mathrm{p}')}\ \textsc{Allow 2}$$

$$\omega_{\{\}} \cap B = \emptyset \ \frac{\mathrm{p} \xrightarrow{\omega} \checkmark}{\partial_B(\mathrm{p}) \xrightarrow{\omega} \checkmark}\ \textsc{Block 1}$$

$$\omega_{\{\}} \cap B = \emptyset \ \frac{\mathrm{p} \xrightarrow{\omega} \mathrm{p}'}{\partial_B(\mathrm{p}) \xrightarrow{\omega} \partial_B(\mathrm{p}')}\ \textsc{Block 2}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \checkmark}{\rho_R(\mathrm{p}) \xrightarrow{R \bullet \omega} \checkmark}\ \textsc{Rename 1}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \mathrm{p}'}{\rho_R(\mathrm{p}) \xrightarrow{R \bullet \omega} \rho_R(\mathrm{p}')}\ \textsc{Rename 2}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \checkmark}{\Gamma_C(\mathrm{p}) \xrightarrow{\gamma_C(\omega)} \checkmark}\ \textsc{Comm 1}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \mathrm{p}'}{\Gamma_C(\mathrm{p}) \xrightarrow{\gamma_C(\omega)} \Gamma_C(\mathrm{p}')}\ \textsc{Comm 2}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \checkmark}{\tau_I(\mathrm{p}) \xrightarrow{\theta_I(\omega)} \checkmark}\ \textsc{Hide 1}$$

$$\frac{\mathrm{p} \xrightarrow{\omega} \mathrm{p}'}{\tau_I(\mathrm{p}) \xrightarrow{\theta_I(\omega)} \tau_I(\mathrm{p}')}\ \textsc{Hide 2}$$

Table 2.7: mCRL2 inference rules (2/2)

Finally, the rules in Tables 2.6 and 2.7 make use of $[\![\omega]\!]$-brackets, which denote the semantic value of ω. mCRL2 provides rules for how to apply these brackets are applied to expressions; see Definition 2.1.

Definition 2.1 The following rules apply to $[\![exp]\!]$:

$$\begin{aligned} [\![\tau]\!] &= \tau \\ [\![a(t_1,\cdots,t_n)]\!] &= a([\![t_1]\!],\cdots,[\![t_n]\!]) \\ [\![\alpha|\beta]\!] &= [\![\alpha]\!]|[\![\beta]\!] \end{aligned}$$

These rules were obtained from Definition 15.2.12 of the book that documents mCRL2 [24].

2.3 Special operators

The ∇_V, ∂_B, Γ_C, ρ_R, and τ_I are special operators that affect the actions that a process can perform. The following subsections describe these operators in greater detail.

Allow operator

The allow operator ∇_V only allows τ and actions and multi-actions that are in the set of multi-actions V from occurring in the operand. Its behavior follows ALLOW 1 and ALLOW 2 in Table 2.7.

Block operator

The block operator ∂_B (see BLOCK 1 and BLOCK 2) prevents actions that are in B from occurring in the operand (with the exception of τ). Any (multi-)action sharing an action with B is blocked – the blocked actions are not 'subtracted' from their multi-actions!

Communication operator

The communication operator Γ_C (see COMM 1 and COMM 2) modifies multi-actions that contain combinations of action labels by renaming those combinations to a specific action label. To this end, it requires a set of pairs $(a_1|\ldots|a_n \to b)$ where $a_1|\ldots|a_n$ is a multi-action without data and b an action label. This set is passed to an auxiliary function γ, which is defined as follows:

$$\begin{aligned} \gamma_\emptyset(\alpha) &= \alpha \\ \gamma_{C_1\cup C_2}(\alpha) &= \gamma_{C_1}(\gamma_{C_2}(\alpha)) \\ \gamma_{\{a_1|\ldots|a_n\to b\}}(\alpha) &= \begin{cases} b(\overline{d})|\gamma_C(\alpha \setminus (a_1(\overline{d})|\ldots|a_n(\overline{d}))) \\ \qquad \text{if } (a_1(\overline{d})|\ldots|a_n(\overline{d})) \sqsubseteq \alpha \text{ for some vector } \overline{d} \\ \alpha \text{ otherwise} \end{cases} \end{aligned}$$

where the operators $\setminus$ and $\sqsubseteq$ – although likely intuitively understood – are defined as

$$\begin{aligned}
\tau \setminus \alpha &= \tau \\
\alpha \setminus \tau &= \alpha \\
\alpha \setminus (\beta|\gamma) &= (\alpha \setminus \beta) \setminus \gamma \\
(\alpha|a(d_1,\cdots,d_n)) \setminus a(d_1,\cdots,d_n) &= \alpha \\
(\alpha|a(d_1,\cdots,d_n)) \setminus b(e_1,\cdots,e_n) &= (\alpha \setminus b(e_1,\cdots,e_n))|a(d_1,\cdots,d_n) \\
&\qquad \text{if } a \neq b \vee d_1 \neq e_1 \vee \cdots \vee d_n \neq e_n
\end{aligned}$$

and

$$\begin{aligned}
\tau \sqsubseteq \alpha &= \mathit{true} \\
a(d_1,\cdots,d_n) \sqsubseteq \tau &= \mathit{false} \\
\alpha|a(d_1,\cdots,d_n) \sqsubseteq \beta|a(d_1,\cdots,d_n) &= \alpha \sqsubseteq \beta \\
\alpha|a(d_1,\cdots,d_n) \sqsubseteq \beta|b(e_1,\cdots,e_n) &= (\alpha \setminus b(e_1,\cdots,e_n))|a(d_1,\cdots,d_n) \sqsubseteq \beta \\
&\qquad \text{if } a \neq b \vee d_1 \neq e_1 \vee \cdots \vee d_n \neq e_n
\end{aligned}$$

respectively.

Rename operator

The rename operator ρ_R renames all actions in the operand with the name a to b for all $a \to b \in R$ (where $a \neq \tau$, $b \neq \tau$). To this end, the following auxiliary function is used in Table 2.7:

$$\begin{aligned}
\rho_R(\tau) &= \tau \\
\rho_R(a(d_1,\cdots,d_n)) &= b(d_1,\cdots,d_n) \text{ if } a \to b \in R \text{ for some } b \\
\rho_R(a(d_1,\cdots,d_n)) &= a(d_1,\cdots,d_n) \text{ if } a \to b \notin R \text{ for all } b \\
\rho_R(\alpha|\beta) &= \rho_R(\alpha)|\rho_R(\beta)
\end{aligned}$$

hide operator

The hide operator τ_I replaces all actions in the operand with the name a by τ for all $a \in I$. Due to the property that $\omega|\tau = \tau|\omega = \omega$, the a is simply removed instead if a is part of a multi-action.

The BLOCK 1 and BLOCK 2 inference rules in Table 2.7 describe the behavior of the hide operator by using a function θ. This function is defined as follows:

$$\begin{aligned}
\theta_I(\tau) &= \tau \\
\theta_I(a(d_1,\cdots,d_n)) &= \tau \text{ if } a \in I \\
\theta_I(a(d_1,\cdots,d_n)) &= a(d_1,\cdots,d_n) \text{ if } a \notin I \\
\theta_I(\alpha|\beta) &= \theta_I(\alpha)|\theta_I(\beta)
\end{aligned}$$

2.4 mCRL2 data language

This section gives some details of how the data expression language of mCRL2 works.

Sort declarations

Types are called 'sorts' in mCRL2. There are predefined sorts, such as `Bool` (boolean) and `Nat` (natural numbers), but also custom sorts, which must be declared. With regard to this project, *structured sorts* are particularly relevant.

Structured sorts are declared as a list of constructors with arguments. Consider the following grammar:

$$
\begin{aligned}
\mathit{sort} &::= \textbf{sort}\ \mathit{sortName}\ \texttt{=}\ \mathit{structSort} \quad | \quad \cdots \\
\mathit{structSort} &::= \textbf{struct}\ \mathit{structSortConstructor}_1 \quad | \quad \cdots \quad | \quad \mathit{structSortConstructor}_n \\
\mathit{structSortConstructor} &::= \mathit{constructorName}(\mathit{argName}_1 : \mathit{argSort}_1, \cdots, \mathit{argName}_n : \mathit{argSort}_n)
\end{aligned}
$$

For example, the following code snippet declares a binary tree:

```
1  sort
2    BinTree
3      = struct Leaf | Node(value: Nat, leftChild: BinTree, rightChild: BinTree);
```

In order to create an instance of a structured sort, simply write one of the constructor with the appropriate number of parameter values, each of a compatible sort. Below, a binary tree with 3 nodes is defined:

```
1  Node(1, Node(2, Leaf, Leaf), Node(3, Leaf, Leaf))
```

To obtain the value of one of the arguments of a specific instance, use the name of the argument as a 'getter' function:

```
1  value(Node(9, Leaf, Leaf))
```

The data expression in the example above yields `9`.

Mappings and equations

mCRL2's data language is a *functional* language. Functions in mCRL2 are defined through a *mapping* and a number of *equations*. The mapping specifies the function sort of the function, and the equations specify how the function should be evaluated.

The following code snippet defines a function that changes the value in a `BinTree` to another value:

```
1  map
2    updateNodeValues: BinTree # Nat # Nat -> BinTree;
3  eqn
4    updateNodeValues(Leaf, i, j)
5      = Leaf;
6    updateNodeValues(Node(i, left, right), i, j)
7      = Node(j, updateNodeValues(left, i, j), updateNodeValues(right, i, j));
8    updateNodeValues(Node(v, left, right), i, j)
9      = Node(v, updateNodeValues(left, i, j), updateNodeValues(right, i, j));
```

The mapping specifies that the function takes a `BinTree` and two natural numbers as input and produces a `BinTree` as output. Three equations define a standard recursive algorithm: the first equation is the base case of the recursion; the second equation changes the value of the current node from `i` to `j` and then continues with the child nodes; and the third equation does the same but maintains the value of the current node.

2.5 mCRL2 examples

Listings 2.5 to 2.7 give an example of a system specification in mCRL2. Listing 2.5 first specifies some of the required actions in the `act` block (lines 1 to 3). It then defines the two subsystems that constitute the television system as two separate processes: the `Volume` subsystem (lines 6 to 8) allows the volume of the television to be turned up or down within the range 0 to 50, and the `Channel` subsystem (lines 10 to 12) allows the channel (0 to 99) of the television to be changed by twice pressing a `0` to `9` button. The two subsystems can be easily composed using parallel composition (line 14). The `init` section specifies the first state of the entire television system.

```
1  act
2    volume_down, volume_up;
3    press: Integer;
4
5  proc
6    Volume(volume: Integer) =
7        volume_down . Volume(if(volume > 0, volume - 1, 0))
8      + volume_up . Volume(if(volume < 50, volume + 1, 50));
9
10   Channel(channel: Integer) =
11     sum n1:{0..9} . press(n1) .
12       sum n2:{0..9} . press(n2) . Channel(n1 * 10 + n2);
13
14   Television = Channel(1) || Volume(25);
15
16 init
17   Television;
```

Listing 2.5: Simple mCRL2 television example.

Listing 2.6 extends the television by adding a mute subsystem. This subsystem allows the user to turn the volume of the television on or off in its entirety by pressing the 'mute' button (line 6). The volume of the television is also turned back on when the user changes the volume of the television (line 7 and 8).

```
1  act
2    mute;
3
4  proc
5    Mute(muted: Boolean) =
6        mute . Mute(!muted)
7      + volume_down . Mute(false)
8      + volume_up . Mute(false);
9
10   Television2 = Channel(1) || Volume(25) || Mute(false);
```

Listing 2.6: Naive extension of the simple mCRL2 television example.

Naively, one might be tempted to add the new subsystem to the television by using another parallel composition as in line 10 of Listing 2.6. However, in that case the `volume_up` and `volume_down` actions of the `Volume` and `Mute` subsystems would be independent, and it is therefore possible that the television model changes the volume without unmuting.

Instead, one should make use of the communication operator Γ and the block operator ∂, which are discussed in subsections 2.3 and 2.3, respectively. Listing 2.7 shows how this can be accomplished: with the communication operator, the `volume_up` and `volume_down` actions of the `Volume` and `Mute` subsystems are merged (and renamed to `synced_volume_up` and `synced_volume_down`, respectively) and then prevents any independent occurrences of `volume_up` and `volume_down` with the block operator.

```
1  act
2    synced_volume_down, synced_volume_up;
3
4  proc
5    Television3 = Channel(1) || ∂{volume_up,volume_down}
6        Γ{volume_up|volume_up→synced_volume_up,volume_down|volume_down→synced_volume_down}
7          (Volume(25) || Mute(false));
```

Listing 2.7: Intended extension of the simple mCRL2 television example.

3 Translation function

The project was continued by designing a *translation function* from AWN to mCRL2. The function has been defined by a list of *translation rules* which have been categorized according to one of two purposes, namely for translating AWN's sequential process expressions or for translating higher-level process expressions of AWN. The translation rules for both purposes are discussed in the following sections.

3.1 Translating sequential process expressions

The rules for translating sequential process expressions can be found in Table 2.8.

$$\mathrm{T}_V(\xi, \mathbf{broadcast}(ms).\mathrm{p}) = \textstyle\sum_{\mathtt{D}:\mathrm{T}(\mathrm{Set(IP)})} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p}) \text{ where } \mathtt{D} \notin \mathrm{T}(V) \quad \text{T1}$$

$$\mathrm{T}_V(\xi, \mathbf{groupcast}(dests, ms).\mathrm{p}) = \textstyle\sum_{\mathtt{D}:\mathrm{T}(\mathrm{Set(IP)})} \mathbf{cast}(\mathrm{T}_\xi(dests), \mathtt{D}, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p}) \text{ where } \mathtt{D} \notin \mathrm{T}(V) \quad \text{T2}$$

$$\mathrm{T}_V(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q}) = \mathbf{cast}(\{\mathrm{T}_\xi(dest)\}, \{\mathrm{T}_\xi(dest)\}, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p}) + \neg\mathbf{uni}(\{\mathrm{T}_\xi(dest)\}, \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{q}) \quad \text{T3}$$

$$\mathrm{T}_V(\xi, \mathbf{send}(\mathrm{T}_\xi(ms)).\mathrm{p}) = \mathbf{send}(\emptyset, \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p}) \quad \text{T4}$$

$$\mathrm{T}_V(\xi, \mathbf{deliver}(data).\mathrm{p}) = \textstyle\sum_{\mathtt{ip}:\mathrm{T}(\mathrm{IP})} \mathbf{del}(\mathtt{ip}, \mathrm{T}_\xi(data)).\mathrm{T}_V(\xi, \mathrm{p}) \text{ where } \mathtt{ip} \notin \mathrm{T}(V) \quad \text{T5}$$

$$\mathrm{T}_V(\xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p}) = \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set(IP)}),\mathrm{T}(\mathtt{msg}):\mathrm{T}(\mathrm{MSG})} \mathbf{receive}(\mathtt{D},\mathtt{D'},\mathrm{T}(\mathtt{msg})).\mathrm{T}_{V\cup\{\mathtt{msg}\}}(\xi^{\setminus\mathtt{msg}}, \mathrm{p}) \text{ where } \mathtt{D},\mathtt{D'} \notin \mathrm{T}(V) \quad \text{T6}$$

$$\mathrm{T}_V(\xi, [\![\mathtt{var} := exp]\!]\,\mathrm{p}) = \textstyle\sum_{\mathtt{tmp}:\mathrm{sort}(\mathrm{T}(\mathtt{var}))} (\mathtt{tmp} = \mathrm{T}_\xi(exp)) \to \sum_{\mathrm{T}(\mathtt{var}):\mathrm{sort}(\mathrm{T}(\mathtt{var}))} (\mathrm{T}(\mathtt{var}) = \mathtt{tmp}) \to \mathbf{t}(\emptyset, \emptyset, \mathtt{msg}_{\,dummy}).\mathrm{T}_{V\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}}, \mathrm{p}) \text{ where } \mathtt{tmp} \notin \mathrm{T}(V) \quad \text{T7}$$

$$\mathrm{T}_V(\xi, \mathrm{X}(exp_1, \cdots, exp_n)) = \mathrm{X}(\mathrm{T}_\xi(exp_1), \cdots, \mathrm{T}_\xi(exp_n)) \quad \text{T8}$$

$$\text{where } \mathrm{T}(\mathrm{X}(\mathtt{var}_1, \cdots, \mathtt{var}_n) \stackrel{\text{def}}{=} \mathrm{p}) = \mathrm{X}(\mathrm{T}(\mathtt{var}_1) : \mathrm{sort}(\mathrm{T}_\xi(exp_1)), \cdots, \mathrm{T}(\mathtt{var}_n) : \mathrm{sort}(\mathrm{T}_\xi(exp_n))) \stackrel{\text{def}}{=} \mathrm{T}_{\{\mathtt{var}_1,\cdots,\mathtt{var}_n\}}(\emptyset, \mathrm{p})$$

$$\mathrm{T}_V(\xi, \mathrm{p}+\mathrm{q}) = \mathrm{T}_V(\xi, \mathrm{p}) + \mathrm{T}_V(\xi, \mathrm{q}) \quad \text{T9}$$

$$\mathrm{T}_V(\xi, [\phi]\mathrm{p}) = \textstyle\sum_{\mathrm{T}(\mathrm{FV}(\phi)\setminus V)} \mathrm{T}_\xi(\phi) \to \mathbf{t}(\emptyset, \emptyset, \mathtt{msg}_{\,dummy}).\mathrm{T}_{V\cup\mathrm{FV}(\phi)}(\xi, \mathrm{p}) \quad \text{T10}$$

$$\mathrm{T}(\mathrm{X}(\mathtt{var}_1, \cdots, \mathtt{var}_n) \stackrel{\text{def}}{=} \mathrm{p}) = \mathrm{X}(\mathrm{T}(\mathtt{var}_1) : \mathrm{sort}(\mathrm{T}(\mathtt{var}_1)), \cdots, \mathrm{T}(\mathtt{var}_n) : \mathrm{sort}(\mathrm{T}(\mathtt{var}_n))) \stackrel{\text{def}}{=} \mathrm{T}_{\{\mathtt{var}_1,\cdots,\mathtt{var}_n\}}(\emptyset, \mathrm{p}) \quad \text{T11}$$

Table 2.8: Rules for translating AWN's sequential process expressions

The translation rules in Table 2.8 use $\mathrm{T}_V(\xi, \mathrm{p})$ as the signature of the translation function, where p is the sequential process expression to be translated, ξ is a *valuation* (a mapping from some variables to semantic values), and V is a set of AWN variables (those that are assigned up to this point, to be precise). $\mathrm{T}_V(\xi, \mathrm{p})$ is only defined if $\mathrm{DOM}(\xi) \cup \mathrm{FV}(\mathrm{p}) \subseteq V$ where $\mathrm{FV}(\mathrm{p})$ are the free variables in p. Occasionally, the bindings for a specific variable v is removed from ξ by writing $\xi^{\setminus \mathtt{v}}$ which is defined as ξ restricted to $\mathrm{DOM}(\xi) \setminus \{\mathtt{v}\}$.

ξ is carried around exclusively for the benefit of the $\mathrm{T}_\xi(exp)$ function, which translates data expressions. The definition of $\mathrm{T}_\xi(exp)$ is relatively complex:

$$\mathrm{T}_\xi(exp) \stackrel{\text{def}}{=} \mathrm{T}(exp)\left[\mathrm{T}(\mathtt{x}) := t_{\mathrm{U}(y)} \;\middle|\; \mathtt{x} := y \in \xi\right]$$

where the bijective function $\mathrm{U}(y)$ translates a semantic value of AWN called y to the corresponding semantics value of mCRL2; where the Axiom of Choice is applied to the selection of t_Y (given a

particular semantic value Y, the same syntactic representation t_Y will always be selected, which holds throughout the translation); and where the function $\mathtt{T}(exp)$ translates a syntactic AWN data expression *exp* to a syntactic mCRL2 data expression. In order to abstract from the translation of data expressions at least to some extend, $\mathtt{T}(exp)$ is not further defined but simply assumed to exist with the following properties:

(i) $\mathtt{T}(exp)$ is a total function;
(ii) $\mathtt{T}(exp)$ behaves in such a way that $\forall exp, \xi \,.\, \xi(exp)$ is defined $\Rightarrow [\![\mathtt{T}_\xi(exp)]\!] = \mathtt{U}(\xi(exp))$;
(iii) $exp = \mathtt{v} \Leftrightarrow \mathtt{T}(exp) = \mathtt{v'}$ where $\mathtt{v}$ is an AWN variable and $\mathtt{v'}$ is an mCRL2 variable;
(iv) $\forall exp \,.\, \mathrm{FV}(\mathtt{T}(exp)) = \{\mathtt{T}(\mathtt{v}) \mid \mathtt{v} \in \mathrm{FV}(exp)\}$; and
(v) $\mathtt{T}(\mathtt{var}_1) = \mathtt{T}(\mathtt{var}_2) \Leftrightarrow \mathtt{var}_1 = \mathtt{var}_2$.

The following subsections will elaborate on each of the translation rules in Table 2.8.

Rule T1: Broadcast

$$\mathrm{T}_V(\xi, \mathbf{broadcast}(ms).\mathrm{p}) = \textstyle\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set(IP)})} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathtt{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p}) \text{ where } \mathtt{D} \notin \mathtt{T}(V) \quad \mathrm{T1}$$

This rule defines how a **broadcast** action is translated to mCRL2. The resulting **cast** action has three parameters and is then immediately followed by the translation of p.

The first parameter of the **cast** action is the set of addresses of nodes that are the intended recipients of the message *ms*. Since a **broadcast** action always attempts to reach all nodes with range, the value of this parameter is the universe of all node addresses ($\mathscr{U}_{\mathrm{IP}}$).

The second parameter is the set of actual recipients. The sequential process does not have direct access to which nodes are within range, however, and the value of this parameter is therefore unknown. This has been solved by adding $\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set(IP)})}$, the resulting mCRL2 expression produces a **cast** action for *all* possible values – superfluous actions will be eliminated at a later stage. The side condition of T1 ensures that D is fresh.

The third parameter is the message that is sent translated to an mCRL2 expression by means of the $\mathtt{T}_\xi(exp)$ function.

Rule T2: Groupcast

$$\mathrm{T}_V(\xi, \mathbf{groupcast}(dests, ms).\mathrm{p}) = \textstyle\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set(IP)})} \mathbf{cast}(\mathtt{T}_\xi(dests), \mathtt{D}, \mathtt{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p}) \text{ where } \mathtt{D} \notin \mathtt{T}(V) \quad \mathrm{T2}$$

This rule defines how a **groupcast** action is translated to mCRL2. Just like for the translation of **broadcast**, the resulting **cast** action has three parameters and is then immediately followed by the translation of p. The second and third parameter are also translated the same as for **broadcast** (with the side condition of T2 ensuring that D is fresh), but the first parameter is not: rather than $\mathscr{U}_{\mathrm{IP}}$ the set of destinations specified by the user is inserted. The first parameter must contain the node addresses of the *intended* recipients, after all.

Rule T3: Unicast

$$\mathrm{T}_V(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q}) = \mathbf{cast}(\{\mathrm{T}_\xi(dest)\}, \{\mathrm{T}_\xi(dest)\}, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p}) + \neg\mathbf{uni}(\{\mathrm{T}_\xi(dest)\}, \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{q}) \quad \text{T3}$$

The translation of the **unicast** action produces two possible outcomes: if the destination node is within range, a **cast** action can occur, sending message *ms* to the destination specified by the user and continuing by executing p; and it the destination node is *not* within range, a ¬**uni** action can occur, followed by the execution of q.

Rule T4: Send

$$\mathrm{T}_V(\xi, \mathbf{send}(\mathrm{T}_\xi(ms)).\mathrm{p}) = \mathbf{send}(\emptyset, \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p}) \quad \text{T4}$$

The translation rule for the **send** action is the simplest of them all: it produces a single action by the same name and with three parameters. The first and second parameter of this action are simply initialized with an empty set of node addresses (these parameters are strictly necessary because the signature of **send** must match the signature of the **receive** action) and the third parameter carries the message *ms* that is sent.

Rule T5: Deliver

$$\mathrm{T}_V(\xi, \mathbf{deliver}(data).\mathrm{p}) = \textstyle\sum_{\mathtt{ip}:\mathrm{T}(\mathrm{IP})} \mathbf{del}(\mathtt{ip}, \mathrm{T}_\xi(data)).\mathrm{T}_V(\xi, \mathrm{p}) \text{ where } \mathtt{ip} \notin \mathrm{T}(V) \quad \text{T5}$$

The **deliver** action allows the contents of messages (data) to 'leave' a network. Such an event involves two items of information in AWN: the data that is leaving the network, and the node where the data is leaving. Accordingly, the rule above produces a **del** action with these two items as its parameters.

However, a sequential process is unaware of the node on which it is running. The chosen solution for this problem is that a **del** action is produced for every possible node address `ip` – superfluous actions will be eliminated at a later stage. The side condition of the rule ensures that the variable `ip` is fresh.

Rule T6: Receive

$$\mathrm{T}_V(\xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p}) = \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set(IP)}),\mathrm{T}(\mathtt{msg}):\mathrm{T}(\mathrm{MSG})} \mathbf{receive}(\mathtt{D}, \mathtt{D'}, \mathrm{T}(\mathtt{msg})).\mathrm{T}_{V\cup\{\mathtt{msg}\}}(\xi^{\setminus\mathtt{msg}}, \mathrm{p}) \text{ where } \mathtt{D},\mathtt{D'} \notin \mathrm{T}(V) \quad \text{T6}$$

When receiving a message, a node at the level of a sequential process expression is unaware of three things: the intended recipients of the message, `D`; the actual recipients of the message, `D'`; and the contents of the message. A sequential process therefore considers all possible **receive** actions that could occur, and relies on the composition with other parts of the specification to eliminate the superfluous ones.

The process p is translated with the additional information that `msg` is now assigned, which is of course because it contains the contents of the received message. At the same time, p receives a ξ from which `msg` has been *removed* – otherwise an old value of `msg` could continue to exist in p even though it has been overwritten by the **receive**!

Rule T7: Assignment

$$\mathrm{T}_V(\xi, [\![\mathtt{var} := exp]\!]\ \mathrm{p}) = \textstyle\sum_{\mathtt{tmp}:\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathtt{tmp} = \mathrm{T}_\xi(exp)) \to \sum_{\mathrm{T}(\mathtt{var}):\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathrm{T}(\mathtt{var}) = \mathtt{tmp}) \to \quad \text{T7}$$
$$\mathbf{t}(\emptyset, \emptyset, \mathtt{msg}_{\,dummy}).\mathrm{T}_{V\cup\{\mathtt{var}\}}(\xi^{\setminus \mathtt{var}}, \mathrm{p}) \text{ where } \mathtt{tmp} \notin \mathrm{T}(V)$$

mCRL2 does not provide its own syntax for assignments other than when parameter values are assigned when a new process is instantiated. A more improvised solution is used here, namely one where a $\sum$-operator and a guard ensure that a specific variable matches the assigned value (see the innermost/rightmost $\sum$-operator). The assigned value is fixed beforehand by a similar construction (see the outermost/leftmost $\sum$-operator) in order to allow the target variable to be used in the computation of the assigned value. The variable in which the assigned value is stored, `tmp`, is fresh as a result of the side condition of the rule.

One might wonder about the three placeholder parameters of the $\mathbf{t}$, or indeed about the reason why $\mathbf{t}$ was used rather than τ in the first place. The answer to the first riddle is that actions that *do* carry three useful parameters will be renamed to $\mathbf{t}$ at a later stage, and one of the requirements of mCRL2 for renaming an action a to b is that the action signatures of a and b are identical. The answer to the second riddle is that the concurrent behavior of the τ action in mCRL2 does not match the behavior of the τ action in AWN, and that it is necessary to treat τ as a 'regular' action until the network level has been reached.

Finally, process p is translated. It receives the information that in addition to the variables in V the variable `var` is assigned. `var` is removed from ξ so that a potential old value of `var` cannot be used in the translation of p.

Rule T8: Process recursion

Process 'calls' are simply translated by referencing the mCRL2 counterpart of the named AWN process that is being 'called' and providing it with the translation of each of the parameter values that were provided in AWN:

$$\mathrm{T}_V(\xi, \mathrm{X}(exp_1, \cdots, exp_n)) = \mathrm{X}(\mathrm{T}_\xi(exp_1), \cdots, \mathrm{T}_\xi(exp_n)) \quad \text{T8}$$
$$\text{where } \mathrm{T}(\mathrm{X}(\mathtt{var}_1, \cdots, \mathtt{var}_n) \stackrel{\text{def}}{=} \mathrm{p}) = \mathrm{X}(\mathrm{T}(\mathtt{var}_1) : \mathrm{sort}(\mathrm{T}_\xi(exp_1)), \cdots, \mathrm{T}(\mathtt{var}_n) : \mathrm{sort}(\mathrm{T}_\xi(exp_n))) \stackrel{\text{def}}{=} \mathrm{T}_{\{\mathtt{var}_1, \cdots, \mathtt{var}_n\}}(\emptyset, \mathrm{p})$$

Rule T9: Choice

A choice between two branches in AWN is translated directly to a choice between two summands in mCRL2:

$$\mathrm{T}_V(\xi, \mathrm{p}+\mathrm{q}) = \mathrm{T}_V(\xi, \mathrm{p}) + \mathrm{T}_V(\xi, \mathrm{q}) \quad \text{T9}$$

Rule T10: Guard

$$\mathrm{T}_V(\xi, [\phi]\mathrm{p}) = \sum_{\mathrm{T}(\mathrm{FV}(\phi) \setminus V)} \mathrm{T}_\xi(\phi) \rightarrow \mathbf{t}(\emptyset, \emptyset, \mathtt{msg}_{\mathit{dummy}}).\mathrm{T}_{V \cup \mathrm{FV}(\phi)}(\xi, \mathrm{p}) \quad \text{T10}$$

The exclusive purpose of carrying around the set of assigned variables V is to determine which variables are assigned as part of a guard action: any variable that is free in the guard condition ϕ and *not* in V (and is therefore unassigned at the moment of execution) will be added as a quantifier to the $\sum$-operator. Obviously, this means that the subsequent process p must find $\mathrm{FV}(\phi)$ – all free variables in ϕ – in its index parameter V.

One might wonder about the three placeholder parameters of the **t**, or indeed about the reason why **t** was used rather than τ. The answers to these riddles are given in the description of translation rule T7.

Rule T11: Process definition

$$\mathrm{T}(\mathrm{X}(\mathtt{var}_1, \cdots, \mathtt{var}_n) \stackrel{\text{def}}{=} \mathrm{p}) = \mathrm{X}(\mathrm{T}(\mathtt{var}_1) : \mathrm{sort}(\mathrm{T}(\mathtt{var}_1)), \cdots, \mathrm{T}(\mathtt{var}_n) : \mathrm{sort}(\mathrm{T}(\mathtt{var}_n))) \stackrel{\text{def}}{=} \mathrm{T}_{\{\mathtt{var}_1, \cdots, \mathtt{var}_n\}}(\emptyset, \mathrm{p}) \quad \text{T11}$$

The translation of an AWN process definition X has the same name in mCRL2. Furthermore, each parameter of the mCRL2 definition is a translated parameter of X, and the body of the mCRL2 process definition is the translated body of X. When the body of X is translated, it is assumed that all process parameters are in the set of assigned variables, V, and that the valuation ξ is empty (a process definition 'knows' that the process parameters are assigned, but not what the semantic values of the process parameters are – this is established at execution time).

3.2 Translating higher-level process expressions

The rules for translating sequential process expressions can be found in Table 2.9.

$$\mathrm{T}(\xi, \mathrm{p}) = \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \quad \text{T12}$$

$$\mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q}) = \nabla_V \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \quad \text{T13}$$

$$\text{where } V = \{\mathbf{t}, \mathbf{cast}, \neg\mathbf{uni}, \mathbf{send}, \mathbf{del}, \mathbf{receive}\}$$

$$\mathrm{T}(ip : \mathrm{P} : R) = \nabla_V \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \quad \text{T14}$$

$$\text{where } V = \{\mathbf{t}, \mathbf{starcast}, \mathbf{arrive}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}$$

$$\text{where } C = \{\mathbf{cast}|\overline{\mathbf{cast}} \to \mathbf{starcast}, \neg\mathbf{uni}|\overline{\neg\mathbf{uni}} \to \mathbf{t}, \mathbf{del}|\overline{\mathbf{del}} \to \mathbf{deliver}, \mathbf{receive}|\overline{\mathbf{receive}} \to \mathbf{arrive}\}$$

$$\begin{aligned}\text{where } \mathrm{G}(\mathtt{ip},\mathtt{R}) = {} & \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})}(\mathtt{R} \cap \mathtt{D} = \mathtt{D'}) \to \overline{\mathbf{cast}}(\mathtt{D},\mathtt{D'},\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R}) \\ & + \textstyle\sum_{\mathtt{d}:\mathrm{T}(\mathrm{IP}),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})}(\mathtt{d} \notin \mathtt{R}) \to \overline{\neg\mathbf{uni}}(\{\mathtt{d}\},\emptyset,\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R}) \\ & + \textstyle\sum_{\mathtt{data}:\mathrm{DATA}} \overline{\mathbf{del}}(\mathtt{ip},\mathtt{data}).\mathrm{G}(\mathtt{ip},\mathtt{R}) \\ & + \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})}(\mathtt{ip} \in \mathtt{D'}) \to \overline{\mathbf{receive}}(\mathtt{D},\mathtt{D'},\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R}) \\ & + \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})}(\mathtt{ip} \notin \mathtt{D'}) \to \mathbf{arrive}(\mathtt{D},\mathtt{D'},\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R}) \\ & + \textstyle\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{connect}(\mathtt{ip},\mathtt{ip'}).\mathrm{G}(\mathtt{ip},\mathtt{R} \cup \{\mathtt{ip'}\}) \\ & + \textstyle\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{connect}(\mathtt{ip'},\mathtt{ip}).\mathrm{G}(\mathtt{ip},\mathtt{R} \cup \{\mathtt{ip'}\}) \\ & + \textstyle\sum_{\mathtt{ip'},\mathtt{ip''}:\mathrm{T}(\mathrm{IP})}(\mathtt{ip} \notin \{\mathtt{ip'},\mathtt{ip''}\}) \to \mathbf{connect}(\mathtt{ip'},\mathtt{ip''}).\mathrm{G}(\mathtt{ip},\mathtt{R}) \\ & + \textstyle\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{disconnect}(\mathtt{ip},\mathtt{ip'}).\mathrm{G}(\mathtt{ip},\mathtt{R} \setminus \{\mathtt{ip'}\}) \\ & + \textstyle\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{disconnect}(\mathtt{ip'},\mathtt{ip}).\mathrm{G}(\mathtt{ip},\mathtt{R} \setminus \{\mathtt{ip'}\}) \\ & + \textstyle\sum_{\mathtt{ip'},\mathtt{ip''}:\mathrm{T}(\mathrm{IP})}(\mathtt{ip} \notin \{\mathtt{ip'},\mathtt{ip''}\}) \to \mathbf{disconnect}(\mathtt{ip'},\mathtt{ip''}).\mathrm{G}(\mathtt{ip},\mathtt{R})\end{aligned}$$

$$\mathrm{T}(\mathrm{M} \,||\, \mathrm{N}) = \rho_R \nabla_V \Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}} \Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \quad \text{T15}$$

$$\text{where } R = \{\mathbf{a} \to \mathbf{arrive}, \mathbf{c} \to \mathbf{connect}, \mathbf{d} \to \mathbf{disconnect}, \mathbf{s} \to \mathbf{starcast}\}$$

$$\text{where } V = \{\mathbf{a}, \mathbf{c}, \mathbf{d}, \mathbf{deliver}, \mathbf{s}, \mathbf{t}\}$$

$$\text{where } C = \{\mathbf{starcast}|\mathbf{arrive} \to \mathbf{s}, \mathbf{connect}|\mathbf{connect} \to \mathbf{c}, \mathbf{disconnect}|\mathbf{disconnect} \to \mathbf{d}\}$$

$$\mathrm{T}([\,\mathrm{M}\,]) = \nabla_V \rho_{\{\mathbf{starcast}\to\mathbf{t}\}} \Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \quad \text{T16}$$

$$\text{where } V = \{\mathbf{t}, \mathbf{newpkt}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}$$

$$\text{where } C = \{\overline{\mathbf{newpkt}}|\mathbf{arrive} \to \mathbf{newpkt}\}$$

$$\text{where } \mathrm{H} = \textstyle\sum_{\mathtt{ip}:\mathrm{T}(\mathrm{IP}),\mathtt{data}:\mathrm{T}(\mathrm{DATA}),\mathtt{dest}:\mathrm{T}(\mathrm{IP})} \overline{\mathbf{newpkt}}(\{\mathtt{ip}\},\{\mathtt{ip}\},\mathtt{newpkt}(\mathtt{data},\mathtt{dest})).\mathrm{H}$$

Table 2.9: Rules for translating AWN's higher-level process expressions

The translation rules in Table 2.9 use $\mathrm{T}(\mathrm{p})$ as the signature of the translation function, where p is the process expression to be translated. Clearly, $\mathrm{T}(\mathrm{p})$ no longer has the ξ parameter (ξ, p in rule T12 forms a single parameter, the state of a sequential process in AWN to be precise), and it does not carry a set of assigned variables V anymore. At this level of the AWN specification, syntactic data expressions have disappeared.

The following subsections will elaborate on each of the translation rules in Table 2.9.

Rule T12: Sequential process

This rule allows the rules from Table 2.9 to make use of the rules in Table 2.8:

$$\mathrm{T}(\xi, \mathrm{p}) = \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \quad \text{T12}$$

Note that ξ, p forms a single parameter – namely the state of a sequential process in AWN – and that this state is separated into its two components, the valuation ξ and the process syntax p. Furthermore, the set of assigned variables V is assigned with the domain of ξ (in the implementation, this is almost never of any consequence; when proving the correctness of $\mathrm{T}(\mathrm{p})$, however, this becomes a different story).

Rule T13: Parallel processes

$$\mathrm{T}(\mathrm{P} \lll \mathrm{Q}) = \nabla_V \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \mid\mid \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \quad \text{T13}$$
$$\text{where } V = \{\mathbf{t}, \mathbf{cast}, \neg\mathbf{uni}, \mathbf{send}, \mathbf{del}, \mathbf{receive}\}$$

Two AWN processes in parallel are translated by applying several mCRL2 operators in sequence:

1. It must be possible in the final mCRL2 operator to differentiate between the **receive** actions that occur in $\mathrm{T}(\mathrm{P})$ and those that occur in $\mathrm{T}(\mathrm{Q})$, as well as between the **send** actions that occur in $\mathrm{T}(\mathrm{P})$ and those that occur in $\mathrm{T}(\mathrm{Q})$. The **receive** actions of $\mathrm{T}(\mathrm{P})$ are therefore renamed to **r** and the **send** actions of $\mathrm{T}(\mathrm{Q})$ to **s**.
2. The resulting processes are put in parallel.
3. Occurrences of $\mathbf{r}|\mathbf{s}$ action labels are replaced by a **t** action label (this is why **t** has been given three parameters).
4. Only a specific selection of actions is allowed to occur. This blocks all mCRL2 multi-actions that are impossible in AWN (such as $\mathbf{t}|\mathbf{t}$). In addition, all remaining **r** actions are blocked (from now on, $\mathrm{T}(\mathrm{P})$ can only **receive** messages if they arrive via $\mathrm{T}(\mathrm{Q})$, but $\mathrm{T}(\mathrm{Q})$ can still **receive**) as well as all remaining **s** actions (**send** actions of $\mathrm{T}(\mathrm{Q})$ must have synchronized before now, but $\mathrm{T}(\mathrm{P})$ can still **send**).

Rule T14: Node

$$\begin{aligned}
\mathrm{T}(ip:\mathrm{P}:R) &= \nabla_V \Gamma_C(\mathrm{T}(\mathrm{P}) \mid\mid \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \quad \text{T14}\\
&\quad \text{where } V = \{\mathbf{t}, \mathbf{starcast}, \mathbf{arrive}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}\\
&\quad \text{where } C = \{\mathbf{cast}|\overline{\mathbf{cast}} \to \mathbf{starcast}, \neg\mathbf{uni}|\overline{\neg\mathbf{uni}} \to \mathbf{t}, \mathbf{del}|\overline{\mathbf{del}} \to \mathbf{deliver}, \mathbf{receive}|\overline{\mathbf{receive}} \to \mathbf{arrive}\}\\
\text{where } \mathrm{G}(\mathtt{ip},\mathtt{R}) &= \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (\mathtt{R} \cap \mathtt{D} = \mathtt{D'}) \to \overline{\mathbf{cast}}(\mathtt{D},\mathtt{D'},\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R})\\
&\quad + \textstyle\sum_{\mathtt{d}:\mathrm{T}(\mathrm{IP}),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (\mathtt{d} \notin \mathtt{R}) \to \overline{\neg\mathbf{uni}}(\{\mathtt{d}\},\emptyset,\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R})\\
&\quad + \textstyle\sum_{\mathtt{data}:\mathrm{DATA}} \overline{\mathbf{del}}(\mathtt{ip},\mathtt{data}).\mathrm{G}(\mathtt{ip},\mathtt{R})\\
&\quad + \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (\mathtt{ip} \in \mathtt{D'}) \to \overline{\mathbf{receive}}(\mathtt{D},\mathtt{D'},\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R})\\
&\quad + \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (\mathtt{ip} \notin \mathtt{D'}) \to \mathbf{arrive}(\mathtt{D},\mathtt{D'},\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R})\\
&\quad + \textstyle\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{connect}(\mathtt{ip},\mathtt{ip'}).\mathrm{G}(\mathtt{ip},\mathtt{R} \cup \{\mathtt{ip'}\})\\
&\quad + \textstyle\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{connect}(\mathtt{ip'},\mathtt{ip}).\mathrm{G}(\mathtt{ip},\mathtt{R} \cup \{\mathtt{ip'}\})\\
&\quad + \textstyle\sum_{\mathtt{ip'},\mathtt{ip''}:\mathrm{T}(\mathrm{IP})} (\mathtt{ip} \notin \{\mathtt{ip'},\mathtt{ip''}\}) \to \mathbf{connect}(\mathtt{ip'},\mathtt{ip''}).\mathrm{G}(\mathtt{ip},\mathtt{R})\\
&\quad + \textstyle\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{disconnect}(\mathtt{ip},\mathtt{ip'}).\mathrm{G}(\mathtt{ip},\mathtt{R} \setminus \{\mathtt{ip'}\})\\
&\quad + \textstyle\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{disconnect}(\mathtt{ip'},\mathtt{ip}).\mathrm{G}(\mathtt{ip},\mathtt{R} \setminus \{\mathtt{ip'}\})\\
&\quad + \textstyle\sum_{\mathtt{ip'},\mathtt{ip''}:\mathrm{T}(\mathrm{IP})} (\mathtt{ip} \notin \{\mathtt{ip'},\mathtt{ip''}\}) \to \mathbf{disconnect}(\mathtt{ip'},\mathtt{ip''}).\mathrm{G}(\mathtt{ip},\mathtt{R})
\end{aligned}$$

This rule elevates processes to the network level by synchronizing them with a process G that provides the address of the node on which they run and the set of addresses of nodes that are within transmission range. Through this synchronization, the actions **cast**, ¬**uni**, **del**, and **receive** of a process $\mathrm{T}(\mathrm{P})$ are forced to have certain parameter values, which eliminates many – but not all – of the superfluous mCRL2 actions that are impossible in AWN. G also adds independent node behavior, namely **arrive** (which allows a node to synchronize the arrival of messages at other nodes with the rest of the network), **connect**, and **disconnect** actions.

At the end, only single actions with certain labels are permitted to occur: multi-actions produced by putting $\mathrm{T}(\mathrm{P})$ and G in parallel are blocked, and so are leftover **cast**, $\overline{\mathbf{cast}}$, ¬**uni**, $\overline{\neg\mathbf{uni}}$, **del**, $\overline{\mathbf{del}}$, **receive**, and $\overline{\mathbf{receive}}$ actions.

Rule T15: Parallel nodes

$$\begin{aligned}
\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) &= \rho_R \nabla_V \Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}} \Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \quad \text{T15}\\
&\quad \text{where } R = \{\mathbf{a} \to \mathbf{arrive}, \mathbf{c} \to \mathbf{connect}, \mathbf{d} \to \mathbf{disconnect}, \mathbf{s} \to \mathbf{starcast}\}\\
&\quad \text{where } V = \{\mathbf{a}, \mathbf{c}, \mathbf{d}, \mathbf{deliver}, \mathbf{s}, \mathbf{t}\}\\
&\quad \text{where } C = \{\mathbf{starcast}|\mathbf{arrive} \to \mathbf{s}, \mathbf{connect}|\mathbf{connect} \to \mathbf{c}, \mathbf{disconnect}|\mathbf{disconnect} \to \mathbf{d}\}
\end{aligned}$$

This rule produces a sequence of mCRL2 operators that restricts the behavior of $\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})$ in such a way that any remaining action that can occur has a possible counterpart in AWN. In particular, the sequence of mCRL2 operators enforces that:

- One process does **starcast** action *and* the other process does a matching **arrive** action;
- Both processes do the same **connect**, **disconnect**, or **arrive** action;
- One of the processes does a **deliver** or **t** action and the other process does nothing.

Rule T16: Network

$$\mathrm{T}([\,\mathrm{M}\,]) = \nabla_V \rho_{\{\mathbf{starcast}\to\mathbf{t}\}} \Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \quad \text{T16}$$
$$\text{where } V = \{\mathbf{t}, \mathbf{newpkt}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}$$
$$\text{where } C = \{\overline{\mathbf{newpkt}}|\mathbf{arrive} \to \mathbf{newpkt}\}$$
$$\text{where } \mathrm{H} = \textstyle\sum_{\mathtt{ip}:\mathrm{T}(\mathrm{IP}),\mathtt{data}:\mathrm{T}(\mathrm{DATA}),\mathtt{dest}:\mathrm{T}(\mathrm{IP})} \overline{\mathbf{newpkt}}(\{\mathtt{ip}\},\{\mathtt{ip}\},\mathtt{newpkt}(\mathtt{data},\mathtt{dest})).\mathrm{H}$$

At the highest level of the AWN specification, it is possible to inject messages into the network via **newpkt** actions. In mCRL2, this behavior is enabled by translation rule T16. A **newpkt** action is generated from a remaining **arrive** action (which means that all nodes in the network are taking this action in parallel) that has exactly one node both as intended and as actual destination. In addition, the message parameter of the action is forced to have a specific `newpkt` format.

The rule also renames remnant **starcast** actions to **t** and finally blocks multi-actions such as **connect**$|\overline{\mathbf{newpkt}}$ as well as leftover **arrive** and $\overline{\mathbf{newpkt}}$ actions.

3.3 Totalness

This section shows that the translation function T is total.

Lemma 3.1 The translation function T as defined via the rules in Tables 2.8 and 2.9 is *total*; that is, $\mathrm{T}(\mathrm{P})$ is defined for all valid AWN expressions P.

Proof. The proof is trivial, and therefore given informally.

First, there must exist a matching translation rule for each rule of the grammar of AWN. This is indeed the case.

Second, there must exist a matching translation rule for each input at recursive applications of T. Considering that recursive applications only receive process expressions from the original AWN process expression as input, such as p or q, these process expressions must match the grammar of AWN, and therefore a matching translation rule must exist.

Third, there should be no infinite recursion. Since the input at recursive applications of T is always a strictly smaller expression than the input of the enveloping translation rule, recursion is always finite.

Finally, $\mathrm{T}_\xi(e)$ must be defined for all valid data expressions e. Combining property (i) of $\mathrm{T}_\xi(e)$ with the fact that $\forall e \in M_{\mathrm{sort}(e)} \,.\, \exists t \,.\, [\![t]\!] = e$ (see Section 2.2) where M_D is the set of all semantic values of a sort D, it follows that T_ξ is a total function. ■

3.4 Translation relation

Proving the correctness of the translation function involves showing the existence of a strong bisimulation between an AWN process and its translated counterpart in mCRL2. Because a strong bisimulation is a relation, it is useful to express the translation function as relation:

$$\widetilde{\mathrm{T}} \stackrel{\text{def}}{=} \{\ (\mathrm{P}, \mathrm{T}(\mathrm{P})) \mid \mathrm{P} \text{ is an AWN parallel process, node, or (partial) network expression } \} \tag{2.1}$$

Ultimately, however, the correctness proof will be a statement about the following relation:

$$\widetilde{\mathrm{T}}_\tau \stackrel{\text{def}}{=} \left\{\ ([\![\,\mathrm{M}\,]\!], \tau_{\{\mathbf{t}\}}\mathrm{T}([\![\,\mathrm{M}\,]\!])) \;\middle|\; [\![\,\mathrm{M}\,]\!] \text{ is an AWN network expression } \right\} \tag{2.2}$$

According to Lemma 3.1, both relations are defined for all valid AWN expressions of a matching type.

3.5 Action relation

The translation function T changes AWN actions to mCRL2 actions in a manner that is not always straightforward. To formally prove the correctness of T, the precise relation between AWN actions and mCRL2 actions must be made explicit first. The relation is called $\mathscr{A}$ and its definition can be found in Table 2.10.

$$\begin{aligned}
\mathscr{A} \stackrel{\text{def}}{=} \{\\
&(\{\ \tau\ \}, \{\ \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m))\ \}), && (2.3)\\
&(\{\ \mathbf{broadcast}(\xi(ms))\ \}, \{\ \mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathrm{D}}, [\![\mathrm{T}_\xi(ms)]\!])\ \mid\ \hat{\mathrm{D}} \in \mathrm{T}(\mathrm{Set}(\mathrm{IP}))\ \}), && (2.4)\\
&(\{\ \mathbf{groupcast}(\xi(dests), \xi(ms))\ \}, \{\ \mathbf{cast}([\![\mathrm{T}_\xi(dests)]\!], \hat{\mathrm{D}}, [\![\mathrm{T}_\xi(ms)]\!])\ \mid\ \hat{\mathrm{D}} \in \mathrm{T}(\mathrm{Set}(\mathrm{IP}))\ \}), && (2.5)\\
&(\{\ \mathbf{unicast}(\xi(dest), \xi(ms))\ \}, \{\ \mathbf{cast}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(ms)]\!])\ \}), && (2.6)\\
&(\{\ \neg\mathbf{unicast}(\xi(dest), \xi(ms))\ \}, \{\ \neg\mathbf{uni}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!])\ \}), && (2.7)\\
&(\{\ \mathbf{send}(\xi(ms))\ \}, \{\ \mathbf{send}([\![\emptyset]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!])\ \}), && (2.8)\\
&(\{\ \mathbf{deliver}(\xi(data))\ \}, \{\ \mathbf{del}(\hat{\imath}, [\![\mathrm{T}_\xi(data)]\!])\ \mid\ \hat{\imath} \in \mathrm{T}(\mathrm{IP})\ \}), && (2.9)\\
&(\{\ \mathbf{receive}(m)\ \}, \{\ \mathbf{receive}(\hat{\mathrm{D}}, \hat{\mathrm{D}}', \mathrm{U}(m))\ \mid\ \hat{\mathrm{D}}, \hat{\mathrm{D}}' \in \mathrm{T}(\mathrm{Set}(\mathrm{IP}))\ \}), && (2.10)\\
&(\{\ R : \mathbf{*cast}(m)\ \}, \{\ \mathbf{starcast}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m))\ \}), && (2.11)\\
&(\{\ ip : \mathbf{deliver}(d)\ \}, \{\ \mathbf{deliver}(\mathrm{U}(ip), \mathrm{U}(d))\ \}), && (2.12)\\
&(\{\ H\neg K : \mathbf{arrive}(m)\ \}, \left\{\ \mathbf{arrive}(\hat{\mathrm{D}}, \hat{\mathrm{D}}', \mathrm{U}(m)) \;\middle|\; \begin{array}{c}\hat{\mathrm{D}}, \hat{\mathrm{D}}' \in \mathrm{T}(\mathrm{Set}(\mathrm{IP}))\\ \hat{\mathrm{D}}' \subseteq \hat{\mathrm{D}}\\ H \subseteq \hat{\mathrm{D}}'\\ K \cap \hat{\mathrm{D}}' = \emptyset\end{array} \right\}), && (2.13)\\
&(\{\ \mathbf{connect}(ip', ip'')\ \}, \{\ \mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))\ \}), && (2.14)\\
&(\{\ \mathbf{disconnect}(ip', ip'')\ \}, \{\ \mathbf{disconnect}(\mathrm{U}(ip'), \mathrm{U}(ip''))\ \}), && (2.15)\\
&(\{\ ip : \mathbf{newpkt}(d, dip)\ \}, \{\ \mathbf{newpkt}(\{\mathrm{U}(ip)\}, \{\mathrm{U}(ip)\}, \mathtt{newpkt}(\mathrm{U}(d), \mathrm{U}(dip)))\ \}) && (2.16)\\
\mid\ & m, \xi(ms) \in \mathrm{MSG};\ ip, ip', ip'', dip, dest \in \mathrm{IP};\ d, \xi(data) \in \mathrm{DATA};\ dests, R, D, H, K \in \mathrm{Set}(\mathrm{IP});\ R \subseteq D\ \}
\end{aligned}$$

Table 2.10: Action relation $\mathscr{A}$

4 Correctness proof

This section proves that the translation of any AWN expression $[\![\mathrm{M}]\!]$ is strongly bisimilar to $\mathrm{T}([\![\mathrm{M}]\!])$ up to data congruence, modulo renaming. This is done by proving Theorem 4.11, which shows that $\widetilde{\mathrm{T}}_\tau$ is a satisfactory strong bisimulation (once again, up to data congruence and modulo renaming). An overview of how the proof of the Theorem 4.11 is established is given in Figure 2.1. It shows the lemmas and theorems used by the correctness proof and how they relate: arrows connect (groups of) lemmas and theorems, pointing towards the lemma or theorem for which they are used.

First, it must be established that data congruence is indeed a strong bisimulation in mCRL2 (see Theorem 4.1). From there, it is a short step to show that a strong bisimulation up to data congruence indeed implies strong bisimulation between AWN expressions and their translations (see Theorem 4.2) – this makes it possible to substitute mCRL2 data expressions by semantically equivalent ones in later proofs.

The next part of the proof establishes Lemmas 4.3 to 4.6:

- Lemma 4.3 establishes a useful shorthand equivalence between the meaning of a closed expression in AWN and the meaning of syntactic expressions that denote its corresponding syntactic value in mCRL2.
- Lemma 4.4 states that AWN expressions can be translated to mCRL2 expressions without losing their meaning: the semantic values of the expressions in AWN and mCRL2 must correspond with one another. This is clearly a desirable property of the $\mathrm{T}_\xi(exp)$ function, and it is proven correct with the help of Lemma 4.3.
- Lemma 4.5 concerns substitutions: later parts of the proof require that a substitution σ occurring in an expression such as $\mathrm{T}_{\xi[\sigma]}(exp)$ can be moved *outside* of the T function.
- Similar to Lemma 4.5, Lemma 4.6 enables an expression like $\mathrm{T}_V(\xi[\sigma],\mathrm{P})$ to be rewritten to $\mathrm{T}_V(\xi,\mathrm{P})[\mathrm{T}(\sigma)]$ (note that this is just a pseudo-expression to sketch the principle of the lemma).

The work so far has been generic preparatory work. The proof of Theorem 4.11 starts in earnest with Lemmas 4.7, 4.8, and 4.9. Lemma 4.7 proves by structural induction that actions of an AWN process P can be mimicked by its mCRL2 counterpart $\mathrm{T}(\mathrm{P})$. Note that the notion of a *strong $\mathscr{A}$-warped simulation* is used here: an action by the AWN process must be mimicked by the mCRL2 process with a *set of actions*, as defined by the action relation $\mathscr{A}$.

The reverse of Lemma 4.7 has been divided into two theorems because the translation function for the sequential processes of AWN has a different signature than the translation function for higher-level AWN processes. Lemma 4.8 proves that mCRL2 process $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})$ can also be mimicked by AWN process ξ,p. To this end, index parameter V is chosen as $\mathrm{DOM}(\xi)$ because AWN inference rules can only be applied if their data expressions are bound. Lemma 4.9 builds on Lemma 4.8, proving that it generally holds that an mCRL2 process $\mathrm{T}(\mathrm{P})$ can mimic the behavior of AWN process P.

Lemmas 4.7 and 4.9 combined prove that translation relation $\widetilde{\mathrm{T}}$ is a *strong $\mathscr{A}$-warped bisimulation*. This only leaves the proof of Theorem 4.11, where it is shown that $\mathscr{A}$ at the network level can be replaced by a bijective renaming function $\mathfrak{f}$. The conclusion that any AWN expression $[\![\mathrm{M}]\!]$ is strongly bisimilar to $\mathrm{T}([\![\mathrm{M}]\!])$ up to data congruence and modulo renaming immediately follows.

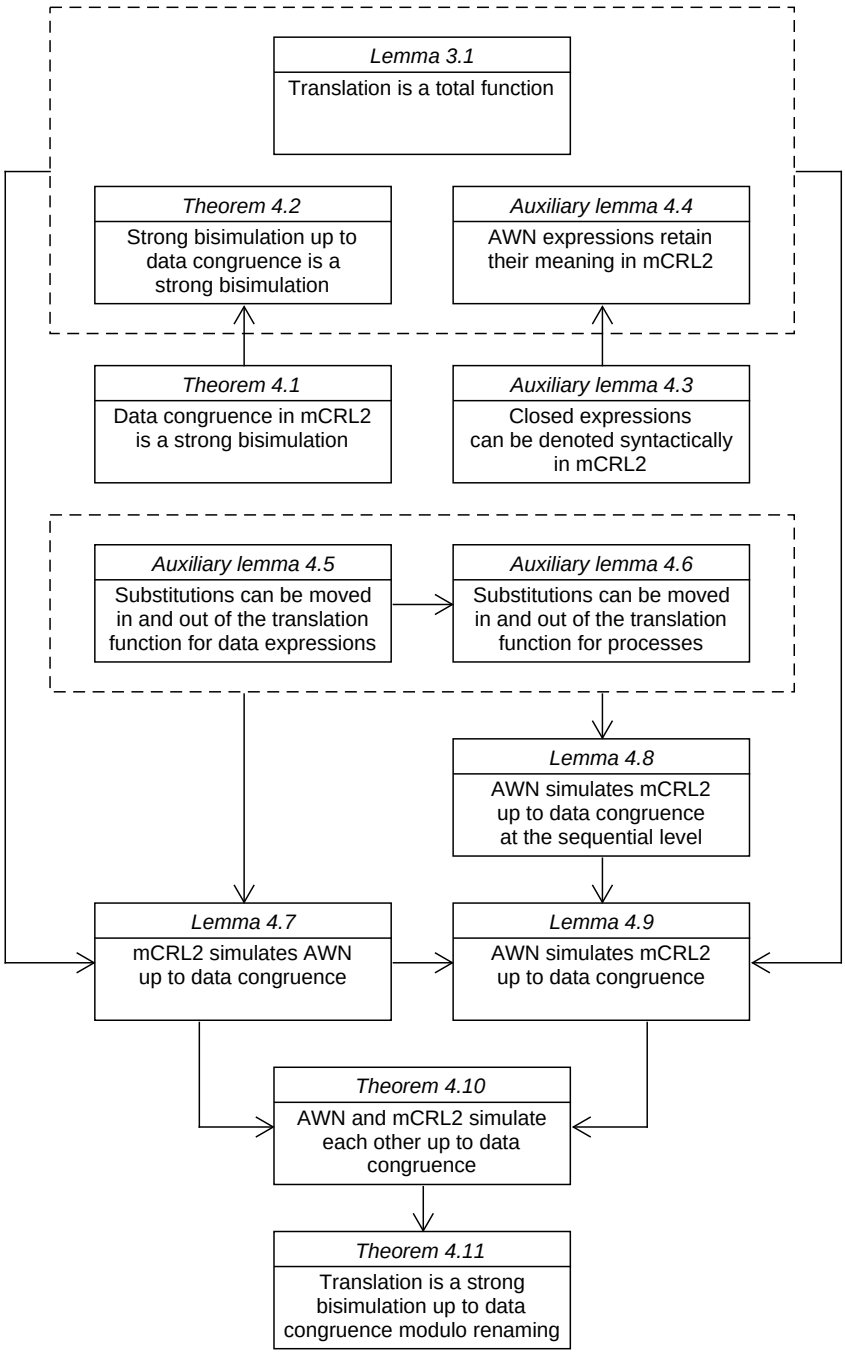


Figure 2.1: Overview of the correctness proof.

4.1 Representative derivations

The proofs for Theorems 4.8 and 4.9 require the analysis of all possible behavior of certain mCRL2 process expressions, namely those process expressions that can be produced by the function T. To obtain all possible behavior that corresponds with an mCRL2 process expression p, all possible derivation trees (or simply 'derivations') based on the mCRL2 inference rules that result in p must be constructed. Clearly, this is a lot of effort.

In order to reduce the number of derivations that must be considered as well as the number of derivations that should be listed in order to convince the reader, the concept of a *representative derivation* is introduced here. First, consider the definition of *acyclic derivations*:

Definition 4.1 An mCRL2 derivation is called *cyclic* if it produces two conclusions that are semantically equivalent and if one of those conclusions is produced *before* the application of an mCRL2 inference rule and the other of those conclusions is produced *after* the same inference rule application. All other mCRL2 derivations are called *acyclic*.

It should be obvious that every mCRL2 process expression has an acyclic derivation, and that all cyclic derivations can be rewritten to some acyclic derivation.

Definition 4.2 A *representative derivation* is an acyclic derivation that describes all acyclic derivations that

- apply the same mCRL2 inference rules as the representative derivation (in the same order and the same number of times), and
- reach a final conclusion that is semantically equivalent to the final conclusion of the representative derivation.

The number of representative derivations in mCRL2 for a process expression is limited because all inference rules of mCRL2 extend their input premisse(s) with some syntax that would not appear in the correct position if the inference rules would be applied in a different order. This even includes the inference rules RECURSION 1 or RECURSION 2 – rules that at first glance seem to remove syntax rather than add it – because particular process definitions are required in order to rewrite process expressions to a process call.

As an illustration of how the number of representative derivations is limited, consider the task of determining the behavior of the expression

$$\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{broadcast}(ms).\mathrm{p})$$

Following the definition of T_{ξ}, this behavior is the same as the behavior of

$$\sum_{\mathtt{D}:\mathrm{T}(\mathrm{Set(IP)})} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathrm{T}_{\xi}(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})$$

The only inference rules of mCRL2 that can produce this expression are SUM 1 and SUM 2, because how else could the $\sum$-operator be where it is? In either case, the new expression is

$$\big(\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathrm{T}_{\xi}(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})\big)[\mathtt{D} := t_{\hat{\mathtt{D}}}]$$

which can be rewritten to

$$\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathrm{T}_{\xi}(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})$$

This expression can only result from the SEQ 1 or SEQ 2 inference rules, neither of which produces ✓ as a potential next state – this means that SUM 1 could not have been used to get here!

Finally, the question becomes whether both SEQ 1 and SEQ 2 are possible or only one of them. The answer is 'no': SEQ 2 would require the **cast** action to solutarily lead to a new state (that is not ✓) and this is not possible. SEQ 1, on the other hand, gives an expression that adheres to the form of AXIOM, the only available mCRL2 axiom. Consequently, there is only one representative derivation for the original expression:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathrm{T}_{\xi}(ms)) \xrightarrow{[\![\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathrm{T}_{\xi}(ms))]\!]} \checkmark}\ \text{AXIOM}}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathrm{T}_{\xi}(ms)) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_{\xi}(ms)]\!])} \checkmark}\ \text{Definition 2.1}}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathrm{T}_{\xi}(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_{\xi}(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{SEQ 1}}{\big(\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathrm{T}_{\xi}(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})\big)[\mathtt{D} := t_{\hat{\mathtt{D}}}] \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_{\xi}(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{Substitution}}{\sum_{\mathtt{D}:\mathrm{T}(\mathrm{Set}(\mathrm{IP}))} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathrm{T}_{\xi}(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_{\xi}(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{SUM 2}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{broadcast}(ms).\mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_{\xi}(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{T1}$$

It should be obvious that providing all representative derivations for a process expression is sufficient to obtain all possible behavior of that process expression. This approach is used several times in this document, often indicated with the phrase 'a representative derivation without alternatives':

Definition 4.3 A *representative derivation without alternatives* is a representative derivation A in mCRL2 such that there does *not* exist another representative derivation B that

- reaches a final conclusion that is semantically equivalent to the final conclusion of A, and
- is *not* described by A.

4.2 Data congruence

In various parts of the correctness proof a specific closed expression in mCRL2 $t_{\mathrm{U}(\xi(exp))}$ has been obtained, and in order to progress it must be shown that $t_{\mathrm{U}(\xi(exp))} = \mathrm{T}_\xi(exp)$. However, this is not necessarily the case. Suppose, for example, that $exp = 1+1$. Then potentially $t_{\mathrm{U}(\xi(exp))} = 2$ and $\mathrm{T}_\xi(exp) = 1+1$ (depending on the evaluations of $t_{\mathrm{U}(\xi(1+1))}$ and $\mathrm{T}(1+1)$), and the (syntactic) expressions 2 and $1+1$ are clearly not the same!

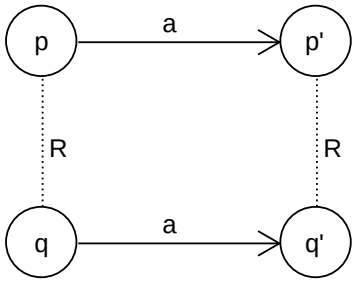
In order for the proof to work, it must be established that such differences can be safely ignored. This is possible if making an exclusively syntactic change to the data expression of a process does not affect the behavior of that process; that is, if a process that results from making such an exclusively syntactic change is *strongly bisimilar* to the original process. The mCRL2 authors do not provide a proof of such a property for their language, and therefore this section has been dedicated to it.

First, define the notion of a *strong bisimulation*:

Definition 4.4 Let $\mathrm{LTS}_1 = (S_1, \mathrm{Act}, \rightarrow_1)$ and $\mathrm{LTS}_2 = (S_2, \mathrm{Act}, \rightarrow_2)$ be labeled transition systems where $\rightarrow_i \subseteq S_i \times \mathrm{Act} \times S_i$. Then $R \subseteq S_1 \times S_2$ is called a *strong simulation of* LTS_1 *by* LTS_2 if and only if

$$\forall (\mathrm{p},\mathrm{q}) \in R, a \in \mathrm{Act}, \mathrm{p}' \in S_1 \,.\, \mathrm{p} \xrightarrow{a}_1 \mathrm{p}' \Rightarrow \exists \mathrm{q}' \in S_2 \,.\, \mathrm{q} \xrightarrow{a}_2 \mathrm{q}' \wedge (\mathrm{p}',\mathrm{q}') \in R$$

Strong simulation is a commonly used concept, but for the correctness proof its definition is extended several times. In preparation of this, the sketch below illustrates strong simulation in its current form:



For the definition to apply, in all cases where an action a from p to p' is possible and $(\mathrm{p},\mathrm{q}) \in R$, there must also exist a q' that q can reach via a such that p' and q' are also related by R.

Definition 4.5 Let $\mathrm{LTS}_1 = (S_1, \mathrm{Act}, \rightarrow_1)$ and $\mathrm{LTS}_2 = (S_2, \mathrm{Act}, \rightarrow_2)$ be labeled transition systems where $\rightarrow_i \subseteq S_i \times \mathrm{Act} \times S_i$. Then $R_1 \subseteq S_1 \times S_2$ is called a *strong bisimulation between* LTS_1 *and* LTS_2 if and only if

- R is a strong simulation of LTS_1 by LTS_2 and
- $R^\smile$ is a strong simulation of LTS_2 by LTS_1.

The meaning of *data congruence* must also be formally specified:

Definition 4.6 Let p and q be two mCRL2 processes. Then p and q are *equivalent up to data congruence* if:

- p and q are only syntactically different in their data expressions, and
- when doing a pairwise comparison, the data expressions of p and q are semantically equivalent.

In this context, 'data expressions' refers to guard conditions, action arguments, summation variables, and the arguments of process recursions.

The equivalence of p and q up to data congruence is denoted as $p \equiv q$.

The following theorem is proposed:

Theorem 4.1 Data congruence in mCRL2 is a *strong bisimulation*.

Proof. A partial proof can be found in Appendix A (because the proof requires a case distinction with many similar cases, only some of these cases are treated). ■

Now consider the following definitions:

Definition 4.7 Let p and q be two processes, let A be a set of actions and let z be an action. Then

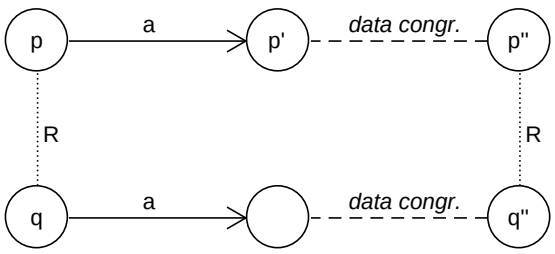
$$p \xrightarrow{z} \equiv q \Rightarrow \exists q' \,.\, p \xrightarrow{z} q' \wedge q' \equiv q$$
$$p \xrightarrow{A} \equiv q \Rightarrow \forall a \in A \,.\, p \xrightarrow{a} \equiv q$$

Note that $p \xrightarrow{z} \equiv q \Rightarrow p \xrightarrow{z} q$ and $p \xrightarrow{A} \equiv q \Rightarrow p \xrightarrow{A} q$.

Definition 4.8 Let $\mathrm{LTS}_1 = (S_1, \mathrm{Act}, \rightarrow_1)$ and $\mathrm{LTS}_2 = (S_2, \mathrm{Act}, \rightarrow_2)$ be labeled transition systems where $\rightarrow_i \subseteq S_i \times \mathrm{Act} \times S_i$. Then $R \subseteq S_1 \times S_2$ is called a *strong simulation up to* $\equiv$ *of* LTS_1 *by* LTS_2 if and only if

$$\forall (p,q) \in R, a \in \mathrm{Act}, p' \in S_1 \,.\, p \xrightarrow{a}_1 p' \Rightarrow$$
$$\exists p'' \in S_1, q'' \in S_2 \,.\, p \xrightarrow{a}_1 \equiv p'' \wedge p'' \equiv p' \wedge q \xrightarrow{a}_2 \equiv q'' \wedge (p'', q'') \in R$$

The sketch below illustrates this definition:



For the definition to apply, in all cases where an action a from p to p′ is possible and $(\mathrm{p},\mathrm{q}) \in R$, the other states, actions, and relations that are depicted must also exist:

- p must be able to reach a state that is data congruent with some state $\mathrm{p}'' \equiv \mathrm{p}'$ via the action a;
- q must be able to reach a state that is data congruent with some state q″ via the action a; and
- p″ and q″ must be related by R.

Definition 4.9 Let $\mathrm{LTS}_1 = (S_1, \mathrm{Act}, \rightarrow_1)$ and $\mathrm{LTS}_2 = (S_2, \mathrm{Act}, \rightarrow_2)$ be labeled transition systems where $\rightarrow_i \subseteq S_i \times \mathrm{Act} \times S_i$. Then $R_1 \subseteq S_1 \times S_2$ is called a *strong bisimulation up to* $\equiv$ *between* LTS_1 *and* LTS_2 if and only if

- R is a strong simulation up to $\equiv$ of LTS_1 by LTS_2 and
- $R^{\smile}$ is a strong simulation up to $\equiv$ of LTS_2 by LTS_1.

This leads to the following theorem:

Theorem 4.2 Let S_1 and S_2 be sets of mCRL2 process expressions and let $R \subseteq S_1 \times S_2$ be a strong bisimulation up to $\equiv$. Then S_1 and S_2 are strongly bisimilar.

Proof. The theorem follows immediately from Theorem 4.1 and a lemma from Milner [26]. ■

For convenience the definition of $\equiv$ is extended by the identity relation for AWN processes:

Definition 4.10 Let p and q be two processes. Then p and q are *equivalent up to* $\equiv$ if:

- p and q are two mCRL2 processes that are equivalent up to $\equiv$; or
- p and q are two AWN processes such that $\mathrm{p} = \mathrm{q}$.

The equivalence of p and q up to $\equiv$ is (still) denoted as $\mathrm{p} \equiv \mathrm{q}$.

4.3 Auxiliary lemmas

This section gives several auxiliary lemmas and their proofs.

Lemma 4.3 If $\xi(exp)$ is defined, then $[\![\mathrm{T}_\xi(exp)]\!] = [\![t_{\mathrm{U}(\xi(exp))}]\!]$.

Proof. Because $\xi(exp)$ is defined, Property (ii) of the $\mathrm{T}_\xi(exp)$ function gives that $[\![\mathrm{T}_\xi(exp)]\!] = \mathrm{U}(\xi(exp))$. Furthermore, it is assumed in mCRL2 that $[\![t_e]\!] = e$ for all semantic values e (see the definition of t in Section 2.2) so that $[\![t_{\mathrm{U}(\xi(exp))}]\!] = \mathrm{U}(\xi(exp)) = [\![\mathrm{T}_\xi(exp)]\!]$. ■

Lemma 4.4 If e is a semantic AWN value and $\xi(exp)$ is defined, then

$$\xi(exp) = e \Leftrightarrow [\![\mathrm{T}_\xi(exp)]\!] = \mathrm{U}(e)$$

Proof. Start with the observation that

$$\mathrm{U}(e) \overset{\text{Definition of } t}{=} [\![t_{\mathrm{U}(e)}]\!] \qquad \text{and} \qquad [\![\mathrm{T}_\xi(exp)]\!] \overset{\text{Lemma 4.3}}{=} [\![t_{\mathrm{U}(\xi(exp))}]\!] \tag{2.17}$$

As a consequence,

$$\xi(exp) = e \overset{\text{U is a bijection}}{\Longleftrightarrow} \mathrm{U}(\xi(exp)) = \mathrm{U}(e) \Leftrightarrow [\![t_{\mathrm{U}(\xi(exp))}]\!] = [\![t_{\mathrm{U}(e)}]\!] \overset{2.17}{\Longleftrightarrow} [\![\mathrm{T}_\xi(exp)]\!] = \mathrm{U}(e)$$

■

Lemma 4.5 If e is a semantic AWN value and $\mathtt{v} \notin \mathrm{DOM}(\xi)$, then

$$\mathrm{T}_\xi(exp)\left[\mathrm{T}(\mathtt{v}) := t_{\mathrm{U}(e)}\right] = \mathrm{T}_{\xi[\mathtt{v}:=e]}(exp)$$

Proof.

$$\begin{aligned}
\mathrm{T}_\xi(exp)\left[\mathrm{T}(\mathtt{v}) := t_{\mathrm{U}(e)}\right] &\overset{\text{Definition of } \mathrm{T}_\xi}{=} \mathrm{T}(exp)\left[\mathrm{T}(\mathtt{x}) := t_{\mathrm{U}(y)} \mid \mathtt{x} := y \in \xi\right]\left[\mathrm{T}(\mathtt{v}) := t_{\mathrm{U}(e)}\right] \\
&\overset{\mathtt{v} \notin \mathrm{DOM}(\xi)}{=} \mathrm{T}(exp)\left[\mathrm{T}(\mathtt{x}) := t_{\mathrm{U}(y)} \mid \mathtt{x} := y \in \xi[\mathtt{v} := e]\right] \\
&\overset{\text{Definition of } \mathrm{T}_\xi}{=} \mathrm{T}_{\xi[\mathtt{v}:=e]}(exp)
\end{aligned}$$

Note that $\mathtt{v} \notin \mathrm{DOM}(\xi)$ is needed because otherwise equating two ordered substitutions with a single substitution is not necessarily valid. ■

Lemma 4.6 If e is a semantic AWN value and $\mathtt{v}$ an AWN variable, then

$$\mathtt{v} \in V \wedge \mathtt{v} \notin \mathrm{DOM}(\xi) \Rightarrow \mathrm{T}_V(\xi, \mathrm{p})\left[\mathrm{T}(\mathtt{v}) := t_{\mathrm{U}(e)}\right] = \mathrm{T}_V(\xi[\mathtt{v} := e], \mathrm{p})$$

Proof. See Appendix B. ■

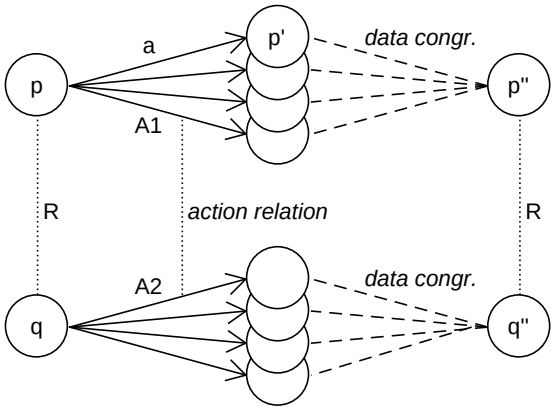
4.4 Proof for strong warped bisimulation

Because mCRL2 mimics AWN actions with sets of actions, the regular notion of strong bisimulation is not suitable for proving Lemmas 4.7, 4.8, and 4.9 in a compositional manner. Instead, a weaker version of bisimulation is introduced, to return to bisimulation up to $\equiv$ later:

Definition 4.11 Let $\text{LTS}_1 = (S_1, \text{Act}_1, \rightarrow_1)$ and $\text{LTS}_2 = (S_2, \text{Act}_2, \rightarrow_2)$ be labeled transition systems where $\rightarrow_i \subseteq S_i \times \text{Act}_i \times S_i$ and let $\mathscr{A} \subseteq \mathscr{P}(\text{Act}_1) \times \mathscr{P}(\text{Act}_2)$. Then $R \subseteq S_1 \times S_2$ is a *strong $\mathscr{A}$-warped simulation up to $\equiv$ of* LTS_1 *by* LTS_2 if and only if

$$\forall (\text{p},\text{q}) \in R, a \in \text{Act}_1, \text{p}' \in S_1 \,.\, \text{p} \xrightarrow{a}_1 \text{p}' \Rightarrow$$
$$\exists (A_1, A_2) \in \mathscr{A}, \text{p}'' \in S_1, \text{q}'' \in S_2 \,.\, a \in A_1 \wedge \text{p} \xrightarrow{A_1}_1 \equiv \text{p}'' \wedge \text{p}'' \equiv \text{p}' \wedge \text{q} \xrightarrow{A_2}_2 \equiv \text{q}'' \wedge (\text{p}'', \text{q}'') \in R$$

A simple sketch is given to illustrate this definition:



For the definition to apply, in all cases where an action a from p to p' is possible and $(\text{p},\text{q}) \in R$, the other states, actions, and relations that are depicted must also exist:

- p must be able to reach a state that is data congruent with some state $\text{p}'' \equiv \text{p}'$ via all actions in A_1, a set of actions containing the action a;
- q must be able to reach a state that is data congruent with some state q'' via all actions in A_2;
- A_1 and A_2 must be related by the action relation $\mathscr{A}$; and
- p'' and q'' must be related by R.

Definition 4.12 Let $\text{LTS}_1 = (S_1, \text{Act}_1, \rightarrow_1)$ and $\text{LTS}_2 = (S_2, \text{Act}_2, \rightarrow_2)$ be labeled transition systems where $\rightarrow_i \subseteq S_i \times \text{Act}_i \times S_i$ and let $\mathscr{A} \subseteq \mathscr{P}(\text{Act}_1) \times \mathscr{P}(\text{Act}_2)$. Then $R \subseteq S_1 \times S_2$ is called a *strong $\mathscr{A}$-warped bisimulation up to $\equiv$ between* LTS_1 *and* LTS_2 if and only if

- R is a strong $\mathscr{A}$-warped simulation up to $\equiv$ of LTS_1 by LTS_2 and
- $R^\smile$ is a strong $\mathscr{A}^\smile$-warped simulation up to $\equiv$ of LTS_2 by LTS_1.

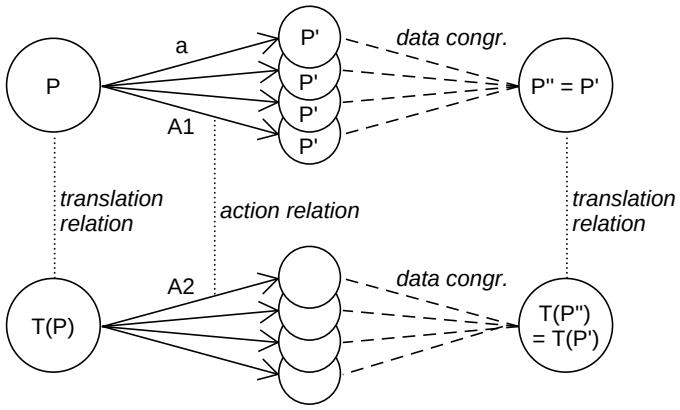
It is claimed that mCRL2 expressions $\mathrm{T}(\mathrm{P})$ mimic AWN expressions P in accordance with Definition 4.11:

Lemma 4.7 Take translation relation $\widetilde{\mathrm{T}}$ from Equation 2.1 and action relation $\mathscr{A}$ from Table 2.10. Then $\widetilde{\mathrm{T}}$ is a strong $\mathscr{A}$-warped simulation up to $\equiv$ of AWN expressions P by mCRL2 expressions $\mathrm{T}(\mathrm{P})$ for all AWN expressions P.

Proof. Using Lemma 3.1, for Definition 4.11 to apply it is sufficient to prove that

$$\forall \mathrm{P}, \mathrm{P}', a \,.\, \mathrm{P} \xrightarrow{a} \mathrm{P}' \Rightarrow \exists (A_1, A_2) \in \mathscr{A} \,.\, a \in A_1 \wedge \mathrm{P} \xrightarrow{A_1} \mathrm{P}' \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_2} \equiv \mathrm{T}(\mathrm{P}') \tag{2.18}$$

The translation relation is defined for every AWN expression P, and it is therefore valid to choose $\mathrm{P}'' = \mathrm{P}'$. This can be depicted graphically as



The proof of Equation 2.18 is by structural induction over the inference rules of AWN.
Induction hypothesis: For all premises $\mathrm{P} \xrightarrow{a} \mathrm{P}'$ of an AWN inference rule, it holds that

$$\exists (A_1, A_2) \in \mathscr{A} \,.\, a \in A_1 \wedge \mathrm{P} \xrightarrow{A_1} \mathrm{P}' \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_2} \equiv \mathrm{T}(\mathrm{P}')$$

Base cases: Show for each axiomatic inference rule of AWN with conclusion $\mathrm{P} \xrightarrow{a} \mathrm{P}'$ that there is some $(A_1, A_2) \in \mathscr{A} \,.\, a \in A_1 \wedge \mathrm{P} \xrightarrow{A_1} \mathrm{P}'$ such that $\mathrm{T}(\mathrm{P}) \xrightarrow{a'} \equiv \mathrm{T}(\mathrm{P}')$ can be derived for all $a' \in A_2$.

■ **Example 2.1** AWN defines the inference rule

$$\frac{}{\xi, \mathbf{broadcast}(ms).\mathrm{p} \xrightarrow{\mathbf{broadcast}(\xi(ms))} \xi, \mathrm{p}} \text{ BROADCAST (T1)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\mathbf{broadcast}(\xi(ms)) \in A_1 \wedge \xi, \mathbf{broadcast}(ms).\mathrm{p} \xrightarrow{A_1} \xi, \mathrm{p}$, namely Pair 2.4 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 such that $\mathrm{T}(\xi, \mathbf{broadcast}(ms).\mathrm{p}) \xrightarrow{a} \equiv \mathrm{T}(\xi, \mathrm{p})$ for all $a \in A_2$ where

$$A_2 = \left\{ \mathbf{cast}(\llbracket \mathscr{U}_{\mathrm{IP}} \rrbracket, \hat{\mathrm{D}}, \llbracket \mathrm{T}_\xi(ms) \rrbracket) \;\middle|\; \hat{\mathrm{D}} \in \mathrm{T}(\mathrm{Set}(\mathrm{IP})) \right\}$$

In mCRL2, the following derivation can be made for all $\hat{\mathtt{D}} \in \mathtt{T}(\mathrm{Set(IP)})$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathtt{T}_\xi(ms)) \xrightarrow{[\![\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathtt{T}_\xi(ms))]\!]} \checkmark}\ \text{Axiom}}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathtt{T}_\xi(ms)) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \checkmark}\ \text{Definition 2.1}}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathtt{T}_\xi(ms)).\mathtt{T}_{\mathrm{DOM}(\xi)}(\xi, \mathtt{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathtt{T}_{\mathrm{DOM}(\xi)}(\xi, \mathtt{p})}\ \text{Seq 1}}{(\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathtt{T}_\xi(ms)).\mathtt{T}_{\mathrm{DOM}(\xi)}(\xi, \mathtt{p}))[\mathtt{D} := t_{\hat{\mathtt{D}}}] \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathtt{T}_{\mathrm{DOM}(\xi)}(\xi, \mathtt{p})}\ \text{Substitution}}{\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set(IP)})} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathtt{T}_\xi(ms)).\mathtt{T}_{\mathrm{DOM}(\xi)}(\xi, \mathtt{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathtt{T}_{\mathrm{DOM}(\xi)}(\xi, \mathtt{p})}\ \text{Sum 2}}{\mathtt{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{broadcast}(ms).\mathtt{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathtt{T}_{\mathrm{DOM}(\xi)}(\xi, \mathtt{p})}\ \text{T1}}{\mathtt{T}(\xi, \mathbf{broadcast}(ms).\mathtt{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathtt{T}(\xi, \mathtt{p})}\ \text{T12}$$

for $\mathtt{D} \notin \mathtt{T}(V)$. In conclusion, the induction hypothesis holds for this base case.

■

Induction step: Given only its side conditions and the induction hypothesis, show for each inference rule of AWN with conclusion $\mathrm{P} \xrightarrow{a} \mathrm{P}'$ that there is some $(A_1, A_2) \in \mathscr{A}$. $a \in A_1 \wedge \mathrm{P} \xrightarrow{A_1} \mathrm{P}'$ such that $\mathrm{T}(\mathrm{P}) \xrightarrow{a'} \equiv \mathrm{T}(\mathrm{P}')$ can be derived for all $a' \in A_2$.

■ **Example 2.2** AWN defines the inference rule

$$\dfrac{\mathrm{P} \xrightarrow{\mathbf{broadcast}(m)} \mathrm{P}'}{ip : \mathrm{P} : R \xrightarrow{R:\mathbf{*cast}(m)} ip : \mathrm{P}' : R}\ \text{Broadcast (T3)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $R : \mathbf{*cast}(m) \in A_1 \wedge ip : \mathrm{P} : R \xrightarrow{A_1} ip : \mathrm{P}' : R$, namely Pair 2.11 in Table 2.10 (note that it is possible to choose $D = \mathscr{U}_{\mathrm{IP}}$). The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R)$ for all $a \in A_2 = \{\, \mathbf{starcast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m)) \,\}$.

In mCRL2, the following derivation can be made:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{\mathbf{cast}}(\mathscr{U}_{\mathrm{IP}}, t_{\mathrm{U}(R)}, t_{\mathrm{U}(m)}) \xrightarrow{[\![\overline{\mathbf{cast}}(\mathscr{U}_{\mathrm{IP}}, t_{\mathrm{U}(R)}, t_{\mathrm{U}(m)})]\!]} \checkmark}\ \text{Axiom}}{\overline{\mathbf{cast}}(\mathscr{U}_{\mathrm{IP}}, t_{\mathrm{U}(R)}, t_{\mathrm{U}(m)}) \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], [\![t_{\mathrm{U}(R)}]\!], [\![t_{\mathrm{U}(m)}]\!])} \checkmark}\ \text{Definition 2.1}}{\overline{\mathbf{cast}}(\mathscr{U}_{\mathrm{IP}}, t_{\mathrm{U}(R)}, t_{\mathrm{U}(m)}) \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \checkmark}\ \text{Definition of } t}{\overline{\mathbf{cast}}(\mathscr{U}_{\mathrm{IP}}, t_{\mathrm{U}(R)}, t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Seq 1}}{[\![t_{\mathrm{U}(R)} \cap \mathscr{U}_{\mathrm{IP}} = t_{\mathrm{U}(R)}]\!] = true}\ \text{Guard 2}}{(t_{\mathrm{U}(R)} \cap \mathscr{U}_{\mathrm{IP}} = t_{\mathrm{U}(R)}) \to \overline{\mathbf{cast}}(\mathscr{U}_{\mathrm{IP}}, t_{\mathrm{U}(R)}, t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Substitution}}{\big(\, (t_{\mathrm{U}(R)} \cap \mathtt{D} = \mathtt{D'}) \to \overline{\mathbf{cast}}(\mathtt{D}, \mathtt{D'}, \mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \,\big)\, [\, \mathtt{D} := \mathscr{U}_{\mathrm{IP}}, \mathtt{D'} := t_{\mathrm{U}(R)}, \mathtt{msg} := t_{\mathrm{U}(m)} \,] \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Sum 2 (3 times)}}{\sum_{\mathtt{D},\mathtt{D'}:\mathtt{T}(\mathrm{Set(IP)}),\mathtt{msg}:\mathtt{T}(\mathrm{MSG})} (t_{\mathrm{U}(R)} \cap \mathtt{D} = \mathtt{D'}) \to \overline{\mathbf{cast}}(\mathtt{D}, \mathtt{D'}, \mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Choice 2}}{\sum_{\mathtt{D},\mathtt{D'}:\mathtt{T}(\mathrm{Set(IP)}),\mathtt{msg}:\mathtt{T}(\mathrm{MSG})} (t_{\mathrm{U}(R)} \cap \mathtt{D} = \mathtt{D'}) \to \overline{\mathbf{cast}}(\mathtt{D}, \mathtt{D'}, \mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) + \mathrm{S}[\, \mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)} \,] \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Substitution}}{\text{Definition of G}\ \dfrac{\big(\, \sum_{\mathtt{D},\mathtt{D'}:\mathtt{T}(\mathrm{Set(IP)}),\mathtt{msg}:\mathtt{T}(\mathrm{MSG})} (\mathtt{R} \cap \mathtt{D} = \mathtt{D'}) \to \overline{\mathbf{cast}}(\mathtt{D}, \mathtt{D'}, \mathtt{msg}).\mathrm{G}(\mathtt{ip}, \mathtt{R}) + \mathrm{S} \,\big)\, [\, \mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)} \,] \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}{\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Recursion 2}$$

where S is an expression equal to all summands of G except for the first one.

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{cast}(\llbracket \mathscr{U}_{\mathrm{IP}} \rrbracket, \mathrm{U}(R), \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')$ because Pair 2.4 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $\mathbf{broadcast}(m) \in B_1$. Combining this with the conclusion of the derivation above gives

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{cast}(\llbracket \mathscr{U}_{\mathrm{IP}} \rrbracket, \mathrm{U}(R), \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')}\ \text{Induction hypothesis} \quad \dfrac{}{\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\llbracket \mathscr{U}_{\mathrm{IP}} \rrbracket, \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textit{(see above)}}{\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{cast}(\llbracket \mathscr{U}_{\mathrm{IP}} \rrbracket, \mathrm{U}(R), \mathrm{U}(m)) | \overline{\mathbf{cast}}(\llbracket \mathscr{U}_{\mathrm{IP}} \rrbracket, \mathrm{U}(R), \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Par}\ 3}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\gamma_C(\mathbf{cast}(\llbracket \mathscr{U}_{\mathrm{IP}} \rrbracket, \mathrm{U}(R), \mathrm{U}(m)) | \overline{\mathbf{cast}}(\llbracket \mathscr{U}_{\mathrm{IP}} \rrbracket, \mathrm{U}(R), \mathrm{U}(m)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{Comm}\ 2}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{starcast}(\llbracket \mathscr{U}_{\mathrm{IP}} \rrbracket, \mathrm{U}(R), \mathrm{U}(m))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \text{Apply } \gamma_C}{\mathbf{starcast} \in V \cup \{\tau\} \quad \nabla_V \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{starcast}(\llbracket \mathscr{U}_{\mathrm{IP}} \rrbracket, \mathrm{U}(R), \mathrm{U}(m))} \equiv \nabla_V \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{Allow}\ 2}{\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{\mathbf{starcast}(\llbracket \mathscr{U}_{\mathrm{IP}} \rrbracket, \mathrm{U}(R), \mathrm{U}(m))} \equiv \mathrm{T}(ip : \mathrm{P}' : R)}\ \mathrm{T}14$$

So it is indeed the case that

$$\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R) \text{ for all } a \in A_2 = \{\, \mathbf{starcast}(\llbracket \mathscr{U}_{\mathrm{IP}} \rrbracket, \mathrm{U}(R), \mathrm{U}(m)) \,\}$$

which finishes the induction step for the case of the BROADCAST (T3) inference rule.

■

Similar proofs must be done for all other inference rules of AWN. These proofs can be found in Appendix C. Given all of the proofs, Equation 2.18 holds. ■

AWN expressions ξ, p should also be able to mimic mCRL2 expressions $\mathrm{T}(\xi, \mathrm{p})$ in accordance with Definition 4.11:

Lemma 4.8 For T_V as defined by the rules T1 to T10 in Table 2.8 it holds that

$$\begin{aligned} &\forall \mathrm{p}, a, \mathrm{Q}, \xi \;.\; \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{a} \mathrm{Q} \Rightarrow \exists (A_1, A_2) \in \mathscr{A}^{\sim}, \mathrm{q}, \sigma \;.\; a \in A_1 \\ &\quad \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{q}) \wedge \xi, \mathrm{p} \xrightarrow{A_2} \xi[\sigma], \mathrm{q} \wedge \mathrm{Q} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{q}) \end{aligned} \tag{2.19}$$

Proof. The proof of Equation 2.19 is by structural induction over the rules T1 to T10 in Table 2.8. This is a sound approach because these rules yield all mCRL2 expressions that can result from a translation by T_V. For each translation rule, all representative derivations (see Definition 4.3) for expressions of the form $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{a} \mathrm{Q}$ are listed, and for each possible derivation Lemma 4.8 is proven.

Induction hypothesis: For all recursions $\mathrm{T}_W(\xi, \mathrm{p})$ that are part of a translation rule defining T_V, it holds that

$$\begin{aligned} &\forall a, \mathrm{Q} \;.\; \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{a} \mathrm{Q} \Rightarrow \exists (A_1, A_2) \in \mathscr{A}^{\sim}, \mathrm{q}, \sigma \;.\; a \in A_1 \wedge \\ &\quad \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{q}) \wedge \xi, \mathrm{p} \xrightarrow{A_2} \xi[\sigma], \mathrm{q} \wedge \mathrm{Q} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{q}) \end{aligned}$$

Base cases: Show for translation rules T1 to T7 and T10 that for all a and Q such that $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{a}$ Q there is some $(A_1, A_2) \in \mathscr{A}^{\sim}$. $a \in A_1$ and some σ, q such that $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{q})$ and $\xi, \mathrm{p} \xrightarrow{a'} \xi[\sigma], \mathrm{q}$ can be derived for all $a' \in A_2$ and $\mathrm{Q} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{q})$.

■ **Example 2.3** The translation function T_V is partially defined by translation rule

$$\mathrm{T}_V(\xi, \mathbf{broadcast}(ms).\mathrm{p}) = \textstyle\sum_{\mathtt{D}:\mathrm{T}(\mathrm{Set(IP)})} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p}) \text{ where } \mathtt{D} \notin \mathrm{T}(V) \quad \text{T1}$$

For expressions of the form $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{broadcast}(ms).\mathrm{p}) \xrightarrow{a} \mathrm{Q}$, consider the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathrm{T}_\xi(ms)) \xrightarrow{[\![\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathrm{T}_\xi(ms))]\!]} \checkmark}\ \text{Axiom}}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathrm{T}_\xi(ms)) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(ms)]\!])} \checkmark}\ \text{Definition 2.1}}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{Seq 1}}{(\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}))[\mathtt{D} := t_{\hat{\mathtt{D}}}] \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{Substitution}}{\textstyle\sum_{\mathtt{D}:\mathrm{T}(\mathrm{Set(IP)})} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{Sum 2}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{broadcast}(ms).\mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{T1}$$

This derivation is a representative derivation without alternatives (see Definition 4.3). It follows that $a \in \{\, \mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(ms)]\!]) \mid \hat{\mathtt{D}} \in \mathrm{T}(\mathrm{Set(IP)}) \,\}$ and $\mathrm{Q} = \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{p})$ by choosing $\sigma = \emptyset$. There is exactly one pair $(A_1, A_2) \in \mathscr{A}^{\sim}.\, a \in A_1 \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{broadcast}(ms).\mathrm{p}) \xrightarrow{A_1} \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{p})$:

$$(\{\, \mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(ms)]\!]) \mid \hat{\mathtt{D}} \in \mathrm{T}(\mathrm{Set(IP)}) \,\}, \{\, \mathbf{broadcast}(\xi(ms)) \,\})$$

This pair satisfies the condition that $\xi, \mathbf{broadcast}(ms).\mathrm{p} \xrightarrow{a'} \xi[\sigma], \mathrm{p}$ can be derived for all $a' \in A_2$ (via AWN inference rule BROADCAST (T1)), which is sufficient to prove this particular base case.

■

Induction step: Given only its side conditions and the induction hypothesis, show for translation rules T8 and T9 that for all a and Q such that $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{a} \mathrm{Q}$ there is some $(A_1, A_2) \in \mathscr{A}^{\sim}.\, a \in A_1$ and some σ, q such that $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\mathrm{DOM}(\xi[\sigma]), \mathrm{q})$ and $\mathrm{p} \xrightarrow{a'} \mathrm{q}$ can be derived for all $a' \in A_2$ and $\mathrm{Q} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\mathrm{DOM}(\xi[\sigma]), \mathrm{q})$.

■ **Example 2.4** The translation function T_V is partially defined by translation rule

$$\mathrm{T}_V(\xi, \mathrm{X}(exp_1, \cdots, exp_n)) = \mathrm{X}(\mathrm{T}_\xi(exp_1), \cdots, \mathrm{T}_\xi(exp_n)) \quad \text{T8}$$
$$\text{where } \mathrm{T}(\mathrm{X}(\mathtt{var}_1, \cdots, \mathtt{var}_n) \stackrel{\text{def}}{=} \mathrm{p}) = \mathrm{X}(\mathrm{T}(\mathtt{var}_1) : \mathrm{sort}(\mathrm{T}_\xi(exp_1)), \cdots, \mathrm{T}(\mathtt{var}_n) : \mathrm{sort}(\mathrm{T}_\xi(exp_n))) \stackrel{\text{def}}{=} \mathrm{T}_{\{\mathtt{var}_1, \cdots, \mathtt{var}_n\}}(\emptyset, \mathrm{p})$$

The induction hypothesis states that recursions T_V occurring in $\mathrm{X}(\mathrm{T}_\xi(exp_1), \cdots, \mathrm{T}_\xi(exp_n))$ are related. In particular, the induction hypothesis is used here to relate arbitrary instantiations of

process calls:

$$\begin{aligned}
&\forall a, \mathrm{Q} \,.\, \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)],\mathrm{p}) \xrightarrow{a} \mathrm{Q} \Rightarrow \exists (A_1,A_2) \in \mathscr{A}^{\smile}, \mathrm{q}, \sigma \,.\, a \in A_1 \\
&\quad \wedge \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)],\mathrm{p}) \xrightarrow{A_1} \equiv \\
&\qquad\quad \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup \mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q}) \\
&\quad \wedge \emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)],\mathrm{p} \xrightarrow{A_2} \emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q} \\
&\quad \wedge \mathrm{Q} \equiv \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup \mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q})
\end{aligned}$$

The following derivation in AWN can be based on the fourth line of the instantiated induction hypothesis:

$$\dfrac{\dfrac{}{\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)],\mathrm{p} \xrightarrow{a'} \emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q}}\,\text{Induction hypothesis} \qquad \text{Definition of X}}{\xi,\mathrm{X}(exp_1,\cdots,exp_n) \xrightarrow{a'} \emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q}}\,\textsc{Recursion (T1)}$$

This derivation holds for all $a' \in A_2$. Similarly, in mCRL2, it happens to be the case that

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)],\mathrm{p}) \xrightarrow{a} \equiv \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup \mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q})}\,\text{Induction hypothesis}}{\mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}}(\emptyset,\mathrm{p})[\mathrm{T}(\mathtt{d}_1) := t_{\mathrm{U}(\xi(exp_1))},\cdots,\mathrm{T}(\mathtt{d}_n) := t_{\mathrm{U}(\xi(exp_n))}] \xrightarrow{a} \equiv \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup \mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q})}\,\text{Lemma 4.6}}{\mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}}(\emptyset,\mathrm{p})[\mathrm{T}(\mathtt{d}_1) := \mathrm{T}_\xi(exp_1),\cdots,\mathrm{T}(\mathtt{d}_n) := \mathrm{T}_\xi(exp_n)] \xrightarrow{a} \equiv \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup \mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q})}\,\substack{\text{Theorem 4.1}\\ \text{Lemma 4.3}}}{\mathrm{X}(\mathrm{T}_\xi(exp_1),\cdots,\mathrm{T}_\xi(exp_n)) \xrightarrow{a} \equiv \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup \mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q})}\,\textsc{Recursion 2}}{\mathrm{T}_V(\xi,\mathrm{X}(exp_1,\cdots,exp_n)) \xrightarrow{a} \equiv \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup \mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q})}\,\text{T8}}\,\text{Definition of X}$$

for all $a \in A_1$. This derivation is a representative derivation without alternatives (see Definition 4.3).

Because the induction hypothesis is used to express the relationship between *arbitrary* instantiations of process calls, the first line of this derivation plus the inference rules of mCRL2 are sufficient to generate all possible behavior of an mCRL2 process call. The actions available to $\mathrm{T}_V(\xi,\mathrm{X}(exp_1,\cdots,exp_n))$ in mCRL2 can therefore be mimicked by AWN, and so both derivations can be combined into an new equation matching the induction hypothesis:

$$\begin{aligned}
&\forall a, \mathrm{Q} \,.\, \mathrm{T}_V(\xi,\mathrm{X}(exp_1,\cdots,exp_n)) \xrightarrow{a} \mathrm{Q} \Rightarrow \exists (A_1,A_2) \in \mathscr{A}^{\smile}, \mathrm{q}, \sigma \,.\, a \in A_1 \\
&\quad \wedge \mathrm{T}_V(\xi,\mathrm{X}(exp_1,\cdots,exp_n)) \xrightarrow{A_1} \equiv \\
&\qquad\quad \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup \mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q}) \\
&\quad \wedge \emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)],\mathrm{p} \xrightarrow{A_2} \emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q} \\
&\quad \wedge \mathrm{Q} \equiv \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup \mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q})
\end{aligned}$$

This confirms the induction step of the proof of Lemma 4.8.

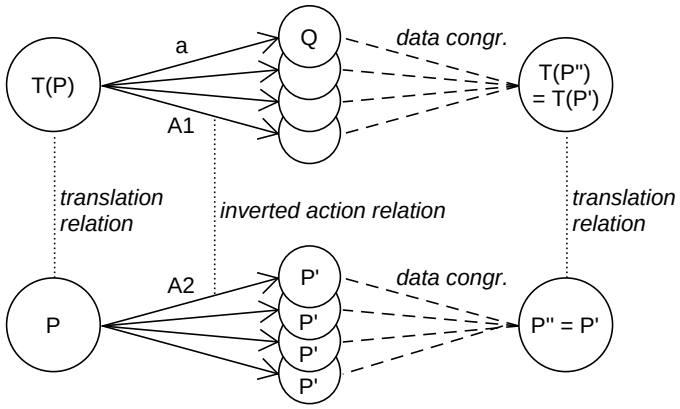
■

Given the proofs for translation rules T1 to T10 (see Appendix D), Equation 2.19 holds. ■

In addition to Lemma 4.7, it should also be the case that AWN expressions P can mimic mCRL2 expressions $\mathrm{T}(\mathrm{p})$ in general (once more in accordance with Definition 4.11):

Lemma 4.9 Let $\widetilde{\mathrm{T}}^{\smile}$ be the converse of translation relation $\widetilde{\mathrm{T}}$ from Equation 2.1 and let $\mathscr{A}^{\smile}$ be the converse of relation $\mathscr{A}$ from Table 2.10. Then $\widetilde{\mathrm{T}}^{\smile}$ is a strong $\mathscr{A}^{\smile}$-warped simulation of mCRL2 expressions $\mathrm{T}(\mathrm{P})$ by AWN expressions P for all AWN expressions P.

Proof. For all actions that $\mathrm{T}(\mathrm{P})$ can do, it must be shown that the resulting state is data congruent with $\mathrm{T}(\mathrm{P}')$. A depiction of the requirements for the proof in this direction is given below:



Note that data congruence is implied by the assumption that $\mathrm{P}'' = \mathrm{P}'$. In combination with Lemma 3.1, this means that for Definition 4.11 to apply it is sufficient to prove that

$$\forall \mathrm{P}, a, \mathrm{Q}' \,.\, \mathrm{T}(\mathrm{P}) \xrightarrow{a} \mathrm{Q} \Rightarrow \exists (A_1, A_2) \in \mathscr{A}^{\smile}, \mathrm{P}' \,.\, a \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}') \wedge \mathrm{P} \xrightarrow{A_2} \mathrm{P}' \wedge \mathrm{Q} \equiv \mathrm{T}(\mathrm{P}') \quad (2.20)$$

The proof of Equation 2.20 is provided by structural induction over the translation rules of mCRL2, which is possible because these yield all possible mCRL2 expressions that can result from a translation by T.

Induction hypothesis: For all recursions $\mathrm{T}(\mathrm{P})$ that are part of a translation rule defining T, it holds that

$$\forall a, \mathrm{Q} \,.\, \mathrm{T}(\mathrm{P}) \xrightarrow{a} \mathrm{Q} \Rightarrow \exists (A_1, A_2) \in \mathscr{A}^{\smile}, \mathrm{P}' \,.\, a \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}') \wedge \mathrm{P} \xrightarrow{A_2} \mathrm{P}' \wedge \mathrm{Q} \equiv \mathrm{T}(\mathrm{P}')$$

Base cases: Show for translation rules T11 and T12 that for all a and Q such that $\mathrm{T}(\mathrm{P}) \xrightarrow{a} \mathrm{Q}$ there is some $(A_1, A_2) \in \mathscr{A}^{\smile} \,.\, a \in A_1$ and some P' such that $\mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}')$ and $\mathrm{P} \xrightarrow{a'} \mathrm{P}$ can be derived for all $a' \in A_2$ and $\mathrm{Q} \equiv \mathrm{T}(\mathrm{P}')$.

■ **Example 2.5** The translation function T is partially defined by translation rule T12:

$$\mathrm{T}(\xi,\mathrm{p}) = \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})$$

Suppose that $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \xrightarrow{a} \mathrm{Q}$. Then, according to Lemma 4.8, there exists some $(A_1,A_2) \in \mathscr{A}^{\sim}$. $a \in A_1$ and some σ,p' such that $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p}')$ and $\xi,\mathrm{p} \xrightarrow{a'} \xi[\sigma],\mathrm{p}'$ for all $a' \in A_2$ and $\mathrm{Q} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p}')$.
Define $\zeta = \xi[\sigma]$. Q is also data congruent with $\mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p}')$ which, as stated by rule T12, equals $\mathrm{T}(\zeta,\mathrm{p}')$. As a result, there must exist some $(A_1,A_2) \in \mathscr{A}^{\sim}$. $a \in A_1$ and some p' so that $\mathrm{T}(\xi,\mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}(\zeta,\mathrm{p}')$ and $\xi,\mathrm{p} \xrightarrow{a'} \zeta,\mathrm{p}'$ for all $a' \in A_2$ and $\mathrm{Q} \equiv \mathrm{T}(\zeta,\mathrm{p}')$, proving this particular base case.

■

Induction step: Given only their side conditions and the induction hypothesis, show for translation rules T13 to T16 that for all a and Q such that $\mathrm{T}(\mathrm{P}) \xrightarrow{a} \mathrm{Q}$ there is some $(A_1,A_2) \in \mathscr{A}^{\sim}$. $a \in A_1$ and some P' such that $\mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}')$ and $\mathrm{P} \xrightarrow{a'} \mathrm{P}'$ can be derived for all $a' \in A_2$ and $\mathrm{Q} \equiv \mathrm{T}(\mathrm{P}')$.

■ **Example 2.6** The translation function T is partially defined by translation rule

$$\mathrm{T}([\,\mathrm{M}\,]) = \nabla_V \rho_{\{\mathbf{starcast}\to\mathbf{t}\}} \Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \quad \text{T16}$$
$$\text{where } V = \{\mathbf{t}, \mathbf{newpkt}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}$$
$$\text{where } C = \{\overline{\mathbf{newpkt}} | \mathbf{arrive} \to \mathbf{newpkt}\}$$
$$\text{where } \mathrm{H} = \textstyle\sum_{\mathtt{ip}:\mathrm{T}(\mathrm{IP}),\mathtt{data}:\mathrm{T}(\mathrm{DATA}),\mathtt{dest}:\mathrm{T}(\mathrm{IP})} \overline{\mathbf{newpkt}}(\{\mathtt{ip}\},\{\mathtt{ip}\},\mathtt{newpkt}(\mathtt{data},\mathtt{dest}))\mathrm{.H}$$

Consider two pairs from Table 2.10:

$$(\{\, H\neg K : \mathbf{arrive}(m) \,\}, \left\{ \mathbf{arrive}(\hat{\mathtt{D}},\hat{\mathtt{D}}\mathtt{'},\mathrm{U}(m)) \;\middle|\; \begin{array}{c} \hat{\mathtt{D}},\hat{\mathtt{D}}\mathtt{'}\in\mathrm{T}(\mathrm{Set(IP)}) \\ \hat{\mathtt{D}}\mathtt{'}\subseteq\hat{\mathtt{D}} \\ \mathrm{U}(H)\subseteq\hat{\mathtt{D}}\mathtt{'} \\ \mathrm{U}(K)\cap\hat{\mathtt{D}}\mathtt{'}=\emptyset \end{array} \right\}),$$
$$(\{\, ip : \mathbf{newpkt}(d,dip) \,\}, \{\, \mathbf{newpkt}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip))) \,\})$$

Let A_1 and N_1 be defined as the second members of these pairs; that is, let

$$A_1 \stackrel{\text{def}}{=} \left\{ \mathbf{arrive}(\hat{\mathtt{D}},\hat{\mathtt{D}}\mathtt{'},\mathrm{U}(m)) \;\middle|\; \begin{array}{c} \hat{\mathtt{D}},\hat{\mathtt{D}}\mathtt{'}\in\mathrm{T}(\mathrm{Set(IP)}) \\ \hat{\mathtt{D}}\mathtt{'}\subseteq\hat{\mathtt{D}} \\ \mathrm{U}(H)\subseteq\hat{\mathtt{D}}\mathtt{'} \\ \mathrm{U}(K)\cap\hat{\mathtt{D}}\mathtt{'}=\emptyset \end{array} \right\}$$
$$N_1 \stackrel{\text{def}}{=} \{\, \mathbf{newpkt}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip))) \,\}$$

and let

$$\overline{N}_1 \stackrel{\text{def}}{=} \{\, \overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip))) \,\}$$

for some $d \in \mathrm{DATA}$, $ip \in \mathrm{IP}$, $H,K \in \mathrm{Set(IP)}$, and $m \in \mathrm{MSG}$.

The following cases are distinguished:

1: $\mathrm{T}(\mathrm{M})$ does a transition $a \in A_1$ and H does a transition $b \in \overline{N}_1$.
2: $\mathrm{T}(\mathrm{M})$ does a transition $a \notin A_1$ and H does a transition $b \in \overline{N}_1$.
3: $\mathrm{T}(\mathrm{M})$ does nothing and H does a transition $b \in \overline{N}_1$.
4: $\mathrm{T}(\mathrm{M})$ does a transition a where $\underline{a} = \mathbf{starcast}$ and H does nothing.
5: $\mathrm{T}(\mathrm{M})$ does a transition a where $\underline{a} \in \{\mathbf{t}, \mathbf{newpkt}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}$ and H does nothing.
6: $\mathrm{T}(\mathrm{M})$ does a transition a where $\underline{a} \notin \{\mathbf{t}, \mathbf{starcast}, \mathbf{newpkt}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}$ and H does nothing.

Note that these cases cover all combinations of behavior of $\mathrm{T}(\mathrm{M})$ and H (H can only do $\overline{\mathbf{newpkt}}$ actions; see the first part of the derivation in the Lemma 4.7-proof for Newpkt (T4), which is a representative derivation without alternatives).

The proof is provided below for each of the cases:

1: $\mathrm{T}(\mathrm{M})$ does a transition $a \in A_1$ and H does a transition $b \in \overline{N}_1$. Following the induction hypothesis and eliminating pairs from the action relation $\mathscr{A}$ in Table 2.10 that do not match the transition labels from A_1, it can first be concluded that

$$\mathrm{T}(\mathrm{M}) \xrightarrow{L_1} \equiv \mathrm{T}(\mathrm{M}') \wedge \mathrm{M} \xrightarrow{H\neg K:\mathbf{arrive}(m)} \mathrm{M}'$$

The mCRL2 derivation in the Lemma 4.7-proof for Newpkt (T4) shows how the first conjunct is sufficient to prove that $\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{L_1} \equiv \mathrm{T}([\,\mathrm{M}'\,])$. Under the constraints of this case, the derivation is a representative derivation without alternatives (see Definition 4.3) and so all related behavior of $\mathrm{T}([\,\mathrm{M}\,])$ is covered.
On the AWN side, the second conjunct can be used as premise in

$$\frac{\mathrm{M} \xrightarrow{\{ip\}\neg K:\mathbf{arrive}(\mathtt{newpkt}(d,dip))} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{ip:\mathbf{newpkt}(d,dip)} [\,\mathrm{M}'\,]}\ \textsc{Newpkt (T4)}$$

This case is proven if there exists a pair $(A_1, A_2) \in \mathscr{A}^{\sim}$ that satisfies $a \in A_1 \wedge \mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{A_1} \equiv \mathrm{T}([\,\mathrm{M}'\,]) \wedge [\,\mathrm{M}\,] \xrightarrow{A_2} [\,\mathrm{M}'\,]$, and the converse of Pair 2.16 satisfies this requirement.

2: $\mathrm{T}(\mathrm{M})$ does a transition $a \notin A_1$ and H does a transition $b \in \overline{N}_1$. This situation fails to generate behavior for $\mathrm{T}([\,\mathrm{M}\,])$ because no derivation similar to the one in the Lemma 4.7-proof for Newpkt (T4) is possible:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{T}(\mathrm{M}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{M}') \qquad \mathrm{H} \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathtt{U}(ip)\},\{\mathtt{U}(ip)\},\mathtt{newpkt}(\mathtt{U}(d),\mathtt{U}(dip)))} \mathrm{H}}{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H} \xrightarrow{a|\overline{\mathbf{newpkt}}(\{\mathtt{U}(ip)\},\{\mathtt{U}(ip)\},\mathtt{newpkt}(\mathtt{U}(d),\mathtt{U}(dip)))} \equiv \mathrm{T}(\mathrm{M}') \,||\, \mathrm{H}}\ \textsc{Par 3}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\gamma_C(a|\overline{\mathbf{newpkt}}(\{\mathtt{U}(ip)\},\{\mathtt{U}(ip)\},\mathtt{newpkt}(\mathtt{U}(d),\mathtt{U}(dip))))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{a|\overline{\mathbf{newpkt}}(\{\mathtt{U}(ip)\},\{\mathtt{U}(ip)\},\mathtt{newpkt}(\mathtt{U}(d),\mathtt{U}(dip)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \gamma_C}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\{\mathbf{starcast}\to\mathbf{t}\}\bullet a|\overline{\mathbf{newpkt}}(\{\mathtt{U}(ip)\},\{\mathtt{U}(ip)\},\mathtt{newpkt}(\mathtt{U}(d),\mathtt{U}(dip)))} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Rename 2}}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{a|\overline{\mathbf{newpkt}}(\{\mathtt{U}(ip)\},\{\mathtt{U}(ip)\},\mathtt{newpkt}(\mathtt{U}(d),\mathtt{U}(dip)))} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \{\mathbf{starcast}\to\mathbf{t}\}\bullet$$

The ALLOW 2 operator cannot be applied next because $\underline{a}|\overline{\mathbf{newpkt}} \notin V \cup \{\tau\}$. Since this means that $\mathrm{T}([\,\mathrm{M}\,])$ cannot do a transition in mCRL2 under the circumstances specified in this particular case, there is no behavior for AWN to mimic.

3: $\mathrm{T}(\mathrm{M})$ does nothing and H does a transition $\overline{\mathbf{newpkt}}(\{\mathtt{U}(ip)\},\{\mathtt{U}(ip)\},\mathtt{newpkt}(\mathtt{U}(d),\mathtt{U}(dip))) \in \overline{N}_1$. This situation fails to generate behavior for $\mathrm{T}([\,\mathrm{M}\,])$ because no derivation similar to the one in the Lemma 4.7-proof for Newpkt (T4) is possible:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{H} \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathtt{U}(ip)\},\{\mathtt{U}(ip)\},\mathtt{newpkt}(\mathtt{U}(d),\mathtt{U}(dip)))} \mathrm{H}}{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H} \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathtt{U}(ip)\},\{\mathtt{U}(ip)\},\mathtt{newpkt}(\mathtt{U}(d),\mathtt{U}(dip)))} \mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}}\,\text{PAR 5}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\gamma_C(\overline{\mathbf{newpkt}}(\{\mathtt{U}(ip)\},\{\mathtt{U}(ip)\},\mathtt{newpkt}(\mathtt{U}(d),\mathtt{U}(dip))))} \Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H})}\,\text{COMM 2}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathtt{U}(ip)\},\{\mathtt{U}(ip)\},\mathtt{newpkt}(\mathtt{U}(d),\mathtt{U}(dip)))} \Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H})}\,\text{Apply } \gamma_C}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\{\mathbf{starcast}\to\mathbf{t}\}\bullet\overline{\mathbf{newpkt}}(\{\mathtt{U}(ip)\},\{\mathtt{U}(ip)\},\mathtt{newpkt}(\mathtt{U}(d),\mathtt{U}(dip)))} \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H})}\,\text{RENAME 2}}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathtt{U}(ip)\},\{\mathtt{U}(ip)\},\mathtt{newpkt}(\mathtt{U}(d),\mathtt{U}(dip)))} \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H})}\,\text{Apply } \{\mathbf{starcast}\to\mathbf{t}\}\bullet$$

The ALLOW 2 operator cannot be applied next because $\overline{\mathbf{newpkt}} \notin V \cup \{\tau\}$. Since this means that $\mathrm{T}([\,\mathrm{M}\,])$ cannot do a transition in mCRL2 under the circumstances specified in this particular case, there is no behavior for AWN to mimic.

4: $\mathrm{T}(\mathrm{M})$ does a transition a where $\underline{a} = \mathbf{starcast}$ and H does nothing.
Following the induction hypothesis and eliminating pairs from the action relation $\mathscr{A}$ in Table 2.10 that do not match the transition label **starcast**, it can first be concluded that

$$\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{starcast}(\mathtt{U}(D),\mathtt{U}(R),\mathtt{U}(m))} \equiv \mathrm{T}(\mathrm{M}') \wedge \mathrm{M} \xrightarrow{R:\ast\mathbf{cast}(m)} \mathrm{M}'$$

for all $D,R \in \mathrm{Set(IP)}$ and $m \in \mathrm{MSG}$ where $R \subseteq D$.
The mCRL2 derivation in the Lemma 4.7-proof for Cast (T4-4) shows how the first conjunct is sufficient to prove that $\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{\mathbf{t}(\mathtt{U}(D),\mathtt{U}(R),\mathtt{U}(m))} \equiv \mathrm{T}([\,\mathrm{M}'\,])$. Under the constraints of this case, the derivation is a representative derivation without alternatives (see Definition 4.3) and so all related behavior of $\mathrm{T}([\,\mathrm{M}\,])$ is covered.
On the AWN side, the second conjunct can be used as premise in

$$\dfrac{\mathrm{M} \xrightarrow{R:\ast\mathbf{cast}(m)} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{\tau} [\,\mathrm{M}'\,]}\,\text{CAST (T4-4)}$$

This case is proven if there exists a pair $(A_1,A_2) \in \mathscr{A}^{\sim}$ that satisfies $a \in A_1 \wedge \mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{A_1} \equiv \mathrm{T}([\,\mathrm{M}'\,]) \wedge [\,\mathrm{M}\,] \xrightarrow{A_2} [\,\mathrm{M}'\,]$, and the converse of Pair 2.11 satisfies this requirement.

5: $\mathrm{T}(\mathrm{M})$ does a transition a where $\underline{a} \in \{\mathbf{t},\mathbf{newpkt},\mathbf{deliver},\mathbf{connect},\mathbf{disconnect}\}$ and H does nothing. The induction hypothesis states that

$$\exists (A_1,A_2) \in \mathscr{A}^{\sim} \,.\, a \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}') \wedge \mathrm{P} \xrightarrow{A_2} \mathrm{P}'$$

Define $\mathscr{A}'$ as a subset of $\mathscr{A}^{\sim}$ that contains the possible values of $(A_1,A_2) \in \mathscr{A}^{\sim}$ where A_1 are actions that $\mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q})$ can perform in mCRL2 and where A_2 are the actions that AWN should use to mimic the actions in A_1 in order for this case to be proven:

$$\mathscr{A}' \stackrel{\text{def}}{=} \left\{ (A_1,A_2) \;\middle|\; (A_1,A_2) \in \mathscr{A}^{\sim}, \quad \mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}' \langle\langle \mathrm{Q}) \right\}$$

The following derivation is possible in mCRL2 for all $a \in A_1$ such that $(A_1, A_2) \in \mathscr{A}'$:

$$a \in V \cup \{\tau\} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T(M)} \xrightarrow{a} \equiv \mathrm{T(M')}}\text{ Induction hypothesis}}{\mathrm{T(M)} \mid\mid \mathrm{H} \xrightarrow{a} \equiv \mathrm{T(M')} \mid\mid \mathrm{H}}\text{ PAR 2}}{\Gamma_C(\mathrm{T(M)} \mid\mid \mathrm{H}) \xrightarrow{\gamma_C(a)} \equiv \Gamma_C(\mathrm{T(M')} \mid\mid \mathrm{H})}\text{ COMM 2}}{\Gamma_C(\mathrm{T(M)} \mid\mid \mathrm{H}) \xrightarrow{a} \equiv \Gamma_C(\mathrm{T(M')} \mid\mid \mathrm{H})}\text{ Apply } \gamma_C}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T(M)} \mid\mid \mathrm{H}) \xrightarrow{\{\mathbf{starcast}\to\mathbf{t}\}\bullet a} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T(M')} \mid\mid \mathrm{H})}\text{ RENAME 2}}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T(M)} \mid\mid \mathrm{H}) \xrightarrow{a} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T(M')} \mid\mid \mathrm{H})}\text{ Apply } \{\mathbf{starcast} \to \mathbf{t}\}\bullet}{\nabla_V\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T(M)} \mid\mid \mathrm{H}) \xrightarrow{a} \equiv \nabla_V\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T(M')} \mid\mid \mathrm{H})}\text{ ALLOW 2}}{\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{a} \equiv \mathrm{T}([\,\mathrm{M'}\,])}\text{ T16}$$

This derivation is valid only for $a \in A_1$ such that $\underline{a} \in V \cup \{\tau\}$. Under the constraints of this case, the derivation is also a representative derivation without alternatives (see Definition 4.3). Finally it must be observed that it is impossible for $\mathrm{T(M)}$ to produce a **newpkt** action independent of H, meaning that $\underline{a} \neq$ **newpkt**. Consequently, $\mathscr{A}'$ is as follows:
{

$$\begin{aligned}
&(\{\, \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \,\}, \{\, \tau \,\}),\\
&(\{\, \mathbf{deliver}(\mathrm{U}(ip), \mathrm{U}(d)) \,\}, \{\, ip : \mathbf{deliver}(d) \,\}),\\
&(\{\, \mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip'')) \,\}, \{\, \mathbf{connect}(ip', ip'') \,\}),\\
&(\{\, \mathbf{disconnect}(\mathrm{U}(ip'), \mathrm{U}(ip'')) \,\}, \{\, \mathbf{disconnect}(ip', ip'') \,\})
\end{aligned}$$

| $m \in \mathrm{MSG}$; $ip, ip', ip'' \in \mathrm{IP}$; $d \in \mathrm{DATA}$; $dests, R, D \in \mathrm{Set(IP)}$; $R \subseteq D$ }

For all $(A_1, A_2) \in \mathscr{A}'$ the actions A_1 in mCRL2 can be mimicked by the actions A_2 in AWN by means of the inference rules

$$\frac{\mathrm{M} \xrightarrow{\tau} \mathrm{M'}}{[\,\mathrm{M}\,] \xrightarrow{\tau} [\,\mathrm{M'}\,]}\text{ INTERNAL (T4-3)}$$

$$\frac{\mathrm{M} \xrightarrow{ip:\mathbf{deliver}(d)} \mathrm{M'}}{[\,\mathrm{M}\,] \xrightarrow{ip:\mathbf{deliver}(d)} [\,\mathrm{M'}\,]}\text{ DELIVER (T4-3)}$$

$$\frac{\mathrm{M} \xrightarrow{\mathbf{connect}(ip,ip')} \mathrm{M'}}{[\,\mathrm{M}\,] \xrightarrow{\mathbf{connect}(ip,ip')} [\,\mathrm{M'}\,]}\text{ CONNECT (T4-2)}$$

$$\frac{\mathrm{M} \xrightarrow{\mathbf{disconnect}(ip,ip')} \mathrm{M'}}{[\,\mathrm{M}\,] \xrightarrow{\mathbf{disconnect}(ip,ip')} [\,\mathrm{M'}\,]}\text{ DISCONNECT (T4-2)}$$

respectively. Therefore the induction hypothesis holds for this case.

6: $\mathrm{T}(\mathrm{M})$ does a transition a where $\underline{a} \notin \{\mathbf{t}, \mathbf{starcast}, \mathbf{newpkt}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}$ and H does nothing. This situation fails to generate behavior for $\mathrm{T}([\,\mathrm{M}\,])$ because no derivation similar to the one in the Lemma 4.7-proof for Newpkt (T4) is possible:

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{T}(\mathrm{M}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{M}')}{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H} \xrightarrow{a} \equiv \mathrm{T}(\mathrm{M}') \,||\, \mathrm{H}}\,\textsc{Par}\ 2}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\gamma_C(a)} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\,\textsc{Comm}\ 2}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{a} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\,\text{Apply } \gamma_C}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\{\mathbf{starcast}\to\mathbf{t}\}\bullet a} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\,\textsc{Rename}\ 2$$

The $\textsc{Allow}\ 2$ operator cannot be applied next because $\underline{a} \notin V \cup \{\tau\}$. Since this means that $\mathrm{T}([\,\mathrm{M}\,])$ cannot do a transition in mCRL2 under the circumstances specified in this particular case, there is no behavior for AWN to mimic.

■

Similar proofs must be done for all other translation rules defining T. These proofs can be found in Appendix E. Given all of the proofs, Equation 2.20 holds. ■

Finally, Lemmas 4.7 and 4.9 can be united in a single theorem:

Theorem 4.10 Take translation relation $\widetilde{\mathrm{T}}$ from Equation 2.1 and action relation $\mathscr{A}$ from Table 2.10. Then $\widetilde{\mathrm{T}}$ is a strong $\mathscr{A}$-warped bisimulation up to $\equiv$ (see Definition 4.12) between AWN expressions P and mCRL2 expressions $\mathrm{T}(\mathrm{P})$ for all AWN expressions P.

Proof. $\widetilde{\mathrm{T}}$ is a strong $\mathscr{A}$-warped bisimulation up to $\equiv$ of P by $\mathrm{T}(\mathrm{P})$ for all P if and only if Definition 4.12 applies. This definition consists of two conditions, the first of which is established by the proof for Lemma 4.7 and the second of which is established by the proof for Lemma 4.9. ■

4.5 Proof for strong bisimulation modulo renaming

The definition of strong warped bisimulation in the previous section is just a stepping stone towards proving a more commonly known type of strong bisimulation, namely *strong bisimulation modulo renaming*:

Definition 4.13 Let $\mathrm{LTS}_1 = (S_1, \mathrm{Act}_1, \rightarrow_1)$ and $\mathrm{LTS}_2 = (S_2, \mathrm{Act}_2, \rightarrow_2)$ be labeled transition systems where $\rightarrow_i \subseteq S_i \times \mathrm{Act}_i \times S_i$ and let $\mathfrak{f} : \mathrm{Act}_1 \rightarrow \mathrm{Act}_2$ be a bijective function. Then $R \subseteq S_1 \times S_2$ is called a *strong simulation up to* $\equiv$ *of* LTS_1 *by* LTS_2 *modulo renaming by* $\mathfrak{f}$ if and only if

$$\forall (\mathrm{p},\mathrm{q}) \in R, a \in \mathrm{Act}_1, \mathrm{p}' \in S_1 \,.\, \mathrm{p} \xrightarrow{a}_2 \mathrm{p}' \Rightarrow \exists \mathrm{q}' \in S_2 \,.\, \mathrm{q} \xrightarrow{\mathfrak{f}(a)}_2 \equiv \mathrm{q}' \wedge (\mathrm{p}', \mathrm{q}') \in R$$

Definition 4.14 Let $\mathrm{LTS}_1 = (S_1, \mathrm{Act}_1, \rightarrow_1)$ and $\mathrm{LTS}_2 = (S_2, \mathrm{Act}, \rightarrow_2)$ be labeled transition systems where $\rightarrow_i \subseteq S_i \times \mathrm{Act} \times S_i$ and let $\mathfrak{f} : \mathrm{Act}_1 \rightarrow \mathrm{Act}_2$ be a bijective function. Then $R \subseteq S_1 \times S_2$ is called a *strong bisimulation up to* $\equiv$ *between* LTS_1 *and* LTS_2 *modulo renaming by* $\mathfrak{f}$ if and only if

- R is a simulation up to $\equiv$ of LTS_1 by LTS_2 modulo renaming by $\mathfrak{f}$ and
- $R^\smile$ is a simulation up to $\equiv$ of LTS_2 by LTS_1 modulo renaming by $\mathfrak{f}^{-1}$.

The final theorem that must be proven is as follows:

Theorem 4.11 Take translation relation $\widetilde{\mathrm{T}}_\tau$ from Equation 2.2. Then there exists a bijective function $\mathfrak{f} : \mathrm{Act}_{\mathrm{AWN}} \rightarrow \mathrm{Act}_{\mathrm{mCRL2}}$ so that $\widetilde{\mathrm{T}}_\tau$ is a strong bisimulation up to $\equiv$ between AWN expressions $[\,\mathrm{M}\,]$ and mCRL2 expressions $\tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}\,])$ modulo renaming by $\mathfrak{f}$ for all AWN expressions $[\,\mathrm{M}\,]$.

Proof. The inference rules of AWN (see Tables 2.1 and 2.2) clearly show that

$$\forall \mathrm{M}, \mathrm{Q}, a \,.\, [\,\mathrm{M}\,] \xrightarrow{a} \mathrm{Q} \quad \Rightarrow \quad \exists \mathrm{M}' \,.\, \mathrm{Q} = [\,\mathrm{M}'\,]$$

which means that Equation 2.18 implies that

$$\forall \mathrm{M}, \mathrm{M}', a \,.\, [\,\mathrm{M}\,] \xrightarrow{a} [\,\mathrm{M}'\,] \Rightarrow \qquad (2.21)$$
$$\exists (A_1, A_2) \in \mathscr{A} \,.\, a \in A_1 \wedge [\,\mathrm{M}\,] \xrightarrow{A_1} [\,\mathrm{M}'\,] \wedge \mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{A_2} \equiv \mathrm{T}([\,\mathrm{M}'\,])$$

Similarly, the Lemma 4.9-proof for Translation rule T16 shows that

$$\forall \mathrm{M}, \mathrm{Q}, a \,.\, [\,\mathrm{T}(\mathrm{M})\,] \xrightarrow{a} \equiv \mathrm{Q} \quad \Rightarrow \quad \exists \mathrm{M}' \,.\, \mathrm{Q} = \mathrm{T}([\,\mathrm{M}'\,])$$

which means that Equation 2.20 implies that

$$\forall \mathrm{M}, \mathrm{M}', a' \,.\, \mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{a'} \equiv \mathrm{T}([\,\mathrm{M}'\,]) \Rightarrow \qquad (2.22)$$
$$\exists (A_1, A_2) \in \mathscr{A}^\smile \,.\, a' \in A_1 \wedge \mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{A_1} \equiv \mathrm{T}([\,\mathrm{M}'\,]) \wedge [\,\mathrm{M}\,] \xrightarrow{A_2} [\,\mathrm{M}'\,]$$

Equations 2.21 and 2.22 only require a subset $\mathscr{A}'$ of $\mathscr{A}$ in order to be valid, namely the subset containing the pairs in $\mathscr{A}$ with the actions that $[\,\mathrm{M}\,]$ and $\mathrm{T}([\,\mathrm{M}\,])$ can actually perform. To determine the contents of $\mathscr{A}'$, the inference rules of AWN and the Lemma 4.9-proof for Translation rule T16 can be used once again because they exhaustively list all possible behavior of $[\,\mathrm{M}\,]$ and $\mathrm{T}([\,\mathrm{M}\,])$. This results in the following value of $\mathscr{A}'$:

$$\begin{aligned}\mathscr{A}' = \{\ & \\ & (\{\,\tau\,\},\{\,\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))\,\}),\\ & (\{\,ip:\mathbf{deliver}(d)\,\},\{\,\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))\,\}),\\ & (\{\,\mathbf{connect}(ip',ip'')\,\},\{\,\mathbf{connect}(\mathrm{U}(ip'),\mathrm{U}(ip''))\,\}),\\ & (\{\,\mathbf{disconnect}(ip',ip'')\,\},\{\,\mathbf{disconnect}(\mathrm{U}(ip'),\mathrm{U}(ip''))\,\}),\\ & (\{\,ip:\mathbf{newpkt}(d,dip)\,\},\{\,\mathbf{newpkt}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))\,\})\\ & \\ \mid\ & m\in\mathrm{MSG};\quad ip,ip',ip'',dip\in\mathrm{IP};\quad d\in\mathrm{DATA};\quad R,D\in\mathrm{Set(IP)};\quad R\subseteq D\quad\}\end{aligned}$$

Furthermore, the different manifestations of the **t** action are not visible at the network level, and therefore they can be hidden in mCRL2 using a $\tau_{\{\mathbf{t}\}}$ operation:

$$\frac{\mathrm{T}([\,\mathrm{M}\,])\xrightarrow{a'}\equiv\mathrm{T}([\,\mathrm{M}'\,])}{\tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}\,])\xrightarrow{\theta_{\{\mathbf{t}\}}(a')}\equiv\tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}'\,])}\ \textsc{Hide}\ 2$$

This changes Equations 2.21 and 2.22 and the value of $\mathscr{A}'$ to the following:

$$\begin{aligned}&\forall\mathrm{M},\mathrm{M}',a\,.[\,\mathrm{M}\,]\xrightarrow{a}[\,\mathrm{M}'\,]\Rightarrow\\ &\qquad\exists(A_1,A_2)\in\mathscr{A}'_\tau\,.\,a\in A_1\wedge[\,\mathrm{M}\,]\xrightarrow{A_1}[\,\mathrm{M}'\,]\wedge\tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}\,])\xrightarrow{A_2}\equiv\tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}'\,])\\ &\forall\mathrm{M},\mathrm{M}',a'\,.\tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}\,])\xrightarrow{a'}\equiv\tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}'\,])\Rightarrow\\ &\qquad\exists(A_1,A_2)\in\mathscr{A}'^{\smile}_\tau\,.\,a'\in A_1\tau_{\{\mathbf{t}\}}\wedge\tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}\,])\xrightarrow{A_1}\equiv\tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}'\,])\wedge[\,\mathrm{M}\,]\xrightarrow{A_2}[\,\mathrm{M}'\,]\end{aligned}$$

where

$$\begin{aligned}\mathscr{A}'_\tau\stackrel{\mathrm{def}}{=}\{\ & \\ & (\{\,\tau\,\},\{\,\tau\,\}),\\ & (\{\,ip:\mathbf{deliver}(d)\,\},\{\,\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))\,\}),\\ & (\{\,\mathbf{connect}(ip',ip'')\,\},\{\,\mathbf{connect}(\mathrm{U}(ip'),\mathrm{U}(ip''))\,\}),\\ & (\{\,\mathbf{disconnect}(ip',ip'')\,\},\{\,\mathbf{disconnect}(\mathrm{U}(ip'),\mathrm{U}(ip''))\,\}),\\ & (\{\,ip:\mathbf{newpkt}(d,dip)\,\},\{\,\mathbf{newpkt}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))\,\})\\ & \\ \mid\ & ip,ip',ip'',dip\in\mathrm{IP};\quad d\in\mathrm{DATA}\quad\}\end{aligned}$$

Note that all pairs in $\mathscr{A}'_\tau$ are singleton sets, which means that they can be replaced by the bijective function $\mathfrak{f}$:

$$\forall \mathrm{M}, \mathrm{M}', a \,.\, \left([\,\mathrm{M}\,] \xrightarrow{a} [\,\mathrm{M}'\,] \Rightarrow \tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{\mathfrak{f}(a)} \equiv \tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}'\,]) \right) \tag{2.23}$$

$$\forall \mathrm{M}, \mathrm{M}', a' \,.\, \left(\tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{a'} \equiv \tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}'\,]) \Rightarrow [\,\mathrm{M}\,] \xrightarrow{\mathfrak{f}^{-1}(a')} \equiv [\,\mathrm{M}'\,] \right) \tag{2.24}$$

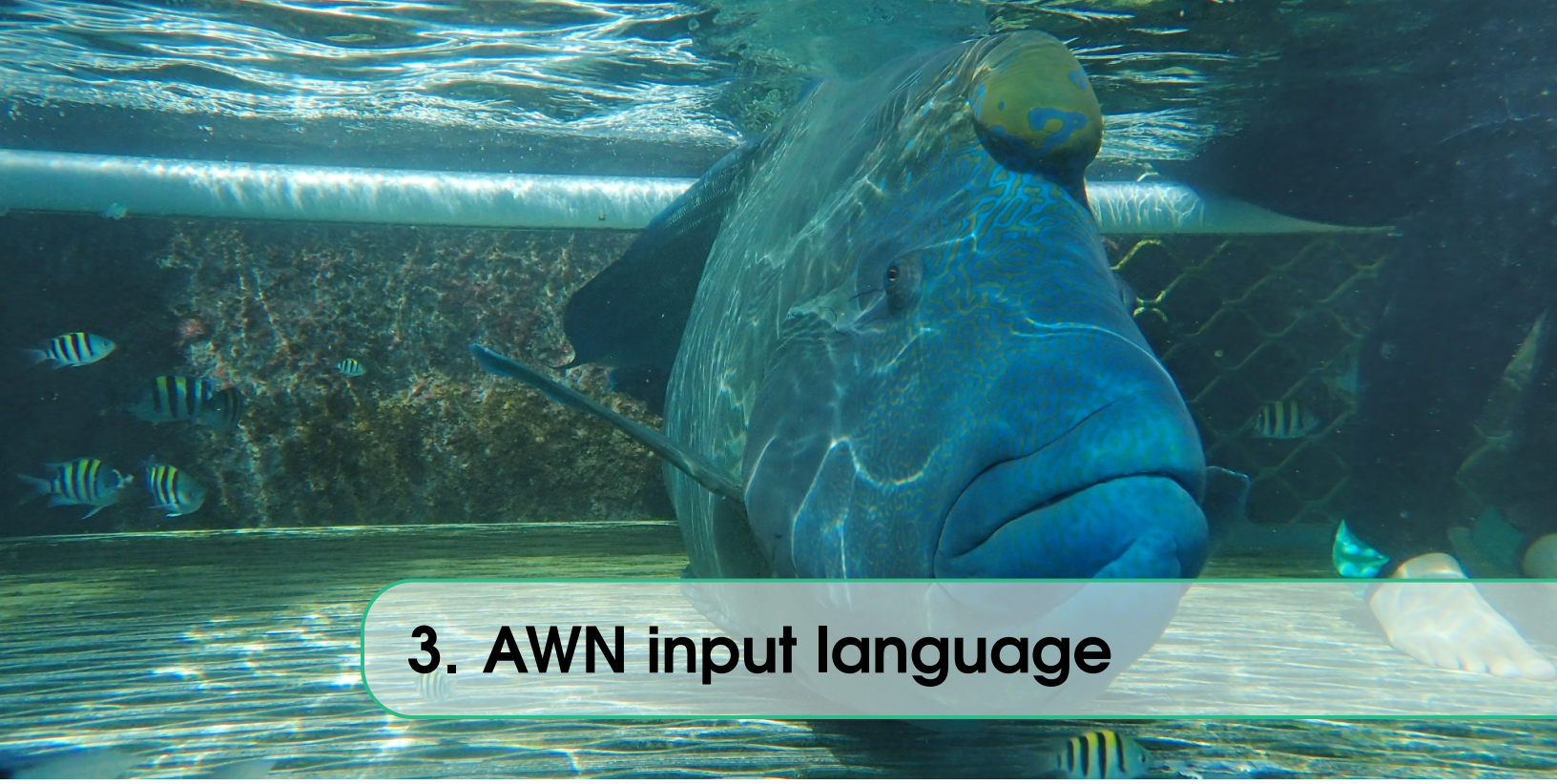
where

$$\mathfrak{f}(a) = \begin{cases} \tau & \text{if } a = \tau \\ \mathbf{deliver}(\mathrm{U}(ip), \mathrm{U}(d)) & \text{if } a = ip : \mathbf{deliver}(d) \\ \mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip")) & \text{if } a = \mathbf{connect}(ip', ip") \\ \mathbf{disconnect}(\mathrm{U}(ip'), \mathrm{U}(ip")) & \text{if } a = \mathbf{disconnect}(ip', ip") \\ \mathbf{newpkt}(\{\mathrm{U}(ip)\}, \{\mathrm{U}(ip)\}, \mathtt{newpkt}(\mathrm{U}(d), \mathrm{U}(dip))) & \text{if } a = ip : \mathbf{newpkt}(d, dip) \end{cases}$$

from which can be concluded immediately that

$$\left\{ ([\,\mathrm{M}\,], \tau_{\{\mathbf{t}\}}\mathrm{T}([\,\mathrm{M}\,])) \;\middle|\; [\,\mathrm{M}\,] \text{ is an AWN network expression} \right\}$$

is a strong bisimulation up to $\equiv$ modulo renaming by $\mathfrak{f}$. ■



3. AWN input language

The formal translation has been implemented in practice in the form of a tool. This tool requires an AWN specification as input. This input is written in the *AWN input language*, which is described in this section.

1 Files

The AWN input language allows an AWN specification to be distributed over multiple files. One of these files must be a *protocol file* and the other files must be *library files*.

1.1 Header

The type of a file is determined by its header. Protocol files start with

protocol *protocolName* ;

In the current version of the AWN input language, a protocol header only gives a name to the specified protocol and indicates the starting point of an AWN specification. In the future, however, the protocol header could be extended with syntax to make certain assumptions made by the protocol explicit (one example of such an assumption is whether connections in a network are *symmetric* or *asymmetric*).

Library files start with the header

library *libraryName* ;

By providing a name, the library file allows other files to access its definitions. This is done by *importing* the library file with the following syntax:

import *libraryName* ;

Imports are recursive; that is, if a library file B imports a library file C, then a file A that imports B gains access to the definitions in C as well.

1.2 Body

After the file header, an AWN file can contain a number of consecutive *declarations*. Declarations link a name to an object – which can be a type, a constant, a function, a process, or a network – so that the name will be interpreted accordingly elsewhere in the specification The following section and Sections 4 to 8 elaborate on the available declaration categories.

2 Type declarations

The AWN specification will almost certainly make use of data, and the AWN input language requires information about the *type* of that data. Some types are immediately available (see subsection 2.1) whereas other types must be declared before they can be used.

Type declarations follow this grammar:

type *typeName* (ε | `=` *typeExpr*);

Type declarations do not have to make the actual type explicit in the case that the AWN specification should not make any assumptions about the type. Otherwise, a type declaration refers to a *type expression*, which adheres to the grammar

typeExpr ::= *typeName*
| (*typeExpr*)
| `Boolean` | `Integer` | `String` | `$IP` | `$MSG` | `$DATA` | `$STRUCT` | `$TRACE`
| **enum**(*enumValue*$_1$, $\cdots$, *enumValue*$_n$)
| **range**(*dataExpr*, *dataExpr*)
| **list of** *typeExpr*
| **set of** *typeExpr*
| **struct**(*fieldName*: *typeExpr*, $\cdots$, *fieldName*: *typeExpr*) (ε | **extends** *typeExpr*)
| *typeExpr* `#` $\cdots$ `#` *typeExpr* `->` *typeExpr*
| *typeExpr* `#` $\cdots$ `#` *typeExpr* `+->` *typeExpr*

The grammar allows the declaration of primitive types, enumerable types, range types, list types, set types, struct types, and function types (which can be total or partial). Each of these type categories is discussed in the following subsections.

2.1 Primitive types

In each specification, the AWN input language provides the following types automatically:

- A boolean type named `Boolean` (`true` or `false`);
- An integer type named `Integer` (`...`, `-2`, `-1`, `0`, `1`, `2`, `3`, `...`);
- A floating point type named `Real` (`3.14159`, `...`);
- A struct type named `$IP`, which is the type of node addresses;
- A struct type named `$MSG`, the type used for storing messages;
- A struct type named `$DATA`, the type used for storing the data that travels through a network;
- A struct type named `$STRUCT`, which can be used for generic data structures;
- A struct type named `$TRACE`, which can be passed to **trace** actions for model-checking purposes (see Section 6).

2.2 Enumerable types

Enumerable types define a static set of values, and instances of a specific enumerable type can only have values that are in the set that corresponds with their type.

2.3 Range types

Range types are integer types restricted to a static range. For example,

```
1  type Byte = range(0, 255);
```

declares an integer named `Byte` restricted to values between `0` and `255`.

2.4 List types

List types are collections of elements of a specified type. For example,

```
1  type Array = list of Integer;
```

declares a list of integers. Elements in a list are ordered – they have a position relative to the head of the list – and do not have to be unique. Lists can only contain a finite number of elements their contents (elements are maintained exhaustively).

2.5 Set types

Similar to list types, set types are collections of elements of a specified type. Unlike lists, however, sets do not maintain the positions of their elements, and they can also denote an infinite number of elements by using a symbolic representation.

2.6 Struct types

Struct types (also called *hierarchical* types) are types that can:

- Define fields, which are named containers of a specific type;
- Inherit fields from another hierarchical type.

For example,

```
1  type Entry = struct(key: Integer, value: Integer);
```

defines an struct type `Entry` with two fields, one named `key` and one named `value`, that both are of type `Integer`. The supertype of `Entry` is left out, which means that the default supertype is used, `$STRUCT`. `$STRUCT` is a primitive struct type that, like the other primitive struct types, has no fields, and consequently `Entry` does not have any inherited fields.

The `Entry` type can, in turn, also be used by another struct type as a supertype:

```
1  type MarkedEntry = struct(marked: boolean) extends Entry;
```

The `MarkedEntry` type has 3 fields: it inherits `key` and `value` from `Entry`, and then there is `marked`, which it added itself.

Struct types that inherit from `$STRUCT` are meant for defining general data structures. Types that inherit from `$IP`, `$MSG`, `$DATA`, or `$TRACE` exist for more specific purposes, namely for identifying nodes, storing messages, storing the data that passes through a network, and making protocol behavior visible, respectively. To illustrate, a typical extension of `$IP` is

For example,

```
1  type IP = struct(address: Integer) extends $IP; //$IP refers to a primitive type.
```

which allows the type `IP` to be used for the identification of nodes.

2.7 Function types

Function types pair one or more *source types* with exactly one *target type*. By providing function types, the AWN input language makes it possible to pass functions as a parameter to other functions, which greatly increases the expressiveness of the language.

Functions can be total or partial depending on whether they are defined for all possible input values or not, and a function's totalness is reflected in its type: total functions use the regular `->` arrow whereas partial functions use `+->`.

3 Data expressions

Data expressions allow for the analysis and manipulation of data. They conform to the following grammar:

dataExpr ::= *booleanValue* | *integerValue* | *realValue*
| *enumTypeName*`::`*enumValue*
| *variableName*
| *functionName*(*dataExpr*, ⋯ ,*dataExpr*)
| *variableName*(*dataExpr*, ⋯ ,*dataExpr*)
| *typeName*(*dataExpr*)
| **list of** *typeExpr* (*dataExpr*) | **set of** *typeExpr* (*dataExpr*)
| *unaryOp dataExpr*
| *dataExpr binaryOp dataExpr*
| *dataExpr* `.` *fieldName*
| *dataExpr*`[`*dataExpr*`]`
| *dataExpr* **is** *typeExpr* | *dataExpr* **istype** *typeExpr*
| (*dataExpr*)
| *dataExpr* `#` ⋯ `#` *dataExpr* `->` *dataExpr*
| `|`*dataExpr*`|`
| **low**(*typeExpr*) | **high**(*typeExpr*)
| **head**(*dataExpr*) | **tail**(*dataExpr*) | **rhead**(*dataExpr*) | **rtail**(*dataExpr*)
| **floor**(*dataExpr*) | **ceil**(*dataExpr*) | **round**(*dataExpr*)
| **collapse**(*dataExpr*)
| **lambda** *varDecls* `.` *dataExpr*
| { *dataExpr* `..` *dataExpr* } (ε | **of** *typeExpr*)
| { *dataExpr*, ⋯ ,*dataExpr* } (ε | **of** *typeExpr*)
| { *dataExpr* (ε | *quantDecls* (ε | `@` *dataExpr*)) } (ε | **of** *typeExpr*)
| [*dataExpr* `..` *dataExpr*] (ε | **of** *typeExpr*)
| [*dataExpr*, ⋯ ,*dataExpr*] (ε | **of** *typeExpr*)
| [*dataExpr* (ε | *quantDecls* (ε | `@` *dataExpr*))] (ε | **of** *typeExpr*)
| **new** *typeName*(*dataExpr*, ⋯ ,*dataExpr*)
| **if** *dataExpr* **then** *dataExpr* **else** *dataExpr* **end**
| **forall**(*quantDecls* `@` *dataExpr*) | **exists**(*quantDecls* `@` *dataExpr*)
| **ifexists** *quantDecls* `@` *dataExpr* **then** *dataExpr* **else** *dataExpr* **end**
| **with** *assignExprs* **do** *dataExpr* **end**
| **with init** *variableName* `:=` *dataExpr*, *quantDecls* **do** *dataExpr* **end**
| **undefined** *typeExpr* | **arbitrary** *typeExpr*

varDecls ::= *variableName*: *typeExpr*, ⋯ , *variableName*: *typeExpr*

quantDecls ::= *variableName* **in** *dataExpr*, ⋯ , *variableName* **in** *dataExpr*

assignExprs ::= *variableName* `:=` *dataExpr*, ⋯ , *variableName* `:=` *dataExpr*

The following subsections describe each of these types of data expressions.

3.1 Literals

As in many programming languages, data expressions may consist of simple literal values. In case of the AWN input language, there are boolean literals, integer literals, and real literals:

$$\textit{dataExpr} ::= \textit{booleanValue} \mid \textit{integerValue} \mid \textit{realValue} \mid \textit{enumTypeName}::\textit{enumValue}$$

In addition, the values of the declared enumerable types can also be considered literals. Note that the name of the enumerable type and the $::$ symbol are obligatory!

3.2 Variables

Data expressions can refer to the values stored in variables by writing the names of those variables:

$$\textit{dataExpr} ::= \textit{variableName}$$

3.3 Function calls

Functions (see Section 5) can be called by writing the function name with the appropriate number of parameter values in parentheses, and if a function is stored in a variable the same can be done by writing the name of that variable instead:

$$\begin{aligned}\textit{dataExpr} ::= {} & \textit{functionName}(\textit{dataExpr}, \cdots, \textit{dataExpr}) \\ \mid {} & \textit{variableName}(\textit{dataExpr}, \cdots, \textit{dataExpr})\end{aligned}$$

3.4 Casting

A special type of function consists of the name of a type followed by an expression in parentheses:

$$\begin{aligned}\textit{dataExpr} ::= {} & \textit{typeName}(\textit{dataExpr}) \\ \mid {} & \textbf{list of } \textit{typeExpr}\ (\textit{dataExpr}) \\ \mid {} & \textbf{set of } \textit{typeExpr}\ (\textit{dataExpr})\end{aligned}$$

This is a *casting operation*: the expression in parentheses will be interpreted as if it were of the type in front of the parentheses. The same can be done without using a type name in case of lists and sets – this syntax is useful in particular for casting the element type of a collection.

Note that it is good practice (but not obligatory) to precede a cast by an **is** or **istype** operation (see subsection 3.5) in order to check at run-time whether the cast is legal.

3.5 Operations

Data expressions can, of course, consist of unary or binary operations (a list of all available unary and binary operations can be found in Appendix F):

dataExpr ::= *unaryOp dataExpr*
| *dataExpr binaryOp dataExpr*
| *dataExpr* . *fieldName*
| *dataExpr*`[`*dataExpr*`]`
| *dataExpr* **is** *typeExpr*
| *dataExpr* **istype** *typeExpr*
| (*dataExpr*)

The following binary operations in particular are special:

- The . operator followed by the name of a struct field is used to access the value stored in the field by that name in the preceding struct;
- `[`n`]` accesses the $n+1$-th element of the preceding list (n must be of type `Integer`);
- The **is** operator checks whether the object left of the operator is of the type right of the operator or a subtype;
- The **istype** operator checks whether the object left of the operator is exactly of the type right of the operator;

In order to override operator precedence, place the appropriate data expression between parentheses in the traditional manner.

3.6 Partial function construction

The `->` operator can be used to construct an instance of a partial function:

dataExpr ::= *dataExpr* `#` ⋯ `#` *dataExpr* `->` *dataExpr*

3.7 Predefined functions

The AWN input language provides a number of predefined functions:

Function	*Description*
|*dataExpr*|	Returns the absolute value of an integer or real, or the size of a list.
low(*typeExpr*)	Returns the first value of an enumerable type, or the lower bound of a range type.
high(*typeExpr*)	Returns the last value of an enumerable type, or the upper bound of a range type.
head(*dataExpr*)	Returns the first element of a list.
tail(*dataExpr*)	Returns a list containing all elements of a list *after* its first element.
rhead(*dataExpr*)	Return the last element of a list.
rtail(*dataExpr*)	Returns a list containing all elements of a list *before* its last element.
floor(*dataExpr*)	Returns the greatest integer smaller than or equal to a real.
ceil(*dataExpr*)	Returns the smallest integer greater than or equal to a real.
round(*dataExpr*)	Returns a real rounded to an integer.
collapse(*dataExpr*)	Remove all duplicates from a list.

3.8 Lambda functions

Lambda functions are anonymous functions that are specified inside data expressions themselves:

$$\begin{aligned} \textit{dataExpr} &::= \textbf{lambda}\ \textit{varDecls}\ .\ \textit{dataExpr} \\ \textit{varDecls} &::= \textit{variableName}\colon\ \textit{dataExpr},\ \cdots,\ \textit{variableName}\colon\ \textit{dataExpr} \end{aligned}$$

3.9 Collection expressions

Both sets and lists can be constructed manually by using the syntax below:

$$\begin{aligned} \textit{dataExpr} ::= {} & \{\ \textit{dataExpr}\ ..\ \textit{dataExpr}\ \}\ (\varepsilon \mid \textbf{of}\ \textit{typeExpr}) \\ \mid {} & \{\ \textit{dataExpr}, \cdots, \textit{dataExpr}\ \}\ (\varepsilon \mid \textbf{of}\ \textit{typeExpr}) \\ \mid {} & \{\ \textit{dataExpr}\ (\varepsilon \mid \textit{quantDecls}\ (\varepsilon \mid \texttt{@}\ \textit{dataExpr}\ )\ )\ \}\ (\varepsilon \mid \textbf{of}\ \textit{typeExpr}) \\ \mid {} & [\ \textit{dataExpr}\ ..\ \textit{dataExpr}\ ]\ (\varepsilon \mid \textbf{of}\ \textit{typeExpr}) \\ \mid {} & [\ \textit{dataExpr}, \cdots, \textit{dataExpr}\ ]\ (\varepsilon \mid \textbf{of}\ \textit{typeExpr}) \\ \mid {} & [\ \textit{dataExpr}\ (\varepsilon \mid \textit{quantDecls}\ (\varepsilon \mid \texttt{@}\ \textit{dataExpr}\ )\ )\ ]\ (\varepsilon \mid \textbf{of}\ \textit{typeExpr}) \end{aligned}$$

The first notation allows a set to be specified by defining a lower and upper bound. Obviously, this only works in the case of certain types (integers, reals, and enumerable types). The second notation specifies set elements one by one, and the third notation uses set comprehension to specify the contents of the set.

The notations for list expressions are similar, with one restriction: the quantifiers in the last notation *must* be lists – after all, it is not generally possible to convert a symbolic expression to an exhaustive list of elements!

3.10 Struct construction

Struct types are instantiated by using the **new** keyword:

$$\textit{dataExpr} ::= \textbf{new}\ \textit{typeName}(\textit{dataExpr}, \cdots, \textit{dataExpr})$$

The required parameters provide initial values for all fields of the struct, including inherited fields (the fields are ordered from oldest to newest). The number of parameters and the types of the parameters should, of course, be compatible with the types of those fields.

3.11 Conditional expression

The following expression allows one of two branches to be selected based on the evaluation of a condition:

$$\textit{dataExpr} ::= \textbf{if}\ \textit{dataExpr}\ \textbf{then}\ \textit{dataExpr}\ \textbf{else}\ \textit{dataExpr}\ \textbf{end}$$

The types of both branches must be compatible. If the condition is undefined, the result of the conditional expression will also be undefined.

3.12 Quantified expressions

The application of the ∀ and ∃ operators have the following notations in the AWN input language:

$$\begin{aligned} \textit{dataExpr} &::= \textbf{forall}(\textit{quantDecls}\ \texttt{@}\ \textit{dataExpr}) \\ &\mid\ \textbf{exists}(\textit{quantDecls}\ \texttt{@}\ \textit{dataExpr}) \\ \textit{quantDecls} &::= \textit{variableName}\ \textbf{in}\ \textit{dataExpr}, \cdots, \textit{variableName}\ \textbf{in}\ \textit{dataExpr} \end{aligned}$$

3.13 Ifexists expression

The following syntax allows an ∃ operator to be executed, and then one of two branches to be selected depending on whether the ∃-condition could be satisfied:

$$\begin{aligned} \textit{dataExpr} &::= \textbf{ifexists}\ \textit{quantDecls}\ \texttt{@}\ \textit{dataExpr}\ \textbf{then}\ \textit{dataExpr}\ \textbf{else}\ \textit{dataExpr}\ \textbf{end} \\ \textit{quantDecls} &::= \textit{variableName}\ \textbf{in}\ \textit{dataExpr}, \cdots, \textit{variableName}\ \textbf{in}\ \textit{dataExpr} \end{aligned}$$

In the positive branch, the quantifiers have the values of an arbitrary instance for which the ∃-condition holds. The negative branch can be used to set a default (or undefined) value.
The code snippet below demonstrates a function that returns the greatest integer in a list

```
1  function max(ints: list of Integer): Integer
2   = ifexists e1 in list @ forall(e2 in ints @ e1 >= e2) then e1 else 0 end;
```

3.14 With expression

With a **with** expression, a value can be computed and stored in a variable for later use:

$$\begin{aligned} \textit{dataExpr} &::= \textbf{with}\ \textit{assignExprs}\ \textbf{do}\ \textit{dataExpr}\ \textbf{end} \\ \textit{assignExprs} &::= \textit{variableName}\ \texttt{:=}\ \textit{dataExpr}, \cdots, \textit{variableName}\ \texttt{:=}\ \textit{dataExpr} \end{aligned}$$

The following example illustrates how the **with** expression can be used in practice:

```
1  function pythagoras(x1, x2, y1, y2: Integer): Real
2    = with dx := x2 - x1, dy := y2 - y1 do
3        sqrt(dx * dx + dy * dy)
4      end;
```

(The example above assumes the existence of the **sqrt** function, which is not a predefined function in the AWN input language.)

3.15 With-init expression

The **with-init** expression can be applied to lists in order to compute a value based on their contents:

dataExpr ::= **with init** *variableName* := *dataExpr*, *quantDecls* **do** *dataExpr* **end**
quantDecls ::= *variableName* **in** *dataExpr*, ⋯ , *variableName* **in** *dataExpr*

As an example, the following code snippet shows how **with init** is used to count the number of non-negative integers in a list:

```
1  function getPosIntCount(ints: list of Integer): Integer
2   = with init c := 0, i in ints do
3       if i < 0 then c else c + 1 end
4     end;
```

The variable `c` is initialized to `0`, after which the data expression in the body is evaluated for every element `i` in the list `ints`: for every `i >= 0`, `c` is incremented, otherwise it remains the same.
Elements of a list are processed in order of their position in the list. When using multiple quantifiers, the data expression in the body is evaluated for every combination of values of those quantifiers, with each quantifier iterating once for every combination of values of its preceding quantifiers.

3.16 Undefined expression

If a data expression is not defined under certain circumstances, the following syntax can be used to denote this:

dataExpr ::= **undefined** *typeExpr*

For example,

```
1  partial function safeDivide(value, divisor: Integer): Integer
2   = if divisor == 0 then
3       undefined Integer
4     else
5       value / divisor
6     end;
```

prevents divisions by zero from taking place, returning an undefined value instead. The function must be defined as *partial* (see Section 5) as it is not defined for certain input values.
Operations with undefined operands and functions with undefined parameters yield an undefined value of their return type. This means that `undefined Integer == 12` evaluates to `undefined Boolean` and that `safeDivide(12, 0) + 1.0` evaluates to `undefined Real`.

3.17 Arbitrary expression

For model-checking purposes, it is sometimes useful to be able to give a non-specific literal value. This does *not* mean that the literal has a random value, but that the AWN specification will be considered for all possible values of the literal:

dataExpr ::= **arbitrary** *typeExpr*

4 Constants

The AWN input language provides the option to define *constants*. Constants are global identifiers with a type and a value that cannot be modified after initialization. The grammar for constants is as follows:

$$\textit{constDecl} \ ::= \ \textbf{const}\ \textit{variableName}\colon \textit{typeExpr};$$

A simple example of a constant declaration is

```
1  const PI: Real = 3.14159;
```

5 Functions

Functions allow computations to be expressed more efficiently. To define a function, use the following grammar:

$$\begin{aligned}\textit{function} \ &::= \ (\varepsilon \mid \textbf{partial})\ \textbf{function}\ \textit{functionName}\ (\ \textit{varDecls}\ )\ :\ \textit{typeExpr} = \textit{dataExpr}\ ;\\ \textit{varDecls} \ &::= \ \textit{variableName}\colon \textit{dataExpr}, \cdots, \textit{variableName}\colon \textit{dataExpr}\end{aligned}$$

Clearly, the distinction between a partial and total function is made with the **partial** keyword. A sequence of variable declarations (or rather, parameter declarations) defines which identifiers refer to the parameter values of the function and which types the type of those parameters. The function should also specify its return type. Finally, the function must give its body as a data expression.

6 Sequential processes

Sequential processes make use of the following grammar:

seqProcessDecl ::=
(ε | **sequential**) **process** *seqProcessName*(*varDecls*) : *typeExpr*
(ε | **uses** *varDecls*) = *seqProcessExpr*;

Parameters allow information to be passed to a sequential process when it is instantiated. Sequential processes also have an optional **uses** clause for variables that are initialized by guards (this reduces the risk of new variables being introduced by accident).
The body of a sequential process adheres to this grammar:

seqProcessExpr ::= **broadcast**(*dataExpr*) . *seqProcessExpr*
| **groupcast**(*dataExpr*, *dataExpr*) . *seqProcessExpr*
| **unicast**(*dataExpr*, *dataExpr*) . *seqProcessExpr* > (*seqProcessExpr* | ...)
| **send**(*dataExpr*) . *seqProcessExpr*
| **deliver**(*dataExpr*) . *seqProcessExpr*
| **receive**(*variableName*) . *seqProcessExpr*
| [[*dataExpr*]] *seqProcessExpr*
| *seqProcessName*(*dataExpr*, ⋯, *dataExpr*)
| *seqProcessExpr* + *seqProcessExpr*
| [*dataExpr*] *seqProcessExpr*
| (*seqProcessExpr*)
| **if** *dataExpr* **then** *seqProcessExpr* **else** *seqProcessExpr* **end**
| **trace**(*dataExpr*) . *seqProcessExpr*

The grammar only introduces a small number of new notations compared to the behavior of AWN:

- The **unicast** action now can use ... instead of a process expression as its 'else' branch, which denotes that its 'else' branch is the same as its 'then' branch.
- Instead of the [_] notation for guards, it is now possible to use an **if** expression. The main advantage of the **if** expression is the presence of an **else** branch. (Both branches produce a τ-transition when taken.)
- The **trace** action is introduced in order to facilitate model-checking. The **trace** action takes a parameter of type $TRACE (or a subtype of $TRACE) which can carry interesting data. Because **trace** actions are visible actions from outside the protocol, the data will also be visible.

The code snippet below shows an example of a sequential process that forwards a message to its destination:

```
1  process P uses m: $MSG, dest: $IP, text: String = receive(m) .
2    [DataMsg(m) == new DataMsg(dest, text)] unicast(dest, m) . P > ...;
```

7 Parallel processes

Parallel processes use the following syntax:

parProcessDecl ::=
parallel process *parProcessName*(*varDecls*) : *typeExpr*
(*ε* | **uses** *varDecls*) = *parProcessExpr*;

The body of a parallel process makes use of this grammar:

parProcessExpr ::= *seqProcessExpr* | *parProcessExpr* << *parProcessExpr*

A parallel process expression can put a number of sequential processes in parallel, exactly as specified by the formal AWN language.

8 Networks

Networks adhere to the following grammar:

networkDecl ::= **network** *networkName* = *networkExpr*;
networkExpr ::= *networkExpr* || *networkExpr* | *nodeExpr*
nodeExpr ::= *dataExpr*:*parProcessExpr*:*dataExpr*
| *dataExpr*:*parProcessName*(*dataExpr*, ···, *dataExpr*):*dataExpr*

The building blocks of a network are *node expressions*. These expressions start with the address of the node (which must be of type `$IP` or a subtype of `$IP`) followed by a colon and then a parallel process expression *or* the instantiation of a parallel process (see subsection 7) followed by a second colon and then a set of the addresses of the nodes that are within range of the current node.
To illustrate, the following code snippet defines a fully connected network of 3 nodes:

```
1  network ThreeNodes =
2        new IP(1) : Node() : { new IP(2), new IP(3) }
3     || new IP(2) : Node() : { new IP(1), new IP(3) }
4     || new IP(3) : Node() : { new IP(1), new IP(2) };
```

4. Implementation

The translation from AWN to mCRL2 has been created inside of a newly developed framework that supports future translations from AWN to other model-checkers and code generators. The implementation of this framework and of the AWN-to-mCRL2 translation is described here.

1 Translation framework

The framework, which has mostly been developed in Java but also makes use of several domain-specific languages, can be used in two ways:

- As standalone software; or
- As a plug-in of the popular coding environment Eclipse[1].

Both cases make use of the same underlying code.

Figure 4.1 gives a high-level overview of how different parts of the implementation relate. The core of the implementation is the so-called 'transformation chain executor'. This component can execute a series of instructions contained in a given *transformation chain specification*, such as loading an AWN model from a text file (that is, compiling an AWN specification), transforming models (translating AWN to a different format), and exporting a model as text.

In order to compile an AWN specification, the transformation chain executor uses the AWN compiler. This compiler is generated from an Xtext grammar. The compiler also has several modules for which Xtext only generates stubs; they must be implemented further manually if needed. The important modules are the scope provider, which determines the objects that are accessible from a specific position in an AWN specification, and the validator, which checks for problems such as type mismatches.

The Eclipse plug-in contains functionality that redirects the files of the selected project to the same code as the standalone implementation. However, it also makes use of the scope provider and

[1] `https://www.eclipse.org/`

validator in a more direct manner, namely by extracting suggestions for code completion from them and by showing the user immediately whether there are errors in the AWN specification. There are also Xtext modules that are only used by user-interfaces, such as a module that generates a layout of the code that can be used for navigation and a module that shows more information on an object when hovering over it with the mouse. Some of these modules have been implemented to a limited degree.

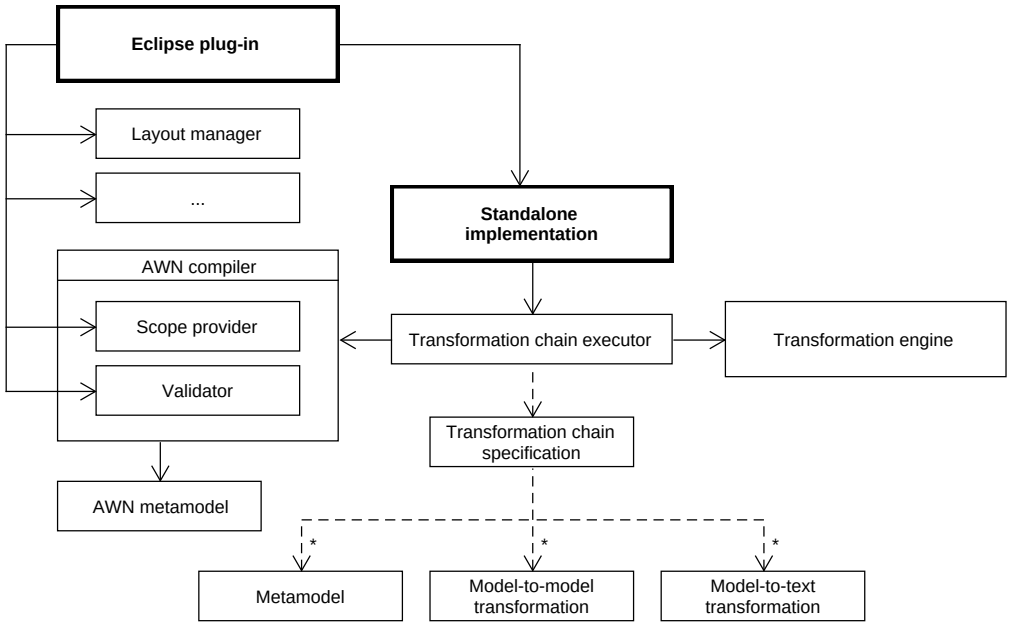


Figure 4.1: Chain of transformations used by the implementation of the AWN-to-mCRL2 translation.

2 AWN-to-mCRL2 translation

The translation framework currently contains a single translation chain specification, namely the one from AWN to mCRL2. The specification can be found in the file `awn2mcrl2.chain`. It executes 4 main instructions:

1. First, the AWN specification is parsed and validated, resulting in an AWN model if successful. This model is still in it 'raw' form – this means that it is very similar to an abstract syntax tree, and that more work is needed to make manipulation of the AWN specification more convenient.
2. The AWN model is reorganized and decorated with more information, which gives a more forthcoming AWN model.
3. The third transformation constructs an mCRL2 model based on the new AWN model; this transformation is therefore that which actually performs the AWN-to-mCRL2 translation.
4. The fourth instruction is to export the mCRL2 model to one or more `.mcrl2` files.

Other translations (a translation from AWN to UPPAAL, for example) may require additional intermediate steps or no intermediate steps at all.

The following sections describe each of the main instructions in greater detail. A graphical representation of the AWN-to-mCRL2 transformation chain can be found in Figure 4.2.

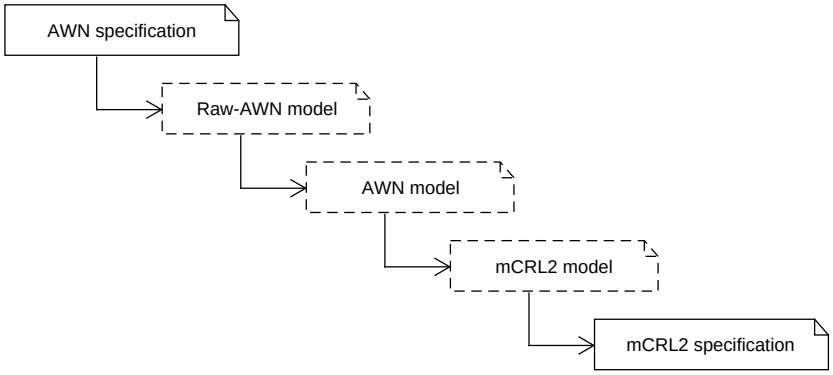


Figure 4.2: Simplified transformation chain for the AWN-to-mCRL2 translation.

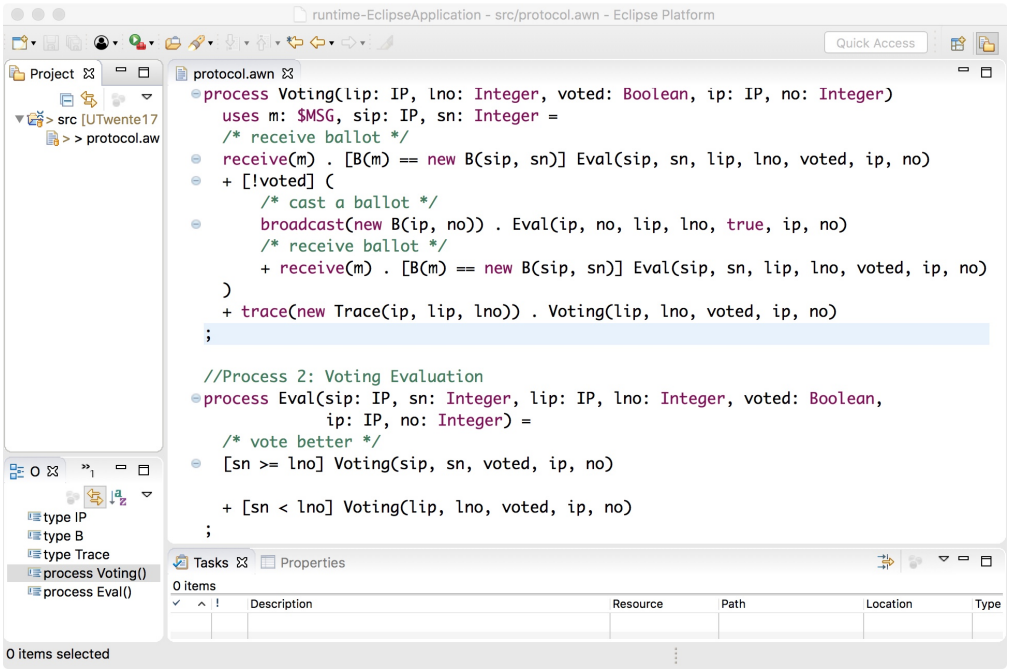


Figure 4.3: Screen capture of the graphical user interface of the Eclipse plug-in.

2.1 Compiling AWN

The first transformation in the transformation chain is a compilation of plain-text files: the AWN specification in text form is converted by a lexer to a token stream, which is subsequently used by a parser to construct a 'raw' AWN model instance (as if it were an abstract syntax tree). Finally, the 'raw' AWN model is checked for problems by a validator. The transformation chain is aborted if problems are found.

The lexer and parser mentioned above are all generated from a language specification by Xtext, a framework for developing domain-specific languages. The grammar used by the parser can be found across Section 3, where the syntax of AWN (as developed in the course of this project) is discussed.

The validator is an extension of a stub generated by Xtext. It checks the validity of declarations on the top level of the abstract syntax tree (such as type declarations, function declarations, and process declarations) by recursively exploring the corresponding branch in the abstract syntax tree. The validator ignores problems at a higher level if problems at a lower level are found – otherwise the user would be bombarded with error messages that are mostly irrelevant!

Several (Java) helper classes have been written to assist with the validation. These classes mostly contain functionality to determine the type of expressions and to analyze the relationships between types.

Xtext automatically links the lexer, parser, and validator to the compiler. Besides these bare necessities, Xtext also provides components for an integrated development environment (IDE), such as a scope provider (which dictates which objects are accessible at a certain position in the AWN specification) and a content assistant (which helps the user with modifying code). Some of these components have been implemented in order to make the Xtext-generated IDE for AWN suitable for more serious use.

2.2 Transformation from Raw-AWN to AWN

The 'raw' AWN model produced by the AWN compiler has many characteristics of an abstract syntax tree. Before doing the transformation to an mCRL2 model, it is convenient to do a preparatory step that rearranges and restructures parts of the 'raw' model first.

This step is performed by another transformation. This transformation is contained in `raw2awn.qvto`, where the extension '.qvto' refers to the programming language in which transformation has been implemented: 'QVT' stands for Query/View/Transformation, which are three concepts that are integrated in the language, and the 'o' stands for operational, indicating that certain instructions in the language are executed imperatively.

The `raw2awn.qvto` transformation handles the following tasks:

- It introduces actual objects for primitive types, and it changes references to primitive types to references to those objects. Consequently, subsequent transformations can handle primitive type references as regular type references.
- The transformation modifies object names if necessary in order to make them unique. This is done in order to preempt naming conflicts.
- Certain elements in the 'raw' model represent multiple object types. The reason for this is that there exists overlap between the syntax for those different object types, and Xtext does not have the capability of differentiating between the object types at parsing time.

As a prime example, elements of type `NamedDefinitionObjectRef` are used for variable references, values of enumerable types, function calls, and casting operations. The transformation of the 'raw' model determines which object type was meant and replaces the original element by one that more accurately represents that object type.

- Some operators for data expressions have ambiguous uses: the `+` can be used to add up two numbers, take the union of two sets, or concatenate two lists; the `-` operator can be used to subtract a number, set, or list from another number, set, or list; the `!` operator can apply logical negation to a boolean expression or take the complement of a set; and the `|`*expr*`|` function either returns the absolute value of a number or the size of a list.
  Because of the ambiguity, the Xtext parser makes a default choice, letting the transformation of the 'raw' AWN model resolve the ambiguity by potentially overwriting the operation type based on operand types.
- The transformation fills in missing fields resulting from optional syntax (so that subsequent transformations do not have to know how to handle these cases).
- The transformation resolves syntactic sugar (such as the syntax of **ifexists** expressions, which are actually refinements of **with-init** expressions).
- The transformation ensures that all types used in the AWN specification have an explicitly named declaration in the output model. This includes the types of functions and expressions. This results in a list of type declarations that is often longer than in the original 'raw' model. Subsequent transformations can refer to this list in order to more easily identify types and their relations with other types.

2.3 Transformation from AWN to mCRL2

This part of the implementation has been, of course, the focus of the entire project: it is the transformation after which AWN has disappeared and an mCRL2 specification has taken over. This transformation is contained entirely in the file `awn2mcrl2.qvto`.

The `awn2mcrl2.qvto` transformation must perform multiple tasks, and applying the translation function from Section 3 is one of the least laborious.

Broadly, the steps that the transformation goes through are as follows:

1. It defines a list of keywords from both AWN and mCRL2. When encountering these keywords as names of variables or functions at a later stage of the transformation they will be renamed in order to prevent naming conflicts.
2. It makes a list for AWN types and a list of mCRL2 sorts – the idea being that the mCRL2 sort at a specific position in the second list is the counterpart to an AWN type at a specific position in the first list. The first entries added to the lists are the AWN primitive types `Boolean`, `Integer`, `Real`, `$IP`, `$MSG`, `$DATA`, `$STRUCT`, and `$TRACE`. The mCRL2 counterparts are constructed manually.
3. The transformation declares the mCRL2 actions that the translation requires.
4. The transformation declares an mCRL2 sort for each AWN type in the input model, but *without content* for the moment: the sort declaration instances are needed from the very beginning so that they can be referenced before their translation has been performed. The AWN types and their mCRL2 counterparts are added to the two lists mentioned in step 2. Because the previous transformation has ensured that all AWN types are present in the input

model, it is now possible to find the corresponding mCRL2 sort declaration of each AWN type that the transformation can encounter.

5. Now that all mCRL2 sorts can be referenced, the transformation does the actual translation of the AWN types. Most translations are straightforward: set types are translated to set sorts, list types are translated to list sorts, function types are translated to function sorts, and so on. The exception is the translation of struct types, because all struct types that are a subtype of the same primitive struct type (`$IP`, `$MSG`, `$DATA`, `$STRUCT`, or `$TRACE`) are converted to the same, manually created mCRL2 structured sort. For each struct type, the transformation determines its ultimate supertype, and then adds all fields of the struct type as arguments to the appropriate structured sort.
   To be more precise, the mCRL2 structured sort has two constructors: one default 'nil' constructor and one 'new' constructor. It is the second constructor that is modified at this point of the transformation:
   - The value of the first argument of the constructor indicates the struct type in the original AWN specification. The operators **is** and **istype** compare this value with their second operand.
   - The subsequent arguments consist of all fields of all struct types that inherit from the primitive struct type in question. (Note that this will likely cause instances of the structured sort to have many arguments with irrelevant, dummy values.)
6. The constants from the AWN specification are translated.
7. The functions from the AWN specification are translated.
8. The sequential processes from the AWN specification are translated.
9. The parallel processes from the AWN specification are translated.
10. The transformation adds the helper processes G and H (see T14 and T16 in Table 2.9) to the specification.
11. The network declarations from the AWN specification are translated.

Ultimately, the majority of the `awn2mcrl2.qvto` file consists of code that translates data expressions from AWN to mCRL2. This is not surprising considering the number of different expression types.

2.4 mCRL2 to text

The model-to-text part of the translation was implemented with the following features:

- Model-to-text specifications are compiled at run-time. This has the advantage that users can add or modify a translation without having to rebuild the AWN plug-in for Eclipse.
- Names and locations of the output files are specified in the current `.chain` file. This means that all information related to input and output can be found in that `.chain` file.

The model-to-text translation for mCRL2 was defined in a metamodel-independent language developed specifically for this project, namely `TxtGen` (see Appendix I). Not making use of an existing tool deviates from the MDE principle to reuse previous work as much as possible; however, achieving the features listed above by using available tools such as Xpand and Acceleo required more effort than expected, and so a custom alternative was developed. In the future, it may be preferable to replace the mCRL2-to-text translation with one that is based on an existing tool.

5. Use cases

Here two use cases that were performed in the course of the project are discussed.

1 Leader protocol

The protocol chosen of the first use case is the leader protocol, chosen because it is small, making manual inspection – if required – manageable. The purpose of the leader protocol is to allow the nodes of a fully connected network to decide on a leader node. To this end, each node is initialized with a number, which it will eventually transmit to all other nodes. Nodes keep track of which node transmitted the highest number, and the idea is that when all nodes have transmitted their number all nodes have selected the same leader.

The leader protocol was defined using the AWN input language (see Appendix G). A small modification to the original leader protocol was made in order to make certain state variables visible – unfortunately, state variables are hidden from direct analysis in mCRL2. The modification is the addition of a special **trace** action in the body of the main process followed by the main process recursively calling itself, which means that it should not influence the other behavior of the protocol in any way.

Next, the following property (formulated using μ-calculus combined with Hennessy-Milner logic) was checked for the leader protocol:

$$\mu X \,.\, (\nu Y \,.\, \phi \wedge [true]\, Y) \vee ([\neg \mathbf{trace}]\, X \wedge <\neg \mathbf{trace}> X)$$

where ϕ expresses a state in which all nodes in the network can do a **trace** action with the same arguments, namely the number and address of the selected leader node. The first disjunct of the property denotes a state where ϕ holds and from where no state can be reached such that $\neg\phi$. The rather long second disjunct of the property has been added so that any traces are rejected in which a **trace** action occurs while there are still other actions possible.

It was expected that the property would hold for the leader protocol, and this was confirmed by mCRL2. The protocol was subsequently changed so that the property should no longer hold, and this was also confirmed by mCRL2.

2 AODV protocol

The second use case makes use of the Ad hoc On-Demand Distance Vector (AODV) routing protocol [27], a protocol previously formalized in AWN [28, 29]. AODV is a widely used routing protocol designed for Mobile Ad-hoc Networks (MANETs) and Wireless Mesh Networks (WMNs).

As customary for internet protocols, AODV has been specified in an RFC document [27], which contains several ambiguities, contradictions, and sections that are underspecified. As a matter of fact, AWN was first developed to obtain a formal specification of AODV that was consistent and complete [28, 29]. This specification has been input in the implementation of this project in order to translate it to mCRL2 (see Appendix H).

mCRL2 was used to check AODV for a weak form of the *package-delivery property*, the property of whether a sent packet eventually arrives at its destination. This property was expressed as follows:

$$[\mathit{true}^*.\mathbf{trace}(\mathtt{newpkt}(\mathtt{dip},\mathtt{data}))]$$
$$[(\neg\mathbf{deliver}(\mathtt{dip},\mathtt{data}))^*]$$
$$<\mathit{true}^*><\mathbf{deliver}(\mathtt{dip},\mathtt{data})>\mathit{true}$$

The property looks for the special **trace** action, which has in this case been deployed to make the sending of a packet visible to the mCRL2 model-checker. The property states that every path that ends with a **trace** action *must* be followed by a path that, until the packet has been deliver, is at the very least capable of eventually delivering the packet.

The property was tested with mCRL2 for a static linear network of three nodes. Two tests were performed: one test where one packet was inserted into the network and one test where two packets were inserted into the network. As was expected, the first test succeeded and the second test failed: pen-and-paper analysis has shown in the past that the packet-delivery property does not hold for AODV [28].

6. Conclusions

1 Results summary

The project has yielded the following items:

- A formally defined function for translating AWN to mCRL2;
- A proof that shows that the output of the translation function is strongly bisimilar to its input;
- An implementation of the translation function;
- A specification of the AWN input language;
- An editor that accepts the AWN input language, serving as a front-end for the implementation;
- Some initial use cases of the implementation, demonstrating its usability.

2 Discussion

This section discusses (some of the) shortcomings of the items that the project produced. First, the shortcomings of the correctness proof:

- There is an important difference between the semantics of AWN and the semantics of mCRL2: sequential process expressions in AWN can carry semantic values, whereas process expression in mCRL2 consist exclusively of syntax. This difference has been resolved by introducing the translation rule T12, and by adding the index parameter ξ of the T_ξ function (the function that translates data expressions). Several lemmas make it possible to manipulate the contents of this parameter.
  An alternate approach to resolving the difference between the two semantics that was considered was defining an intermediate AWN language with exclusively syntactic expressions. This option was not chosen because it was estimated that the fact that the correctness proof would then also consist of two left-to-right and two right-to-left steps would result in more work than proving a somewhat more complicated left-to-right step and a somewhat more complicated right-to-left step.

If more correctness proofs are needed in the future with similar issues, it may be worthwhile to define an intermediate AWN language after all. It then becomes possible to base multiple correctness proofs on work that is done only once.

- The right-to-left part of the correctness proof (see Lemmas 4.8 and 4.9) requires the evaluation of all possible behavior of specific mCRL2 process expressions. This behavior is defined by the possible derivations for those process expressions, and it requires an understanding of the inference rules of mCRL2 to determine that all derivations for a particular process expression have been found (see Section 4.1). This reasoning is sufficiently convincing for the purposes of this project. Regardless, it is one of the weaker parts of the correctness proof.
  In order to increase the confidence in that part of the proof, one could use software that systematically searches for all possible derivations. These derivations can then be compared to the derivations found in the pen-and-paper proof in this report.

Second, the shortcomings of the implementation:

- The code that implements the AWN-to-mCRL2 translation closely reflects the rules of the translation function (see Section 3), which means that the correctness of the code can be assumed with a high degree of confidence. The same does *not* apply to the code that translates AWN data expressions, from which the correctness proof abstracts (see Section 3.16). The implementation of the translation consists for 68% (circa 1980 out of 2900 lines) of code that translates data expressions.
  Clearly, the translation of the data expressions is a significant weakness of the implementation. However, proving its correctness is much more involved than proving the correctness of the process algebra only, and the work required would not have fitted within the time frame of the project.
- The behavior of **undefined** values does not match the description in Section 3.16 at the time of writing. Instead, the current behavior treats each **undefined** value more or less as a constant; for example, `undefined Integer == undefined Integer` yields *true* rather than `undefined Boolean`.
  The use cases do not require the official behavior, and implementing it has therefore been given a low priority. Eventually, the work was skipped altogether due to time constraints.
  A full implementation would also make mCRL2 specifications more complicated, which could make manually debugging the software much more difficult. The additional mCRL2 code could also significantly increase the amount of time that it takes to analyze a specification. Regardless, it is desirable to implement the full behavior of **undefined** values in the future.
- The implementation does not include unit tests or another form of automated software testing. Having such testing methods is a good way to detect unforeseen errors that result from changes to the code, and it should be a high priority to add them if the translation framework is to used more intensively.
  Automated unit testing is not trivial to create because Xtext adds a *non-deterministic* component to the behavior of the implementation: Xtext can construct different – but equivalent – abstract syntax trees from the same AWN specification. This is a side-effect of ordering the objects in a set according to their hashes. It is an effect that can be overcome, but doing so has been considered out-of-scope for this project.
- The use cases that were performed with the help of the implementation were small; that

is, the network topologies that were tested only contained up to 5 nodes. These network topologies already required much time and memory to analyze (see Chapter 5). However, none of mCRL2's state space reduction methods were used, nor any of LTSmin's multi-threaded algorithms. The expectation is that employing such techniques will make the analysis of larger network topologies feasible.

- The current implementation has been developed so that translations can be altered while the editor for the AWN input language is running: the code that defined the translation is reloaded before each execution. This is useful when developing a translation – one does not have to frequently restart the editor, and third parties do not require knowledge of or access to the translation framework – but it also impacts the performance of the implementation. In the future, it may be worth changing the implementation so that the translation is only loaded at start-up.
- Related to the previous point is the fact that the model-to-text converter of the framework (see Section 2.4) is a custom-built one rather than one that is based on Ecore Modelling Framework, such as Acceleo and Xpand. The reason for this is that after serious efforts no progress was made towards loading model-to-text translations at run-time with those converters. They therefore did not meet the (admittedly secondary) goal of allowing third parties do develop their own translations, and an alternative was developed.
  Existing model-to-text converters also only supported the export of single text files (unless the documentation for exporting multiple text files per model was missing or overlooked). Since translations can very easily be imagined to have multiple output files, this was an additional reason to develop a custom converter.
  The disadvantage of using a custom-built model-to-text converter is, of course, that one must learn to use it. It may also contain errors (considering that there is no wider community that makes use of it) and it will likely require maintenance in the future. Because of this, it would be prudent to re-evaluate the use of the custom-built converter when more translations are added to the translation framework.

3 Future work

Several aspects of the translation framework have been left unfinished and may merit continuation in the future:

- Implement **undefined** according to its description in Section 3.16;
- Add automated (unit) tests to the implementation, in case of further development;
- Make the implementation even more user-friendly, for example by linking mCRL2 (or LTSmin) more directly with Eclipse;
- Analyze larger network topologies by using mCRL2's reduction techniques and possibly LTSmin;
- Implement other translations (such as a translation to UPPAAL) and code generation;
- Implement the translation of properties such as packet deliver and loop freedom;
- Optimize the implementation.

Bibliography

[1] S. Bradner. *IETF Working Group Guidelines and Procedures*. RFC 2418. RFC Editor, Sept. 1998, pages 19–21. URL: https://tools.ietf.org/html/rfc2418 (cited on page 13).

[2] C. Villamizar, R. Chandra, and R. Govindan. *BGP Route Flap Damping*. RFC 2418. RFC Editor, Nov. 1998, pages 19–21. URL: https://www.ietf.org/rfc/rfc2439.txt (cited on page 13).

[3] Zhuoqing Morley Mao et al. "Route flap damping exacerbates Internet routing convergence". In: *ACM SIGCOMM Computer Communication Review*. Volume 32. 4. ACM. 2002, pages 221–233 (cited on page 13).

[4] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. *Ad hoc on-demand distance vector (AODV) routing*. Technical report. 2003 (cited on page 13).

[5] Rob Van Glabbeek et al. "Sequence numbers do not guarantee loop freedom: AODV can yield routing loops". In: *Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems*. ACM. 2013, pages 91–100 (cited on pages 13, 14).

[6] Ansgar Fehnker et al. "A process algebra for wireless mesh networks". In: *European Symposium on Programming*. Springer. 2012, pages 295–315 (cited on pages 13, 14, 19, 20).

[7] Ansgar Fehnker et al. "A process algebra for wireless mesh networks used for modelling, verifying and analysing AODV". In: *arXiv preprint arXiv:1312.7645* (2013) (cited on page 13).

[8] Peter Höfner et al. "A rigorous analysis of AODV and its variants". In: *Proceedings of the 15th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*. ACM. 2012, pages 203–212 (cited on page 14).

[9] Emile Bres, Rob van Glabbeek, and Peter Höfner. "A timed process algebra for wireless networks with an application in routing". In: *European Symposium on Programming Languages and Systems*. Springer. 2016, pages 95–122 (cited on page 14).

[10] Jan Friso Groote et al. "The formal specification language mCRL2". In: *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2007 (cited on page 14).

[11] Rob van Glabbeek, Peter Höfner, and Djurre van der Wal. "Analysing AWN-specifications using mCRL2". In: *International Conference on integrated Formal Methods*. iFM. 2018 (cited on page 14).

[12] *List of model checking tools*. https://en.wikipedia.org/wiki/List_of_model_checking_tools. [Online; accessed 1-October-2017]. 2017 (cited on page 15).

[13] *List of verification and synthesis tools*. https://github.com/johnyf/tool_lists/blob/master/verification_synthesis.md. [Online; accessed 1-October-2017]. 2017 (cited on page 15).

[14] *Yahoda verification tools database*. https://web.archive.org/web/20151106214218/http://anna.fi.muni.cz/yahoda/. [Online; accessed 1-October-2017]. 2017 (cited on page 15).

[15] Stefan Blom, Jaco van de Pol, and Michael Weber. "LTSmin: Distributed and Symbolic Reachability." In: *CAV*. Volume 6174. Springer. 2010, pages 354–359 (cited on pages 16, 17).

[16] André Arnold et al. "The AltaRica formalism for describing concurrent systems". In: *Fundamenta Informaticae* 40.2, 3 (1999), pages 109–124 (cited on page 17).

[17] Hubert Garavel et al. "CADP 2011: a toolbox for the construction and analysis of distributed processes". In: *International Journal on Software Tools for Technology Transfer* 15.2 (2013), pages 89–107 (cited on page 17).

[18] Thomas Gibson-Robinson et al. "FDR3—a modern refinement checker for CSP". In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2014, pages 187–201 (cited on page 17).

[19] Sjoerd Cranen et al. "An Overview of the mCRL2 Toolset and Its Recent Advances." In: *TACAS*. Volume 13. Springer. 2013, pages 199–213 (cited on page 17).

[20] Alessandro Cimatti et al. "NuSMV 2: An opensource tool for symbolic model checking". In: *International Conference on Computer Aided Verification*. Springer. 2002, pages 359–364 (cited on page 17).

[21] Gerard J. Holzmann. "The model checker SPIN". In: *IEEE Transactions on software engineering* 23.5 (1997), pages 279–295 (cited on page 17).

[22] Gregor Kiczales et al. "Aspect-oriented programming". In: *ECOOP'97—Object-oriented programming* (1997), pages 220–242 (cited on page 17).

[23] FPM Stappers et al. "Dogfooding the structural operational semantics of mCRL2". In: *Computer Science Report* 11-18 (2011) (cited on page 28).

[24] Jan Friso Groote and Mohammad Reza Mousavi. *Modeling and analysis of communicating systems*. MIT press, 2014 (cited on pages 28, 31).

[25] Gordon D Plotkin. "A structural approach to operational semantics". In: (1981) (cited on page 28).

[26] Robin Milner. *Communication and concurrency*. Volume 84. Prentice hall New York etc., 1989 (cited on page 52).

[27] C. E. Perkins, E. M. Belding-Royer, and S. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing*. RFC 3561 (Experimental), Network Working Group. 2003. URL: http://www.ietf.org/rfc/rfc3561.txt (cited on page 90).

[28] Ansgar Fehnker et al. *A Process Algebra for Wireless Mesh Networks used for Modelling, Verifying and Analysing AODV*. TR. 2013. URL: http://arxiv.org/abs/1312.7645 (cited on page 90).

[29] R. J. van Glabbeek et al. "Modelling and Verifying the AODV Routing Protocol". In: *Distributed Computing* 29.4 (2016), pages 279–315. DOI: 10.1007/s00446-015-0262-7 (cited on page 90).

Appendices

Appendix A

Proof of Theorem 4.1

Theorem: Data congruence in mCRL2 is a strong bisimulation.

Proof: The proof is provided by structural induction over the inference rules of mCRL2 (see Tables 2.6 and 2.7).

Induction hypothesis: For all premises $p \xrightarrow{\omega} p'$ of an mCRL2 inference rule, it holds that

$$\forall q \,.\, q \equiv p \Rightarrow \exists q' \,.\, q' \equiv p' \wedge q \xrightarrow{\omega} p'$$

Base cases: Show for each axiomatic inference rule of mCRL2 with conclusion $r \xrightarrow{\omega} r'$ that for all $s \equiv r$ there exists some $s' \equiv r'$ such that $s \xrightarrow{\omega} s'$.

Axiom: mCRL2 has only one (untimed) axiomatic inference rule, namely:

$$\frac{}{\alpha \xrightarrow{[\![\alpha]\!]} \checkmark}\ \textsc{Axiom}$$

In other words, $r = \alpha$ and $r' = \checkmark$.
For any $s \equiv r$, it holds that $[\![s]\!] = [\![r]\!] = [\![\alpha]\!]$. This means that the following derivation is valid:

$$\frac{}{s \xrightarrow{[\![\alpha]\!]} \checkmark}\ \textsc{Axiom}$$

Obviously, $\checkmark \equiv \checkmark$. This finishes the base case of Theorem 4.1.

Induction step: Given only its side conditions and the induction hypothesis, show for each inference rule of mCRL2 with conclusion $r \xrightarrow{\omega} r'$ that for all $s \equiv r$ there exists some $s' \equiv r'$ such that $s \xrightarrow{\omega} s'$.

Seq 1: mCRL2 defines inference rule

$$\frac{p \xrightarrow{\omega} \checkmark}{p.q \xrightarrow{\omega} q}\ \textsc{Seq 1}$$

In other words, $r = p.q$ and $r' = q$.
Because data expressions can only be part of a single process, for any $s \equiv p.q$ it must be the case that

$$\exists s_1, s_2 \,.\, s_1 \equiv p \wedge s_2 \equiv q \wedge s = s_1.s_2$$

This means that the following derivation is valid:

$$\dfrac{\dfrac{}{s_1 \xrightarrow{\omega} \checkmark}\ \text{Induction hypothesis}}{s_1.s_2 \xrightarrow{\omega} s_2}\ \textsc{Seq 1}$$

Since $s_2 \equiv q$, this particular induction step of Theorem 4.1 is confirmed.

Sum 1: mCRL2 defines inference rule

$$e \in M_D\ \frac{p[d := t_e] \xrightarrow{\omega} p'}{\sum_{d:D} p \xrightarrow{\omega} p'}\ \textsc{Sum 2}$$

In other words, $r = \sum_{d:D} p$ and $r' = p'$.
Because of the form,

$$\forall s \,.\, s \equiv r \Rightarrow \exists q \,.\, s = \textstyle\sum_{d:D} q$$

Note that $\mathrm{FV}(q) = \mathrm{FV}(p) \subseteq \{d\}$ which, with the previous expression, gives that

$$q[d := t_u] \equiv p[d := t_u]$$

for all $u \in M_D$. In particular, this equation holds for $u = e$.
The induction hypothesis can now be used to determine that

$$p[d := t_e] \xrightarrow{\omega} p' \wedge q[d := t_e] \equiv p[d := t_e] \Rightarrow \left(\exists q' \,.\, q' \equiv p' \wedge q[d := t_e] \xrightarrow{\omega} q'\right)$$

This means that the following derivation is valid:

$$e \in M_D\ \frac{q[d := t_e] \xrightarrow{\omega} q'}{\sum_{d:D} q \xrightarrow{\omega} q'}\ \textsc{Sum 2}$$

Since $q' \equiv r'$, this particular induction step of Theorem 4.1 is confirmed.

Rec 1: mCRL2 defines inference rule

$$\mathrm{P}(\mathtt{d}_1 : \mathrm{D}_1, \cdots, \mathtt{d}_n : \mathrm{D}_n) \stackrel{\text{def}}{=} \mathrm{q} \quad \frac{\mathrm{q}[\mathtt{d}_1 := t_1, \cdots, \mathtt{d}_n := t_n] \xrightarrow{\omega} \mathrm{q}'}{\mathrm{P}(t_1, \cdots, t_n) \xrightarrow{\omega} \mathrm{q}'} \text{ RECURSION 2}$$

In other words, $\mathrm{r} = \mathrm{P}(t_1, \cdots, t_n)$ and $\mathrm{r}' = \mathrm{q}'$.
Because data expressions cannot affect data expressions located within another process argument, for any $\mathrm{s} \equiv \mathrm{P}(t_1, \cdots, t_n)$ it must be the case that

$$\exists u_1, \cdots u_n \,.\, u_1 \equiv t_1 \wedge \cdots \wedge u_n \equiv t_n \wedge \mathrm{s} = \mathrm{P}(u_1, \cdots, u_n)$$

Furthermore,

$$\forall u_1, \cdots u_n \,.\, u_1 \equiv t_1 \wedge \cdots \wedge u_n \equiv t_n \Rightarrow \mathrm{q}[\mathtt{v}_1 := u_1, \cdots, \mathtt{u}_n := t_n] \equiv \mathrm{q}[\mathtt{v}_1 := t_1, \cdots, \mathtt{v}_n := t_n]$$

According to the induction hypothesis, this means that there exists an $\mathrm{s}' \equiv \mathrm{q}'$ such that

$$\mathrm{q}[\mathtt{v}_1 := u_1, \cdots, \mathtt{u}_n := t_n] \xrightarrow{\omega} \mathrm{s}'$$

This yields the start of the following derivation:

$$\mathrm{P}(\mathtt{d}_1 : \mathrm{D}_1, \cdots, \mathtt{d}_n : \mathrm{D}_n) \stackrel{\text{def}}{=} \mathrm{q} \quad \frac{\dfrac{}{\mathrm{q}[\mathtt{d}_1 := u_1, \cdots, \mathtt{d}_n := u_n] \xrightarrow{\omega} \mathrm{s}'}\ \textit{(see above)}}{\mathrm{P}(u_1, \cdots, u_n) \xrightarrow{\omega} \mathrm{s}'} \text{ RECURSION 2}$$

Since $\mathrm{s}' \equiv \mathrm{r}'$, this particular induction step of Theorem 4.1 is confirmed.

The proofs for the other induction steps are similar to the ones above and are not included in this document. All the proofs combined prove that Theorem 4.1 holds.

Appendix B

Complete proof of Lemma 4.6

Lemma: If e is a semantic AWN value and $\mathtt{v}$ an AWN variable, then

$$\mathtt{v} \in V \wedge \mathtt{v} \notin \textsc{dom}(\xi) \Rightarrow \mathrm{T}_V(\xi, \mathrm{p}) \left[\mathrm{T}(\mathtt{v}) := t_{\mathrm{U}(e)}\right] = \mathrm{T}_V(\xi[\mathtt{v} := e], \mathrm{p})$$

Proof: The proof is provided by structural induction over the translation rules T1 to T10.

Induction hypothesis: For all recursions $\mathrm{T}_W(\xi, \mathrm{p})$ that are part of a translation rule defining T_V, it holds that

$$\mathtt{v} \in W \wedge \mathtt{v} \notin \textsc{dom}(\xi) \Rightarrow \mathrm{T}_W(\xi, \mathrm{p}) \left[\mathrm{T}(\mathtt{v}) := t_{\mathrm{U}(e)}\right] = \mathrm{T}_W(\xi[\mathtt{v} := e], \mathrm{p})$$

1 Base case

Translation rule T8: Applying a substitution to T8 gives the following:

$$\begin{aligned}
&\mathrm{T}_V(\xi, \mathrm{X}(exp_1, \cdots, exp_n))[\mathrm{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\quad \overset{\text{T8}}{=} \quad \mathrm{X}(\mathrm{T}_\xi(exp_1), \cdots, \mathrm{T}_\xi(exp_n))[\mathrm{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\quad = \quad \mathrm{X}(\mathrm{T}_\xi(exp_1)[\mathrm{T}(\mathtt{v}) := t_{\mathrm{U}(e)}], \cdots, \mathrm{T}_\xi(exp_n)[\mathrm{T}(\mathtt{v}) := t_{\mathrm{U}(e)}]) \\
&\quad \overset{\text{Lemma 4.5 } (n \text{ times})}{=} \quad \mathrm{X}(\mathrm{T}_{\xi[\mathtt{v}:=e]}(exp_1), \cdots, \mathrm{T}_{\xi[\mathtt{v}:=e]}(exp_n)) \\
&\quad \overset{\text{T8}}{=} \quad \mathrm{T}_V(\xi[\mathtt{v} := e], \mathrm{X}(exp_1, \cdots, exp_n))
\end{aligned}$$

2 Induction step

Translation rule T1: As a preparatory step, observe that

$$\left.\begin{array}{l} \mathtt{v} \in V \Rightarrow \mathtt{T}(\mathtt{v}) \in \mathtt{T}(V) \\ \mathtt{D} \notin \mathtt{T}(V) \end{array}\right\} \Rightarrow \mathtt{D} \neq \mathtt{T}(\mathtt{v}) \tag{B.1}$$

Applying a substitution to T1 gives the following:

$$\begin{aligned}
&\mathrm{T}_V(\xi, \mathbf{broadcast}(ms).\mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{T1}}{=} \left(\textstyle\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set}(\mathrm{IP}))} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathtt{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p})\right)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{B.1}}{=} \textstyle\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set}(\mathrm{IP}))} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}}[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}], \mathtt{D}, \mathtt{T}_\xi(ms)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}]).\mathrm{T}_V(\xi, \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\mathscr{U}_{\mathrm{IP}} \text{ is closed}}{=} \textstyle\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set}(\mathrm{IP}))} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathtt{T}_\xi(ms)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}]).\mathrm{T}_V(\xi, \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{Lemma 4.5}}{=} \textstyle\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set}(\mathrm{IP}))} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathtt{T}_{\xi[\mathtt{v}:=e]}(ms)).\mathrm{T}_V(\xi, \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{I.H.}}{=} \textstyle\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set}(\mathrm{IP}))} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathtt{T}_{\xi[\mathtt{v}:=e]}(ms)).\mathrm{T}_V(\xi[\mathtt{v} := e], \mathrm{p}) \\
&\overset{\text{T1}}{=} \mathrm{T}_V(\xi[\mathtt{v} := e], \mathbf{broadcast}(ms).\mathrm{p})
\end{aligned}$$

This proves this particular induction step.

Translation rule T2: As a preparatory step, observe that

$$\left.\begin{array}{l} \mathtt{v} \in V \Rightarrow \mathtt{T}(\mathtt{v}) \in \mathtt{T}(V) \\ \mathtt{D} \notin \mathtt{T}(V) \end{array}\right\} \Rightarrow \mathtt{D} \neq \mathtt{T}(\mathtt{v}) \tag{B.2}$$

Applying a substitution to T2 gives the following:

$$\begin{aligned}
&\mathrm{T}_V(\xi, \mathbf{groupcast}(dests, ms).\mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{T2}}{=} \left(\textstyle\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set}(\mathrm{IP}))} \mathbf{cast}(\mathtt{T}_\xi(dests), \mathtt{D}, \mathtt{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p})\right)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{B.2}}{=} \textstyle\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set}(\mathrm{IP}))} \mathbf{cast}(\mathtt{T}_\xi(dests)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}], \mathtt{D}, \mathtt{T}_\xi(ms)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}]).\mathrm{T}_V(\xi, \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{Lemma 4.5 (2 times)}}{=} \textstyle\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set}(\mathrm{IP}))} \mathbf{cast}(\mathtt{T}_{\xi[\mathtt{v}:=e]}(dests), \mathtt{D}, \mathtt{T}_{\xi[\mathtt{v}:=e]}(ms)).\mathrm{T}_V(\xi, \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{I.H.}}{=} \textstyle\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set}(\mathrm{IP}))} \mathbf{cast}(\mathtt{T}_{\xi[\mathtt{v}:=e]}(dests), \mathtt{D}, \mathtt{T}_{\xi[\mathtt{v}:=e]}(ms)).\mathrm{T}_V(\xi[\mathtt{v} := e], \mathrm{p}) \\
&\overset{\text{T2}}{=} \mathrm{T}_V(\xi[\mathtt{v} := e], \mathbf{groupcast}(dests, ms).\mathrm{p})
\end{aligned}$$

This proves this particular induction step.

Translation rule T3: Applying a substitution to T3 gives the following:

$$
\begin{aligned}
&\mathrm{T}_V(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{T3}}{=} (\mathbf{cast}(\{\mathrm{T}_\xi(dest)\}, \{\mathrm{T}_\xi(dest)\}, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p}) \\
&\qquad + \neg\mathbf{uni}(\{\mathrm{T}_\xi(dest)\}, \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{q}))[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&= \big(\mathbf{cast}(\{\mathrm{T}_\xi(dest)\}, \{\mathrm{T}_\xi(dest)\}, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p})\big)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\qquad + \big(\neg\mathbf{uni}(\{\mathrm{T}_\xi(dest)\}, \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{q})\big)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{Lemma 4.5 (5 times)}}{=} \mathbf{cast}(\{\mathrm{T}_{\xi[\mathtt{v}:=e]}(dest)\}, \{\mathrm{T}_{\xi[\mathtt{v}:=e]}(dest)\}, \mathrm{T}_{\xi[\mathtt{v}:=e]}(ms)).\mathrm{T}_V(\xi, \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\qquad + \neg\mathbf{uni}(\{\mathrm{T}_{\xi[\mathtt{v}:=e]}(dest)\}, \emptyset, \mathrm{T}_{\xi[\mathtt{v}:=e]}(ms)).\mathrm{T}_V(\xi, \mathrm{q})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{I.H. (2 times)}}{=} \mathbf{cast}(\{\mathrm{T}_{\xi[\mathtt{v}:=e]}(dest)\}, \{\mathrm{T}_{\xi[\mathtt{v}:=e]}(dest)\}, \mathrm{T}_{\xi[\mathtt{v}:=e]}(ms)).\mathrm{T}_V(\xi[\mathtt{v} := e], \mathrm{p}) \\
&\qquad + \neg\mathbf{uni}(\{\mathrm{T}_{\xi[\mathtt{v}:=e]}(dest)\}, \emptyset, \mathrm{T}_{\xi[\mathtt{v}:=e]}(ms)).\mathrm{T}_V(\xi[\mathtt{v} := e], \mathrm{q}) \\
&\overset{\text{T3}}{=} \mathrm{T}_V(\xi[\mathtt{v} := e], \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q})
\end{aligned}
$$

This proves this particular induction step.

Translation rule T4: Applying a substitution to T4 gives the following:

$$
\begin{aligned}
&\mathrm{T}_V(\xi, \mathbf{send}(\mathrm{T}_\xi(ms)).\mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{T3}}{=} \big(\mathbf{send}(\emptyset, \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p})\big)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&= \mathbf{send}(\emptyset, \emptyset, \mathrm{T}_\xi(ms)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}]).\mathrm{T}_V(\xi, \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{Lemma 4.5}}{=} \mathbf{send}(\emptyset, \emptyset, \mathrm{T}_{\xi[\mathtt{v}:=e]}(ms)).\mathrm{T}_V(\xi, \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{I.H.}}{=} \mathbf{send}(\emptyset, \emptyset, \mathrm{T}_{\xi[\mathtt{v}:=e]}(ms)).\mathrm{T}_V(\xi[\mathtt{v} := e], \mathrm{p}) \\
&\overset{\text{T4}}{=} \mathrm{T}_V(\xi, \mathbf{send}(\mathrm{T}_\xi(ms)).\mathrm{p})
\end{aligned}
$$

This proves this particular induction step.

Translation rule T5: Applying a substitution to T5 gives the following:

$$
\begin{aligned}
&\mathrm{T}_V(\xi, \mathbf{deliver}(data).\mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{T5}}{=} \big(\textstyle\sum_{\mathtt{ip}:\mathrm{T}(\mathrm{IP})} \mathbf{del}(\mathtt{ip}, \mathrm{T}_\xi(data)).\mathrm{T}_V(\xi, \mathrm{p})\big)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\mathtt{ip} \neq \mathtt{T}(\mathtt{v})}{=} \textstyle\sum_{\mathtt{ip}:\mathrm{T}(\mathrm{IP})} \mathbf{del}(\mathtt{ip}, \mathrm{T}_\xi(data)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}]).\mathrm{T}_V(\xi, \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{Lemma 4.5}}{=} \textstyle\sum_{\mathtt{ip}:\mathrm{T}(\mathrm{IP})} \mathbf{del}(\mathtt{ip}, \mathrm{T}_{\xi[\mathtt{v}:=e]}(data)).\mathrm{T}_V(\xi, \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{I.H.}}{=} \textstyle\sum_{\mathtt{ip}:\mathrm{T}(\mathrm{IP})} \mathbf{del}(\mathtt{ip}, \mathrm{T}_{\xi[\mathtt{v}:=e]}(data)).\mathrm{T}_V(\xi[\mathtt{v} := e], \mathrm{p}) \\
&\overset{\text{T5}}{=} \mathrm{T}_V(\xi[\mathtt{v} := e], \mathbf{deliver}(data).\mathrm{p})
\end{aligned}
$$

This proves this particular induction step.

Translation rule T6: A case distinction is made depending on whether $\mathtt{T}(\mathtt{v}) = \mathtt{T}(\mathtt{msg})$. If it is assumed that $\mathtt{T}(\mathtt{v}) = \mathtt{T}(\mathtt{msg})$, applying a substitution to T6 gives the following:

$$
\begin{aligned}
&\mathrm{T}_V(\xi, \textbf{receive}(\mathtt{msg}).\mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\mathtt{T}(\mathtt{v})=\mathtt{T}(\mathtt{msg})}{=} \mathrm{T}_V(\xi, \textbf{receive}(\mathtt{msg}).\mathrm{p})[\mathtt{T}(\mathtt{msg}) := t_{\mathrm{U}(e)}] \\
&\overset{\mathrm{T6}}{=} \left(\textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathtt{T}(\mathrm{Set(IP)}),\mathtt{T}(\mathtt{msg}):\mathtt{T}(\mathrm{MSG})} \textbf{receive}(\mathtt{D},\mathtt{D'},\mathtt{T}(\mathtt{msg})).\mathrm{T}_{V\cup\{\mathtt{msg}\}}(\xi^{\setminus\mathtt{msg}},\mathrm{p})\right)[\mathtt{T}(\mathtt{msg}) := t_{\mathrm{U}(e)}] \\
&= \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathtt{T}(\mathrm{Set(IP)}),\mathtt{T}(\mathtt{msg}):\mathtt{T}(\mathrm{MSG})} \textbf{receive}(\mathtt{D},\mathtt{D'},\mathtt{T}(\mathtt{msg})).\mathrm{T}_{V\cup\{\mathtt{msg}\}}(\xi^{\setminus\mathtt{msg}},\mathrm{p}) \\
&= \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathtt{T}(\mathrm{Set(IP)}),\mathtt{T}(\mathtt{msg}):\mathtt{T}(\mathrm{MSG})} \textbf{receive}(\mathtt{D},\mathtt{D'},\mathtt{T}(\mathtt{msg})).\mathrm{T}_{V\cup\{\mathtt{msg}\}}((\xi[\mathtt{msg} := e])^{\setminus\{\mathtt{msg}\}},\mathrm{p}) \\
&\overset{\mathrm{T6}}{=} \mathrm{T}_V(\xi[\mathtt{msg} := e], \textbf{receive}(\mathtt{msg}).\mathrm{p})
\end{aligned}
$$

If it is assumed that $\mathtt{T}(\mathtt{v}) \neq \mathtt{T}(\mathtt{msg})$, it first must be observed that

$$
\left.\begin{array}{l} \mathtt{v} \in V \Rightarrow \mathtt{T}(\mathtt{v}) \in \mathtt{T}(V) \\ \mathtt{D},\mathtt{D'} \notin \mathtt{T}(V) \end{array}\right\} \Rightarrow \mathtt{D},\mathtt{D'} \neq \mathtt{T}(\mathtt{v}) \tag{B.3}
$$

Now the same result can be obtained as before:

$$
\begin{aligned}
&\mathrm{T}_V(\xi, \textbf{receive}(\mathtt{msg}).\mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\mathrm{T6}}{=} \left(\textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathtt{T}(\mathrm{Set(IP)}),\mathtt{T}(\mathtt{msg}):\mathtt{T}(\mathrm{MSG})} \textbf{receive}(\mathtt{D},\mathtt{D'},\mathtt{T}(\mathtt{msg})).\mathrm{T}_{V\cup\{\mathtt{msg}\}}(\xi^{\setminus\mathtt{msg}},\mathrm{p})\right)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\mathrm{B.3},\,\mathtt{T}(\mathtt{v})\neq\mathtt{T}(\mathtt{msg})}{=} \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathtt{T}(\mathrm{Set(IP)}),\mathtt{T}(\mathtt{msg}):\mathtt{T}(\mathrm{MSG})} \textbf{receive}(\mathtt{D},\mathtt{D'},\mathtt{T}(\mathtt{msg})).\mathrm{T}_{V\cup\{\mathtt{msg}\}}(\xi^{\setminus\mathtt{msg}},\mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\mathrm{I.H.}}{=} \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathtt{T}(\mathrm{Set(IP)}),\mathtt{T}(\mathtt{msg}):\mathtt{T}(\mathrm{MSG})} \textbf{receive}(\mathtt{D},\mathtt{D'},\mathtt{T}(\mathtt{msg})).\mathrm{T}_{V\cup\{\mathtt{msg}\}}(\xi^{\setminus\mathtt{msg}}[\mathtt{v} := e],\mathrm{p}) \\
&= \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathtt{T}(\mathrm{Set(IP)}),\mathtt{T}(\mathtt{msg}):\mathtt{T}(\mathrm{MSG})} \textbf{receive}(\mathtt{D},\mathtt{D'},\mathtt{T}(\mathtt{msg})).\mathrm{T}_{V\cup\{\mathtt{msg}\}}((\xi[\mathtt{v} := e])^{\setminus\{\mathtt{msg}\}},\mathrm{p}) \\
&\overset{\mathrm{T6}}{=} \mathrm{T}_V(\xi[\mathtt{v} := e], \textbf{receive}(\mathtt{msg}).\mathrm{p})
\end{aligned}
$$

This proves this particular induction step.

Translation rule T7: As a preparatory step, note that

$$\left.\begin{array}{l} \mathtt{v} \in V \Rightarrow \mathtt{T}(\mathtt{v}) \in \mathtt{T}(V) \\ \mathtt{tmp} \notin \mathtt{T}(V) \end{array}\right\} \Rightarrow \mathtt{tmp} \neq \mathtt{T}(\mathtt{v}) \tag{B.4}$$

A case distinction is made depending on whether $\mathtt{T}(\mathtt{v}) = \mathtt{T}(\mathtt{var})$. If it is assumed that $\mathtt{T}(\mathtt{v}) = \mathtt{T}(\mathtt{var})$, applying a substitution to T7 gives the following:

$$\begin{array}{ll}
& \mathrm{T}_V(\xi, [\![\mathtt{var} := exp]\!]\ \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
\overset{\mathtt{T}(\mathtt{v})=\mathtt{T}(\mathtt{var})}{=} & \mathrm{T}_V(\xi, [\![\mathtt{var} := exp]\!]\ \mathrm{p})[\mathtt{T}(\mathtt{var}) := t_{\mathrm{U}(e)}] \\
\overset{\mathrm{T7}}{=} & (\textstyle\sum_{\mathtt{tmp}:\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{tmp} = \mathrm{T}_\xi(exp)) \to \sum_{\mathtt{T}(\mathtt{var}):\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{T}(\mathtt{var}) = \mathtt{tmp}) \to \\
& \qquad \mathbf{t}(\emptyset, \emptyset, \mathtt{msg}_{\,dummy}).\mathrm{T}_{V\cup\{\mathtt{var}\}}(\xi^{\setminus \mathtt{var}}, \mathrm{p}))[\mathtt{T}(\mathtt{var}) := t_{\mathrm{U}(e)}] \\
\overset{\text{B.4}}{=} & \textstyle\sum_{\mathtt{tmp}:\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{tmp} = \mathrm{T}_\xi(exp)[\mathtt{T}(\mathtt{var}) := t_{\mathrm{U}(e)}]) \to \sum_{\mathtt{T}(\mathtt{var}):\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{T}(\mathtt{var}) = \mathtt{tmp}) \to \\
& \qquad \mathbf{t}(\emptyset, \emptyset, \mathtt{msg}_{\,dummy}).\mathrm{T}_{V\cup\{\mathtt{var}\}}(\xi^{\setminus \mathtt{var}}, \mathrm{p}) \\
\overset{\text{Lemma 4.5}}{=} & \textstyle\sum_{\mathtt{tmp}:\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{tmp} = \mathrm{T}_{\xi[\mathtt{var}:=e]}(exp)) \to \sum_{\mathtt{T}(\mathtt{var}):\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{T}(\mathtt{var}) = \mathtt{tmp}) \to \\
& \qquad \mathbf{t}(\emptyset, \emptyset, \mathtt{msg}_{\,dummy}).\mathrm{T}_{V\cup\{\mathtt{var}\}}(\xi^{\setminus \mathtt{var}}, \mathrm{p}) \\
\overset{\mathrm{T7}}{=} & \mathrm{T}_V(\xi[\mathtt{var} := e], [\![\mathtt{var} := exp]\!]\ \mathrm{p})
\end{array}$$

If it is assumed that $\mathtt{T}(\mathtt{v}) \neq \mathtt{T}(\mathtt{var})$, the same result can be obtained:

$$\begin{array}{ll}
& \mathrm{T}_V(\xi, [\![\mathtt{var} := exp]\!]\ \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
\overset{\mathrm{T7}}{=} & (\textstyle\sum_{\mathtt{tmp}:\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{tmp} = \mathrm{T}_\xi(exp)) \to \sum_{\mathtt{T}(\mathtt{var}):\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{T}(\mathtt{var}) = \mathtt{tmp}) \to \\
& \qquad \mathbf{t}(\emptyset, \emptyset, \mathtt{msg}_{\,dummy}).\mathrm{T}_{V\cup\{\mathtt{var}\}}(\xi^{\setminus \mathtt{var}}, \mathrm{p}))[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
\overset{\text{B.4}}{=} & \textstyle\sum_{\mathtt{tmp}:\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{tmp} = \mathrm{T}_\xi(exp)[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}]) \to (\sum_{\mathtt{T}(\mathtt{var}):\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{T}(\mathtt{var}) = \mathtt{tmp}) \to \\
\overset{\mathtt{T}(\mathtt{v})\neq\mathtt{T}(\mathtt{var})}{=} & \textstyle\sum_{\mathtt{tmp}:\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{tmp} = \mathrm{T}_\xi(exp)) \to \sum_{\mathtt{T}(\mathtt{var}):\mathrm{sort}(\mathtt{T}(\mathtt{var}))})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \to \\
& \qquad \mathbf{t}(\emptyset, \emptyset, \mathtt{msg}_{\,dummy}).\mathrm{T}_{V\cup\{\mathtt{var}\}}(\xi^{\setminus \mathtt{var}}, \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
\overset{\text{Lemma 4.5}}{=} & \textstyle\sum_{\mathtt{tmp}:\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{tmp} = \mathrm{T}_{\xi[\mathtt{v}:=e]}(exp)) \to \sum_{\mathtt{T}(\mathtt{var}):\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{T}(\mathtt{var}) = \mathtt{tmp}) \to \\
& \qquad \mathbf{t}(\emptyset, \emptyset, \mathtt{msg}_{\,dummy}).\mathrm{T}_{V\cup\{\mathtt{var}\}}(\xi^{\setminus \{\mathtt{var}\}}, \mathrm{p})[\mathtt{T}(\mathtt{v}) := t_{\mathrm{U}(e)}] \\
\overset{\text{I.H.}}{=} & \textstyle\sum_{\mathtt{tmp}:\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{tmp} = \mathrm{T}_{\xi[\mathtt{v}:=e]}(exp)) \to \sum_{\mathtt{T}(\mathtt{var}):\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{T}(\mathtt{var}) = \mathtt{tmp}) \to \\
& \qquad \mathbf{t}(\emptyset, \emptyset, \mathtt{msg}_{\,dummy}).\mathrm{T}_{V\cup\{\mathtt{var}\}}(\xi^{\setminus \{\mathtt{var}\}}[\mathtt{v} := e], \mathrm{p}) \\
= & \textstyle\sum_{\mathtt{tmp}:\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{tmp} = \mathrm{T}_{\xi[\mathtt{v}:=e]}(exp)) \to \sum_{\mathtt{T}(\mathtt{var}):\mathrm{sort}(\mathtt{T}(\mathtt{var}))}(\mathtt{T}(\mathtt{var}) = \mathtt{tmp}) \to \\
& \qquad \mathbf{t}(\emptyset, \emptyset, \mathtt{msg}_{\,dummy}).\mathrm{T}_{V\cup\{\mathtt{var}\}}((\xi[\mathtt{v} := e])^{\setminus \mathtt{var}}, \mathrm{p}) \\
\overset{\mathrm{T7}}{=} & \mathrm{T}_V(\xi[\mathtt{v} := e], [\![\mathtt{var} := exp]\!]\ \mathrm{p})
\end{array}$$

This proves this particular induction step.

Translation rule T9: Applying a substitution to T9 gives the following:

$$
\begin{aligned}
&\mathrm{T}_V(\xi,\mathrm{p}+\mathrm{q})[\mathrm{T}(\mathrm{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{T9}}{=} \quad (\mathrm{T}_V(\xi,\mathrm{p})+\mathrm{T}_V(\xi,\mathrm{q}))\,[\mathrm{T}(\mathrm{v}) := t_{\mathrm{U}(e)}] \\
&= \quad \mathrm{T}_V(\xi,\mathrm{p})[\mathrm{T}(\mathrm{v}) := t_{\mathrm{U}(e)}] + \mathrm{T}_V(\xi,\mathrm{q})[\mathrm{T}(\mathrm{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{I.H.}}{=} \quad \mathrm{T}_V(\xi[\mathrm{v} := e],\mathrm{p}) + \mathrm{T}_V(\xi[\mathrm{v} := e],\mathrm{q}) \\
&\overset{\text{T9}}{=} \quad \mathrm{T}_V(\xi[\mathrm{v} := e],\mathrm{p}+\mathrm{q})
\end{aligned}
$$

This proves this particular induction step.

Translation rule T10: As a preparatory step, note that

$$
\mathrm{v} \in V \Rightarrow \mathrm{v} \notin \mathrm{Fv}(\phi) \setminus V \xrightarrow{\text{Property (v)}} \mathrm{T}(\mathrm{v}) \notin \mathrm{T}(\mathrm{Fv}(\phi) \setminus V) \tag{B.5}
$$

With B.5 available, applying a substitution to T10 gives the following:

$$
\begin{aligned}
&\mathrm{T}_V(\xi,[\phi]\mathrm{p})[\mathrm{T}(\mathrm{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{T10}}{=} \quad (\textstyle\sum_{\mathrm{T}(\mathrm{Fv}(\phi) \setminus V)} \mathrm{T}_\xi(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{V \cup \mathrm{Fv}(\phi)}(\xi,\mathrm{p}))[\mathrm{T}(\mathrm{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{B.5}}{=} \quad \textstyle\sum_{\mathrm{T}(\mathrm{Fv}(\phi) \setminus V)} \mathrm{T}_\xi(\phi)[\mathrm{T}(\mathrm{v}) := t_{\mathrm{U}(e)}] \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{V \cup \mathrm{Fv}(\phi)}(\xi,\mathrm{p})[\mathrm{T}(\mathrm{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{Lemma 4.5}}{=} \quad \textstyle\sum_{\mathrm{T}(\mathrm{Fv}(\phi) \setminus V)} \mathrm{T}_{\xi[\mathrm{v}:=e]}(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{V \cup \mathrm{Fv}(\phi)}(\xi,\mathrm{p})[\mathrm{T}(\mathrm{v}) := t_{\mathrm{U}(e)}] \\
&\overset{\text{I.H.}}{=} \quad \textstyle\sum_{\mathrm{T}(\mathrm{Fv}(\phi) \setminus V)} \mathrm{T}_{\xi[\mathrm{v}:=e]}(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{V \cup \mathrm{Fv}(\phi)}(\xi[\mathrm{v} := e],\mathrm{p}) \\
&\overset{\text{T10}}{=} \quad \mathrm{T}_V(\xi[\mathrm{v} := e],[\phi]\mathrm{p})
\end{aligned}
$$

This proves this particular induction step.

Appendix C

Complete proof of Lemma 4.7

Lemma: Take translation relation $\widetilde{\mathrm{T}}$ from Equation 2.1 and action relation $\mathscr{A}$ from Table 2.10. Then $\widetilde{\mathrm{T}}$ is a strong $\mathscr{A}$-warped simulation up to $\equiv$ of AWN expressions P by mCRL2 expressions $\mathrm{T}(\mathrm{P})$ for all AWN expressions P.

Proof: Using Lemma 3.1, for Definition 4.11 to apply it is sufficient to prove that

$$\forall \mathrm{P}, \mathrm{P}', a \,.\, \Big(\mathrm{P} \xrightarrow{a} \mathrm{P}' \Rightarrow \exists (A_1, A_2) \in \mathscr{A} \,.\, a \in A_1 \wedge \mathrm{P} \xrightarrow{A_1} \mathrm{P}' \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_2} \equiv \mathrm{T}(\mathrm{P}') \Big)$$

The proof of Equation 2.18 is by structural induction over the inference rules of AWN.

Induction hypothesis: For all premises $\mathrm{P} \xrightarrow{a} \mathrm{P}'$ of an AWN inference rule, it holds that

$$\exists (A_1, A_2) \in \mathscr{A} \,.\, a \in A_1 \wedge \mathrm{P} \xrightarrow{A_1} \mathrm{P}' \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_2} \equiv \mathrm{T}(\mathrm{P}')$$

1 Base cases

Show for each axiomatic inference rule of AWN with conclusion $\mathrm{P} \xrightarrow{a} \mathrm{P}'$ that there is some $(A_1, A_2) \in \mathscr{A} \,.\, a \in A_1 \wedge \mathrm{P} \xrightarrow{A_1} \mathrm{P}'$ such that $\mathrm{T}(\mathrm{P}) \xrightarrow{a'} \equiv \mathrm{T}(\mathrm{P}')$ can be derived for all $a' \in A_2$.

Broadcast (T1): AWN defines the inference rule

$$\frac{}{\xi, \mathbf{broadcast}(ms).\mathrm{p} \xrightarrow{\mathbf{broadcast}(\xi(ms))} \xi, \mathrm{p}} \textsc{Broadcast (T1)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\mathbf{broadcast}(\xi(ms)) \in A_1 \wedge \xi, \mathbf{broadcast}(ms).\mathrm{p} \xrightarrow{A_1} \xi, \mathrm{p}$, namely Pair 2.4 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 such that $\mathrm{T}(\xi, \mathbf{broadcast}(ms).\mathrm{p}) \xrightarrow{a} \equiv \mathrm{T}(\xi, \mathrm{p})$ for all $a \in A_2$ where

$$A_2 = \left\{ \mathbf{cast}(\llbracket \mathscr{U}_{\mathrm{IP}} \rrbracket, \hat{\mathtt{D}}, \llbracket \mathrm{T}_\xi(ms) \rrbracket) \;\middle|\; \hat{\mathtt{D}} \in \mathrm{T}(\mathrm{Set}(\mathrm{IP})) \right\}$$

In mCRL2, the following derivation can be made for all $\hat{\mathtt{D}} \in \mathtt{T}(\mathrm{Set}(\mathrm{IP}))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathtt{T}_\xi(ms)) \xrightarrow{[\![\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathtt{T}_\xi(ms))]\!]} \checkmark}\ \text{Axiom}}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathtt{T}_\xi(ms)) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \checkmark}\ \text{Definition 2.1}}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, t_{\hat{\mathtt{D}}}, \mathtt{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{Seq 1}}{(\mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathtt{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}))[\mathtt{D} := t_{\hat{\mathtt{D}}}] \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{Substitution}}{\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set}(\mathrm{IP}))} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}}, \mathtt{D}, \mathtt{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{Sum 2}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{broadcast}(ms).\mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{T1}}{\mathrm{T}(\xi, \mathbf{broadcast}(ms).\mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathrm{T}(\xi, \mathrm{p})}\ \text{T12}$$

for $\mathtt{D} \notin \mathtt{T}(V)$. In conclusion, the induction hypothesis holds for this base case.

Groupcast (T1): AWN defines the inference rule

$$\dfrac{}{\xi, \mathbf{groupcast}(dests, ms).\mathrm{p} \xrightarrow{\mathbf{groupcast}(\xi(dests), \xi(ms))} \xi, \mathrm{p}}\ \text{Groupcast (T1)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\mathbf{groupcast}(\xi(dests), \xi(ms)) \in A_1 \wedge \xi, \mathbf{groupcast}(ms).\mathrm{p} \xrightarrow{A_1} \xi, \mathrm{p}$, namely Pair 2.5 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 such that $\mathrm{T}(\xi, \mathbf{groupcast}(ms).\mathrm{p}) \xrightarrow{a} \equiv \mathrm{T}(\xi, \mathrm{p})$ for all $a \in A_2 = \left\{ \mathbf{cast}([\![\mathrm{T}_\xi(dests)]\!], \hat{\mathrm{D}}, [\![\mathrm{T}_\xi(ms)]\!]) \;\middle|\; \hat{\mathtt{D}} \in \mathtt{T}(\mathrm{Set}(\mathrm{IP})) \right\}$.
In mCRL2, the following derivation can be made for all $\hat{\mathtt{D}} \in \mathtt{T}(\mathrm{Set}(\mathrm{IP}))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{cast}(\mathrm{T}_\xi(dests), t_{\hat{\mathtt{D}}}, \mathtt{T}_\xi(ms)) \xrightarrow{[\![\mathbf{cast}(\mathrm{T}_\xi(dests), t_{\hat{\mathtt{D}}}, \mathrm{T}_\xi(ms))]\!]} \checkmark}\ \text{Axiom}}{\mathbf{cast}(\mathrm{T}_\xi(dests), t_{\hat{\mathtt{D}}}, \mathtt{T}_\xi(ms)) \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(dests)]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \checkmark}\ \text{Definition 2.1}}{\mathbf{cast}(\mathrm{T}_\xi(dests), t_{\hat{\mathtt{D}}}, \mathtt{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(dests)]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{Seq 1}}{(\mathbf{cast}(\mathrm{T}_\xi(dests), \mathtt{D}, \mathtt{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}))[\mathtt{D} := t_{\hat{\mathtt{D}}}] \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(dests)]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{Substitution}}{\sum_{\mathtt{D}:\mathtt{T}(\mathrm{Set}(\mathrm{IP}))} \mathbf{cast}(\mathrm{T}_\xi(dests), \mathtt{D}, \mathtt{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(dests)]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{Sum 2}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{groupcast}(dests, ms).\mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(dests)]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \text{T2}}{\mathrm{T}(\xi, \mathbf{groupcast}(dests, ms).\mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(dests)]\!], \hat{\mathtt{D}}, [\![\mathtt{T}_\xi(ms)]\!])} \mathrm{T}(\xi, \mathrm{p})}\ \text{T12}$$

for $\mathtt{D} \notin \mathtt{T}(V)$. In conclusion, the induction hypothesis holds for this base case.

Unicast (T1-1): AWN defines the inference rule

$$\frac{}{\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q} \xrightarrow{\mathbf{unicast}(\xi(dest),\xi(ms))} \xi, \mathrm{p}}\ \textsc{Unicast (T1-1)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\mathbf{unicast}(\xi(dest), \xi(ms)) \in A_1 \wedge \xi, \mathbf{unicast}(dest, ms).\mathrm{p} \xrightarrow{A_1} \xi, \mathrm{p}$, namely Pair 2.6 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 such that $\mathrm{T}(\xi, \mathbf{unicast}(dest, ms).\mathrm{p}) \xrightarrow{a} \equiv \mathrm{T}(\xi, \mathrm{p})$ for all $a \in A_2 = \{\, \mathbf{cast}(\llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \mathrm{T}_\xi(ms) \rrbracket) \,\}$.
In mCRL2, the following derivation can be made:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{cast}(\mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(ms)) \xrightarrow{\llbracket \mathbf{cast}(\mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(ms)) \rrbracket} \checkmark}\ \textsc{Axiom}}{\mathbf{cast}(\mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(ms)) \xrightarrow{\mathbf{cast}(\llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \mathrm{T}_\xi(ms) \rrbracket)} \checkmark}\ \text{Definition 2.1}}{\mathbf{cast}(\mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{\mathbf{cast}(\llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \mathrm{T}_\xi(ms) \rrbracket)} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \textsc{Seq 1}}{\mathbf{cast}(\mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) + \neg\mathbf{uni}(\mathrm{T}_\xi(\{dest\}), \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q}) \xrightarrow{\mathbf{cast}(\llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \mathrm{T}_\xi(ms) \rrbracket)} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \textsc{Choice 2}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q}) \xrightarrow{\mathbf{cast}(\llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \mathrm{T}_\xi(ms) \rrbracket)} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\ \textsc{T3}}{\mathrm{T}(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q}) \xrightarrow{\mathbf{cast}(\llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \mathrm{T}_\xi(ms) \rrbracket)} \mathrm{T}(\xi, \mathrm{p})}\ \textsc{T12}$$

In conclusion, the induction hypothesis holds for this base case.

Unicast (T1-2): AWN defines the inference rule

$$\frac{}{\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q} \xrightarrow{\neg\mathbf{unicast}(\xi(dest),\xi(ms))} \xi, \mathrm{q}}\ \textsc{Unicast (T1-2)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\neg\mathbf{unicast}(\xi(dest), \xi(ms)) \in A_1 \wedge \xi, \mathbf{unicast}(dest, ms).\mathrm{p} \xrightarrow{A_1} \xi, \mathrm{p}$, namely Pair 2.7 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 such that $\mathrm{T}(\xi, \mathbf{unicast}(dest, ms).\mathrm{p}) \xrightarrow{a} \mathrm{T}(\xi, \mathrm{p})$ for all $a \in A_2 = \{\, \neg\mathbf{uni}(\llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \emptyset \rrbracket, \llbracket \mathrm{T}_\xi(ms) \rrbracket) \,\}$.
In mCRL2, the following derivation can be made:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg\mathbf{uni}(\mathrm{T}_\xi(\{dest\}), \emptyset, \mathrm{T}_\xi(ms)) \xrightarrow{\llbracket \neg\mathbf{uni}(\mathrm{T}_\xi(\{dest\}), \emptyset, \mathrm{T}_\xi(ms)) \rrbracket} \checkmark}\ \textsc{Axiom}}{\neg\mathbf{uni}(\mathrm{T}_\xi(\{dest\}), \emptyset, \mathrm{T}_\xi(ms)) \xrightarrow{\neg\mathbf{uni}(\llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \emptyset \rrbracket, \llbracket \mathrm{T}_\xi(ms) \rrbracket)} \checkmark}\ \text{Definition 2.1}}{\neg\mathbf{uni}(\mathrm{T}_\xi(\{dest\}), \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q}) \xrightarrow{\neg\mathbf{uni}(\llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \emptyset \rrbracket, \llbracket \mathrm{T}_\xi(ms) \rrbracket)} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q})}\ \textsc{Seq 1}}{\mathbf{cast}(\mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) + \neg\mathbf{uni}(\mathrm{T}_\xi(\{dest\}), \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q}) \xrightarrow{\neg\mathbf{uni}(\llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \emptyset \rrbracket, \llbracket \mathrm{T}_\xi(ms) \rrbracket)} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q})}\ \textsc{Choice 4}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q}) \xrightarrow{\neg\mathbf{uni}(\llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \emptyset \rrbracket, \llbracket \mathrm{T}_\xi(ms) \rrbracket)} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q})}\ \textsc{T3}}{\mathrm{T}(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q}) \xrightarrow{\neg\mathbf{uni}(\llbracket \mathrm{T}_\xi(\{dest\}) \rrbracket, \llbracket \emptyset \rrbracket, \llbracket \mathrm{T}_\xi(ms) \rrbracket)} \mathrm{T}(\xi, \mathrm{q})}\ \textsc{T12}$$

In conclusion, the induction hypothesis holds for this base case.

Send (T1): AWN defines the inference rule

$$\frac{}{\xi, \mathbf{send}(ms).\mathrm{p} \xrightarrow{\mathbf{send}(\xi(ms))} \xi, \mathrm{p}}\ \textsc{Send (T1)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\mathbf{send}(\xi(ms)) \in A_1 \wedge \xi, \mathbf{send}(ms).\mathrm{p} \xrightarrow{A_1} \xi, \mathrm{p}$, namely Pair 2.8 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 such that $\mathrm{T}(\xi, \mathbf{send}(ms).\mathrm{p}) \xrightarrow{a} \equiv \mathrm{T}(\xi, \mathrm{p})$ for all $a \in A_2 = \left\{ \mathbf{send}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \llbracket \mathrm{T}_\xi(ms)\rrbracket) \right\}$.
In mCRL2, the following derivation can be made:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{send}(\emptyset,\emptyset,\mathrm{T}_\xi(dest)) \xrightarrow{\llbracket\mathbf{send}(\emptyset,\emptyset,\mathrm{T}_\xi(ms))\rrbracket} \checkmark}\ \textsc{Axiom}}{\mathbf{send}(\emptyset,\emptyset,\mathrm{T}_\xi(dest)) \xrightarrow{\mathbf{send}(\llbracket\emptyset\rrbracket,\llbracket\emptyset\rrbracket,\llbracket\mathrm{T}_\xi(ms)\rrbracket)} \checkmark}\ \text{Definition 2.1}}{\mathbf{send}(\emptyset,\emptyset,\mathrm{T}_\xi(dest)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{q}) \xrightarrow{\mathbf{send}(\llbracket\emptyset\rrbracket,\llbracket\emptyset\rrbracket,\llbracket\mathrm{T}_\xi(ms)\rrbracket)} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\ \textsc{Seq 1}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathbf{send}(ms).\mathrm{p}) \xrightarrow{\mathbf{send}(\llbracket\emptyset\rrbracket,\llbracket\emptyset\rrbracket,\llbracket\mathrm{T}_\xi(ms)\rrbracket)} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\ \text{T4}}{\mathrm{T}(\xi,\mathbf{send}(ms).\mathrm{p}) \xrightarrow{\mathbf{send}(\llbracket\emptyset\rrbracket,\llbracket\emptyset\rrbracket,\llbracket\mathrm{T}_\xi(ms)\rrbracket)} \mathrm{T}(\xi,\mathrm{p})}\ \text{T12}$$

In conclusion, the induction hypothesis holds for this base case.

Deliver (T1): AWN defines the inference rule

$$\frac{}{\xi, \mathbf{deliver}(data).\mathrm{p} \xrightarrow{\mathbf{deliver}(\xi(data))} \xi, \mathrm{p}}\ \textsc{Deliver (T1)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\mathbf{deliver}(\xi(data)) \in A_1 \wedge \xi, \mathbf{deliver}(data).\mathrm{p} \xrightarrow{A_1} \xi, \mathrm{p}$, namely Pair 2.9 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 such that $\mathrm{T}(\xi, \mathbf{deliver}(data).\mathrm{p}) \xrightarrow{a} \equiv \mathrm{T}(\xi, \mathrm{p})$ for all $a \in A_2 = \left\{ \mathbf{del}(\hat{\mathtt{i}}, \llbracket \mathrm{T}_\xi(data)\rrbracket) \;\middle|\; \hat{\mathtt{i}} \in \mathrm{T}(\mathrm{IP}) \right\}$.
In mCRL2, the following derivation can be made for all $\hat{\mathtt{i}} \in \mathrm{T}(\mathrm{IP})$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{del}(t_{\hat{\mathtt{i}}},\mathrm{T}_\xi(data)) \xrightarrow{\llbracket\mathbf{del}(t_{\hat{\mathtt{i}}},\mathrm{T}_\xi(data))\rrbracket} \checkmark}\ \textsc{Axiom}}{\mathbf{del}(t_{\hat{\mathtt{i}}},\mathrm{T}_\xi(data)) \xrightarrow{\mathbf{del}(\hat{\mathtt{i}},\llbracket\mathrm{T}_\xi(data)\rrbracket)} \checkmark}\ \text{Definition 2.1}}{\mathbf{del}(t_{\hat{\mathtt{i}}},\mathrm{T}_\xi(data)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{q}) \xrightarrow{\mathbf{del}(\hat{\mathtt{i}},\llbracket\mathrm{T}_\xi(data)\rrbracket)} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\ \textsc{Seq 1}}{\mathbf{del}(\mathtt{ip},\mathrm{T}_\xi(data)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{q})\,[\mathtt{ip} := t_{\hat{\mathtt{i}}}] \xrightarrow{\mathbf{del}(\hat{\mathtt{i}},\llbracket\mathrm{T}_\xi(data)\rrbracket)} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\ \text{Substitution}}{\sum_{\mathtt{ip}:\mathrm{T}(\mathrm{IP})} \mathbf{del}(\mathtt{ip},\mathrm{T}_\xi(data)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{q}) \xrightarrow{\mathbf{del}(\hat{\mathtt{i}},\llbracket\mathrm{T}_\xi(data)\rrbracket)} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\ \textsc{Sum 2}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathbf{deliver}(ms).\mathrm{p}) \xrightarrow{\mathbf{del}(\hat{\mathtt{i}},\llbracket\mathrm{T}_\xi(data)\rrbracket)} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\ \text{T5}}{\mathrm{T}(\xi,\mathbf{deliver}(ms).\mathrm{p}) \xrightarrow{\mathbf{del}(\hat{\mathtt{i}},\llbracket\mathrm{T}_\xi(data)\rrbracket)} \mathrm{T}(\xi,\mathrm{p})}\ \text{T12}$$

for $\mathtt{ip} \notin \mathrm{T}(V)$. In conclusion, the induction hypothesis holds for this base case.

Receive (T1): AWN defines the inference rule

$$\forall m \in \mathrm{MSG} \ \frac{}{\xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p} \xrightarrow{\mathbf{receive}(m)} \xi[\mathtt{msg} := m], \mathrm{p}} \ \textsc{Receive (T1)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\mathbf{receive}(m) \in A_1 \wedge \xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p} \xrightarrow{A_1} \xi[\mathtt{msg} := m], \mathrm{p}$, namely Pair 2.10 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 such that $\mathrm{T}(\xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p}) \xrightarrow{a} \equiv \mathrm{T}(\xi[\mathtt{msg} := m], \mathrm{p})$ for all $a \in A_2 = \left\{ \mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m)) \;\middle|\; \hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'} \in \mathrm{T}(\mathrm{Set(IP)}) \right\}$.
In mCRL2, the following derivation can be made for all $\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'} \in \mathrm{T}(\mathrm{Set(IP)})$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{receive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, t_{\mathrm{U}(m)}) \xrightarrow{[\![\mathbf{receive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, t_{\mathrm{U}(m)})]\!]} \checkmark}\ \textsc{Axiom}}{\mathbf{receive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, t_{\mathrm{U}(m)}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, [\![t_{\mathrm{U}(m)}]\!])} \checkmark}\ \text{Definition 2.1}}{\mathbf{receive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, t_{\mathrm{U}(m)}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \checkmark}\ \text{Definition of } t}{\mathbf{receive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, t_{\mathrm{U}(m)}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg} := m], \mathrm{p}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg} := m], \mathrm{p})}\ \textsc{Seq 1}}{\mathtt{msg} \notin \mathrm{DOM}(\xi^{\setminus \mathtt{msg}})\quad \mathbf{receive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, t_{\mathrm{U}(m)}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi^{\setminus \mathtt{msg}}[\mathtt{msg} := m], \mathrm{p}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg} := m], \mathrm{p})}{\left(\mathbf{receive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, \mathrm{T}(\mathtt{msg})).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi^{\setminus \mathtt{msg}}, \mathrm{p})\right)[\mathrm{T}(\mathtt{msg}) := t_{\mathrm{U}(m)}] \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg} := m], \mathrm{p})}\ \text{Lemma 4.6}}{\left(\mathbf{receive}(\mathtt{D}, \mathtt{D}\mathtt{'}, \mathrm{T}(\mathtt{msg})).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi^{\setminus \mathtt{msg}}, \mathrm{p})\right)[\mathtt{D} := t_{\hat{\mathtt{D}}}, \mathtt{D}\mathtt{'} := t_{\hat{\mathtt{D}}}, \mathrm{T}(\mathtt{msg}) := t_{\mathrm{U}(m)}] \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg} := m], \mathrm{p})}\ \text{Substitution}}{\sum_{\mathtt{D},\mathtt{D}\mathtt{'}:\mathrm{T}(\mathrm{Set(IP)}),\mathrm{T}(\mathtt{msg}):\mathrm{T}(\mathrm{MSG})} \mathbf{receive}(\mathtt{D}, \mathtt{D}\mathtt{'}, \mathrm{T}(\mathtt{msg})).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi^{\setminus \mathtt{msg}}, \mathrm{p}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg} := m], \mathrm{p})}\ \text{SUM 2 (3 times)}}{\dfrac{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg} := m], \mathrm{p})}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi[\mathtt{msg}:=m])}(\xi[\mathtt{msg} := m], \mathrm{p})}\ \text{T12}}{\mathrm{T}(\xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{T}(\xi[\mathtt{msg} := m], \mathrm{p})}\ \text{T6}$$

for $\mathtt{D}, \mathtt{D}\mathtt{'} \notin \mathrm{T}(V)$. In conclusion, the induction hypothesis holds for this base case.

Assignment (T1): AWN defines the inference rule

$$\frac{}{\xi, [\![\mathtt{var} := exp]\!]\mathrm{p} \xrightarrow{\tau} \xi[\mathtt{var} := \xi(exp)], \mathrm{p}} \text{ ASSIGNMENT (T1)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\tau \in A_1 \wedge \xi, [\![\mathtt{var} := exp]\!]\mathrm{p} \xrightarrow{A_1} \xi[\mathtt{var} := \xi(exp)], \mathrm{p}$, namely Pair 2.3 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 such that $\mathrm{T}(\xi, [\![\mathtt{var} := exp]\!]\mathrm{p}) \xrightarrow{a} \equiv \mathrm{T}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p})$ for all $a \in A_2 = \{\, \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \,\}$ for some $D, R \in \mathrm{Set(IP)}$ where $R \subseteq D$ and some $m \in \mathrm{MSG}$. In mCRL2, the following derivation can be made:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}) \xrightarrow{[\![\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy})]\!]} \checkmark}\text{ AXIOM}}{\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \checkmark}\text{ Definition 2.1}}{\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p})}\text{ SEQ 1}}{[\![t_{[\![\mathrm{T}_\xi(exp)]\!]} = t_{[\![\mathrm{T}_\xi(exp)]\!]}]\!] = true \quad \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}}[\mathtt{var} := \xi(exp)], \mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p})}\text{ GUARD 2}}{\mathtt{var} \notin \mathrm{DOM}(\xi^{\setminus\mathtt{var}}) \quad (t_{[\![\mathrm{T}_\xi(exp)]\!]} = t_{[\![\mathrm{T}_\xi(exp)]\!]}) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}}[\mathtt{var} := \xi(exp)], \mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p})}\text{ Lemma 4.6}}{(t_{[\![\mathrm{T}_\xi(exp)]\!]} = t_{[\![\mathrm{T}_\xi(exp)]\!]}) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}}, \mathrm{p})\,[\mathrm{T}(\mathtt{var}) := t_{\mathrm{U}(\xi(exp))}] \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p})}\text{ SUM 2}}{\Big((\mathrm{T}(\mathtt{var}) = t_{[\![\mathrm{T}_\xi(exp)]\!]}) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}}, \mathrm{p})\Big)\,\Big[\mathrm{T}(\mathtt{var}) := t_{[\![\mathrm{T}_\xi(exp)]\!]}\Big] \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p})}\text{ SUM 2}}{[\![t_{[\![\mathrm{T}_\xi(exp)]\!]} = \mathrm{T}_\xi(exp)]\!] = true \quad \textstyle\sum_{\mathrm{T}(\mathtt{var}):\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathrm{T}(\mathtt{var}) = t_{[\![\mathrm{T}_\xi(exp)]\!]}) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}}, \mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p})}\text{ GUARD 2}}{(t_{[\![\mathrm{T}_\xi(exp)]\!]} = \mathrm{T}_\xi(exp)) \to \textstyle\sum_{\mathrm{T}(\mathtt{var}):\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathrm{T}(\mathtt{var}) = t_{[\![\mathrm{T}_\xi(exp)]\!]}) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}}, \mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p})}\text{ SUM 2}}{\Big((\mathtt{tmp} = \mathrm{T}_\xi(exp)) \to \textstyle\sum_{\mathrm{T}(\mathtt{var}):\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathrm{T}(\mathtt{var}) = \mathtt{tmp}) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}}, \mathrm{p})\Big)\,\Big[\mathtt{tmp} := t_{[\![\mathrm{T}_\xi(exp)]\!]}\Big] \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p})}\text{ SUM 2}}{\textstyle\sum_{\mathtt{tmp}:\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathtt{tmp} = \mathrm{T}_\xi(exp)) \to \textstyle\sum_{\mathrm{T}(\mathtt{var}):\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathrm{T}(\mathtt{var}) = \mathtt{tmp}) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}}, \mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p})}\text{ T7}}{\dfrac{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, [\![\mathtt{var} := exp]\!]\mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p})}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, [\![\mathtt{var} := exp]\!]\mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi[\mathtt{var}:=\xi(exp)])}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p})}\text{ T12}}{\mathrm{T}(\xi, [\![\mathtt{var} := exp]\!]\mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}(\xi[\mathtt{var} := \xi(exp)], \mathrm{p})}$$

for $\mathtt{tmp} \notin \mathrm{T}(V)$. In conclusion, the induction hypothesis holds for this base case.

Guard (T1): AWN defines the inference rule

$$\frac{\zeta = \xi[\mathrm{q}_1 := e_1, \cdots, \mathrm{q}_n := e_n]}{\xi, [\phi]\mathrm{p} \xrightarrow{\tau} \zeta, \mathrm{p}} \zeta(\phi) = \mathit{true} \wedge \{\mathrm{q}_1, \cdots, \mathrm{q}_n\} = \mathrm{FV}(\phi) \setminus \mathrm{DOM}(\xi) \ \text{GUARD (T1)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\tau \in A_1 \wedge \xi, [\phi]\mathrm{p} \xrightarrow{A_1} \zeta, \mathrm{p}$, namely Pair 2.3 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 such that $\mathrm{T}(\xi, [\phi]\mathrm{p}) \xrightarrow{a} \equiv \mathrm{T}(\zeta, \mathrm{p})$ for all $a \in A_2 = \{ \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \}$ for some $D, R \in \mathrm{Set(IP)}$ where $R \subseteq D$ and some $m \in \mathrm{MSG}$.
First, the side condition of the inference rule is used to determine that

$$\zeta(\phi) = \mathit{true} \xrightarrow{\text{Property (ii) of T}} [\![\mathrm{T}_\zeta(\phi)]\!] = \mathit{true}$$

With this side condition, the following derivation can be made in mCRL2:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{t}(\emptyset,\emptyset,\mathrm{msg}_{\,dummy}) \xrightarrow{[\mathbf{t}(\emptyset,\emptyset,\mathrm{msg}_{\,dummy})]} \checkmark}\text{Axiom}}{\mathbf{t}(\emptyset,\emptyset,\mathrm{msg}_{\,dummy}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathrm{msg}_{\,dummy}])} \checkmark}\text{Definition 2.1}}{[\![\mathrm{T}_\zeta(\phi)]\!] = \mathit{true} \quad \mathbf{t}(\emptyset,\emptyset,\mathrm{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\zeta,\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathrm{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\zeta,\mathrm{p})}\text{Seq 1}}{\mathrm{T}_\zeta(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathrm{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\zeta,\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathrm{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\zeta,\mathrm{p})}\text{Guard 2}}{\mathrm{T}_{\xi[\mathrm{q}_1:=e_1,\cdots,\mathrm{q}_n:=e_n]}(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathrm{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\xi[\mathrm{q}_1 := e_1,\cdots,\mathrm{q}_n := e_n],\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathrm{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}\text{Definition of } \zeta \text{ (3 times)}}{\{\mathrm{q}_1,\cdots,\mathrm{q}_n\} \cap \mathrm{DOM}(\xi) = \emptyset \quad \mathrm{T}_\xi(\phi)[\mathrm{T}(\mathrm{q}_1) := t_{\mathrm{U}(e_1)},\cdots,\mathrm{T}(\mathrm{q}_n) := t_{\mathrm{U}(e_n)}] \to \mathbf{t}(\emptyset,\emptyset,\mathrm{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\xi[\mathrm{q}_1 := e_1,\cdots,\mathrm{q}_n := e_n],\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathrm{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}\text{Lemma 4.5}}{\{\mathrm{q}_1,\cdots,\mathrm{q}_n\} \cap \mathrm{DOM}(\xi) = \emptyset \quad \mathrm{T}_\xi(\phi)[\mathrm{T}(\mathrm{q}_1) := t_{\mathrm{U}(e_1)},\cdots,\mathrm{T}(\mathrm{q}_n) := t_{\mathrm{U}(e_n)}] \to \mathbf{t}(\emptyset,\emptyset,\mathrm{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\xi,\mathrm{p})[\mathrm{T}(\mathrm{q}_1) := t_{\mathrm{U}(e_1)},\cdots,\mathrm{T}(\mathrm{q}_n) := t_{\mathrm{U}(e_n)}] \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathrm{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}\text{Lemma 4.6}}{\Big(\mathrm{T}_\xi(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathrm{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\xi,\mathrm{p})\Big)[\mathrm{T}(\mathrm{q}_1) := t_{\mathrm{U}(e_1)},\cdots,\mathrm{T}(\mathrm{q}_n) := t_{\mathrm{U}(e_n)}] \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathrm{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}\text{Sum 2 } (n \text{ times})}{\sum_{\mathrm{T}(\{\mathrm{q}_1,\cdots,\mathrm{q}_n\})} \mathrm{T}_\xi(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathrm{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\xi,\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathrm{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}}{\sum_{\mathrm{T}(\mathrm{FV}(\phi)\setminus\mathrm{DOM}(\xi))} \mathrm{T}_\xi(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathrm{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup(\mathrm{FV}(\phi)\setminus\mathrm{DOM}(\xi))}(\xi,\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathrm{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}\text{Definition of } \{\mathrm{q}_1,\cdots,\mathrm{q}_n\} \text{ (2 times)}}{\sum_{\mathrm{T}(\mathrm{FV}(\phi)\setminus\mathrm{DOM}(\xi))} \mathrm{T}_\xi(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathrm{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\mathrm{FV}(\phi)}(\xi,\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathrm{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}\text{T10}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,[\phi]\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathrm{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}\text{T12}}{\mathrm{T}(\xi,[\phi]\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathrm{msg}_{\,dummy}])} \mathrm{T}(\zeta,\mathrm{p})}$$

In conclusion, the induction hypothesis holds for this base case.

Arrive (T3-2): AWN defines the inference rule

$$\frac{}{ip : \mathrm{P} : R \xrightarrow{\emptyset\neg\{ip\}:\mathbf{arrive}(m)} ip : \mathrm{P} : R}\ \textsc{Arrive (T3-2)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\emptyset\neg\{ip\} : \mathbf{arrive}(m) \in A_1 \wedge ip : \mathrm{P} : R \xrightarrow{A_1} ip : \mathrm{P}' : R$, namely Pair 2.13 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 for $\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R)$ for all $a \in A_2$ where

$$A_2 = \left\{ \mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m)) \;\middle|\; \begin{array}{c} \hat{\mathtt{D}},\hat{\mathtt{D}}\mathtt{'} \in \mathrm{T}(\mathrm{Set(IP)}) \\ \hat{\mathtt{D}}\mathtt{'} \subseteq \hat{\mathtt{D}} \\ \{\mathrm{U}(ip)\} \cap \hat{\mathtt{D}}\mathtt{'} = \emptyset \end{array} \right\}$$

Note that $\{\mathrm{U}(ip)\} \cap \hat{\mathtt{D}}\mathtt{'} = \emptyset \Rightarrow [\![t_{\mathrm{U}(ip)} \notin t_{\hat{\mathtt{D}}\mathtt{'}}]\!] = true$
In mCRL2, the following derivation can be made for all $\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'} \in \mathrm{T}(\mathrm{Set(IP)})$ such that $\hat{\mathtt{D}}\mathtt{'} \subseteq \hat{\mathtt{D}} \wedge \mathrm{U}(ip) \notin \hat{\mathtt{D}}\mathtt{'}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{arrive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}\mathtt{'}}, t_{\mathrm{U}(m)}) \xrightarrow{[\![\mathbf{arrive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}\mathtt{'}}, t_{\mathrm{U}(m)})]\!]} \checkmark}\ \textsc{Axiom}}{\mathbf{arrive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}\mathtt{'}}, t_{\mathrm{U}(m)}) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, [\![t_{\mathrm{U}(m)}]\!])} \checkmark}\ \text{Definition 2.1}}{\mathbf{arrive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}\mathtt{'}}, t_{\mathrm{U}(m)}) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \checkmark}\ \text{Definition of } t}{\mathbf{arrive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}\mathtt{'}}, t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Seq 1}}{[\![t_{\mathrm{U}(ip)} \notin t_{\hat{\mathtt{D}}\mathtt{'}}]\!] = true \quad (t_{\mathrm{U}(ip)} \notin t_{\hat{\mathtt{D}}\mathtt{'}}) \to \mathbf{arrive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}\mathtt{'}}, t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Guard 2}}{(t_{\mathrm{U}(ip)} \notin t_{\hat{\mathtt{D}}\mathtt{'}}) \to \mathbf{arrive}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}\mathtt{'}}, t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Substitution}}{\big((t_{\mathrm{U}(ip)} \notin \mathtt{D}\mathtt{'}) \to \mathbf{arrive}(\mathtt{D}, \mathtt{D}\mathtt{'}, \mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \big)[\mathtt{D} := t_{\hat{\mathtt{D}}}, \mathtt{D}\mathtt{'} := t_{\hat{\mathtt{D}}\mathtt{'}}, \mathtt{msg} := t_{\mathrm{U}(m)}] \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Sum 2}\ \ (3 \text{ times})}{\sum_{\mathtt{D},\mathtt{D}\mathtt{'}:\mathrm{T}(\mathrm{Set(IP)}),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (t_{\mathrm{U}(ip)} \notin \mathtt{D}\mathtt{'}) \to \mathbf{arrive}(\mathtt{D}, \mathtt{D}\mathtt{'}, \mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Choice 2}}{\sum_{\mathtt{D},\mathtt{D}\mathtt{'}:\mathrm{T}(\mathrm{Set(IP)}),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (t_{\mathrm{U}(ip)} \notin \mathtt{D}\mathtt{'}) \to \mathbf{arrive}(\mathtt{D}, \mathtt{D}\mathtt{'}, \mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) + \mathrm{S}[\mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)}] \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Substitution}}{\text{Definition of G}\ \dfrac{\big(\sum_{\mathtt{D},\mathtt{D}\mathtt{'}:\mathrm{T}(\mathrm{Set(IP)}),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (\mathtt{ip} \notin \mathtt{D}\mathtt{'}) \to \mathbf{arrive}(\mathtt{D}, \mathtt{D}\mathtt{'}, \mathtt{msg}).\mathrm{G}(\mathtt{ip}, \mathtt{R}) + \mathrm{S} \big)[\mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)}] \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}{\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Recursion 2}}{\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Par 5}}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\gamma_C(\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m)))} \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \text{Apply } \gamma_C}{\mathbf{arrive} \in V \cup \{\tau\} \quad \nabla_V\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \nabla_V\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{Allow 2}}{\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m))} \mathrm{T}(ip : \mathrm{P} : R)}\ \textsc{T14}$$

where S is an expression equal to all summands of G except for the one containing **arrive**. So it is indeed the case that

$$\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P} : R) \text{ for all } a \in A_2 = \left\{ \mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathrm{U}(m)) \;\middle|\; \begin{array}{c} \hat{\mathtt{D}},\hat{\mathtt{D}}\mathtt{'} \in \mathrm{T}(\mathrm{Set(IP)}) \\ \hat{\mathtt{D}}\mathtt{'} \subseteq \hat{\mathtt{D}} \\ \{\mathrm{U}(ip)\} \cap \hat{\mathtt{D}}\mathtt{'} = \emptyset \end{array} \right\}$$

which finishes this particular base case.

Connect (T3-1): AWN defines the inference rule

$$\frac{}{ip : \mathrm{P} : R \xrightarrow{\mathbf{connect}(ip,ip')} ip : \mathrm{P} : R \cup \{ip'\}}\ \textsc{Connect (T3-1)}$$

There exist an (A_1, A_2) in $\mathscr{A}$ such that $\mathbf{connect}(ip, ip') \in A_1 \wedge ip : \mathrm{P} : R \xrightarrow{A_1} ip : \mathrm{P}' : R \cup \{ip'\}$, namely Pair 2.14 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 for $\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R \cup \{ip'\})$ for all $a \in A_2 = \{\, \mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip')) \,\}$.
In mCRL2, the following derivation can be made:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{connect}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(ip')}) \xrightarrow{[\![\mathbf{connect}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(ip')})]\!]} \checkmark}\ \textsc{Axiom}}{\mathbf{connect}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(ip')}) \xrightarrow{\mathbf{connect}([\![t_{\mathrm{U}(ip)}]\!], [\![t_{\mathrm{U}(ip')}]\!])} \checkmark}\ \text{Definition 2.1}}{\mathbf{connect}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(ip')}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip'))} \checkmark}\ \text{Definition of } t}{\mathbf{connect}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(ip')}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R \cup \{ip'\})}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip'))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R \cup \{ip'\})})}\ \textsc{Seq 1}}{(\, \mathbf{connect}(t_{\mathrm{U}(ip)}, \mathtt{ip'}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R \cup \{ip'\})}) \,)\,[\, \mathtt{ip'} := t_{\mathrm{U}(ip')} \,] \xrightarrow{\mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip'))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R \cup \{ip'\})})}\ \text{Substitution}}{\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{connect}(t_{\mathrm{U}(ip)}, \mathtt{ip'}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R \cup \{ip'\})}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip'))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R \cup \{ip'\})})}\ \textsc{Sum 2}}{\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{connect}(\mathtt{ip}, \mathtt{ip'}).\mathrm{G}(\mathtt{ip}, \mathtt{R} \cup \{ip'\}) + \mathrm{S}[\, \mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)} \,] \xrightarrow{\mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip'))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R \cup \{ip'\})})}\ \textsc{Choice 2}}{(\, \sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{connect}(\mathtt{ip}, \mathtt{ip'}).\mathrm{G}(\mathtt{ip}, \mathtt{R} \cup \{ip'\}) + \mathrm{S} \,)\,[\, \mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)} \,] \xrightarrow{\mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip'))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R \cup \{ip'\})})}\ \text{Substitution}}{\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip'))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R \cup \{ip'\})})}\ \text{Definition of G}\ \textsc{Recursion 2}}{\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip'))} \mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R \cup \{ip'\})})}\ \textsc{Par 5}}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\gamma_C(\mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip')))} \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R \cup \{ip'\})}))}\ \textsc{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip'))} \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R \cup \{ip'\})}))}\ \text{Apply } \gamma_C}{\mathbf{connect} \in V \cup \{\tau\}}\ \textsc{Allow 2}}{\nabla_V \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip'))} \nabla_V \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R \cup \{ip'\})}))}\ \textsc{T14}}{\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip'))} \mathrm{T}(ip : \mathrm{P} : R \cup \{ip'\})}$$

where S is an expression equal to all summands of G except for the first summand containing **connect**.
So it is indeed the case that

$$\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P} : R \cup \{ip'\}) \text{ for all } a \in A_2 = \{\, \mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip')) \,\}$$

which finishes this particular base case.

Connect (T3-2): Similar to the Lemma 4.7-proof for Connect (T3-1).

Connect (T3-3): AWN defines the inference rule

$$\frac{ip \notin \{ip', ip''\}}{ip : \mathrm{P} : R \xrightarrow{\mathbf{connect}(ip', ip'')} ip : \mathrm{P} : R}\ \textsc{Connect (T3-3)}$$

There exist an (A_1, A_2) in $\mathscr{A}$ such that $\mathbf{connect}(ip', ip'') \in A_1 \wedge ip : \mathrm{P} : R \xrightarrow{A_1} ip : \mathrm{P}' : R$, namely Pair 2.14 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 for $\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R)$ for all $a \in A_2 = \{\, \mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip'')) \,\}$. In mCRL2, the following derivation can be made:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{connect}(t_{\mathrm{U}(ip')}, t_{\mathrm{U}(ip'')}) \xrightarrow{[\![\mathbf{connect}(t_{\mathrm{U}(ip')}, t_{\mathrm{U}(ip'')})]\!]} \checkmark}\ \textsc{Axiom}}{\mathbf{connect}(t_{\mathrm{U}(ip')}, t_{\mathrm{U}(ip'')}) \xrightarrow{\mathbf{connect}([\![t_{\mathrm{U}(ip')}]\!], [\![t_{\mathrm{U}(ip'')}]\!])} \checkmark}\ \text{Definition 2.1}}{\mathbf{connect}(t_{\mathrm{U}(ip')}, t_{\mathrm{U}(ip'')}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))} \checkmark}\ \text{Definition of } t}{\mathbf{connect}(t_{\mathrm{U}(ip')}, t_{\mathrm{U}(ip'')}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Seq 1}}{[\![t_{\mathrm{U}(ip)} \notin \{t_{\mathrm{U}(ip')}, t_{\mathrm{U}(ip'')}\}]\!] = true \quad \mathbf{connect}(t_{\mathrm{U}(ip')}, t_{\mathrm{U}(ip'')}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Guard 2}}{(t_{\mathrm{U}(ip)} \notin \{t_{\mathrm{U}(ip')}, t_{\mathrm{U}(ip'')}\}) \to \mathbf{connect}(t_{\mathrm{U}(ip')}, t_{\mathrm{U}(ip'')}).\mathrm{G}(t_{\mathrm{U}(ip')}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Substitution}}{\big((t_{\mathrm{U}(ip)} \notin \{\mathtt{ip'}, \mathtt{ip''}\}) \to \mathbf{connect}(\mathtt{ip'}, \mathtt{ip''}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})\big)[\,\mathtt{ip'} := t_{\mathrm{U}(ip')}, \mathtt{ip''} := t_{\mathrm{U}(ip'')}\,] \xrightarrow{\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Sum 2}}{\textstyle\sum_{\mathtt{ip'}, \mathtt{ip''}:\mathrm{T}(\mathrm{IP})} (t_{\mathrm{U}(ip)} \notin \{\mathtt{ip'}, \mathtt{ip''}\}) \to \mathbf{connect}(\mathtt{ip'}, \mathtt{ip''}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Choice 2}}{\textstyle\sum_{\mathtt{ip'}, \mathtt{ip''}:\mathrm{T}(\mathrm{IP})} (t_{\mathrm{U}(ip)} \notin \{\mathtt{ip'}, \mathtt{ip''}\}) \to \mathbf{connect}(\mathtt{ip'}, \mathtt{ip''}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) + \mathrm{S}[\,\mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)}\,] \xrightarrow{\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Substitution}}{\text{Definition of G}\ \big(\textstyle\sum_{\mathtt{ip'}, \mathtt{ip''}:\mathrm{T}(\mathrm{IP})} (\mathtt{ip} \notin \{\mathtt{ip'}, \mathtt{ip''}\}) \to \mathbf{connect}(\mathtt{ip'}, \mathtt{ip''}).\mathrm{G}(\mathtt{ip}, \mathtt{R}) + \mathrm{S}\big)[\,\mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)}\,] \xrightarrow{\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Recursion 2}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Par 5}}{\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))} \mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\gamma_C(\mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip'')))} \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \text{Apply } \gamma_C}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))} \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{Allow 2}}{\mathbf{connect} \in V \cup \{\tau\} \quad \nabla_V\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))} \nabla_V\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{T14}}{\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))} \mathrm{T}(ip : \mathrm{P} : R)}$$

where S is an expression equal to all summands of G except for the third summand containing **connect**. So it is indeed the case that

$$\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P} : R) \text{ for all } a \in A_2 = \{\, \mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip'')) \,\} \text{ with } ip \notin \{ip', ip''\}$$

which finishes this particular base case.

Disconnect (T3-1): Similar to the Lemma 4.7-proof for Connect (T3-1).

Disconnect (T3-2): Similar to the Lemma 4.7-proof for Connect (T3-1).

Disconnect (T3-3): Similar to the Lemma 4.7-proof for Connect (T3-3).

2 Induction step

Given only its side conditions and the induction hypothesis, show for each inference rule of AWN with conclusion $\mathrm{P} \xrightarrow{a} \mathrm{P}'$ that there is some $(A_1, A_2) \in \mathscr{A} \,.\, a \in A_1 \wedge \mathrm{P} \xrightarrow{A_1} \mathrm{P}'$ such that $\mathrm{T}(\mathrm{P}) \xrightarrow{a'} \equiv \mathrm{T}(\mathrm{P}')$ can be derived for all $a' \in A_2$.

Recursion (T1): AWN defines the inference rule

$$\forall a \in \mathrm{Act}\ \frac{\emptyset[\mathtt{var}_i := \xi(exp_i)]_{i=1}^n, \mathrm{p} \xrightarrow{a} \zeta, \mathrm{p}' \qquad \mathrm{X}(\mathtt{var}_1, \cdots, \mathtt{var}_n) \overset{\text{def}}{=} \mathrm{p}}{\xi, \mathrm{X}(exp_1, \cdots, exp_n) \xrightarrow{a} \zeta, \mathrm{p}'}\ \textsc{Recursion (T1)}$$

According to the induction hypothesis,

$$\exists (A_1, A_2) \in \mathscr{A} \,.\, a \in A_1 \wedge \emptyset[\mathtt{var}_i := \xi(exp_i)]_{i=1}^n, \mathrm{p} \xrightarrow{A_1} \zeta, \mathrm{p}' \wedge \mathrm{T}(\emptyset[\mathtt{var}_i := \xi(exp_i)]_{i=1}^n, \mathrm{p}) \xrightarrow{A_2} \equiv \mathrm{T}(\zeta, \mathrm{p}')$$

and so the following derivation can be made for all $a' \in A_2$:

$$\mathrm{X}(\mathtt{var}_1, \cdots, \mathtt{var}_n) \overset{\text{def}}{=} \mathrm{p},\ \text{T11}\ \frac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\emptyset[\mathtt{var}_i := \xi(exp_i)]_{i=1}^n, \mathrm{p}) \xrightarrow{a'} \equiv \mathrm{T}(\zeta, \mathrm{p}')}\ \text{Induction hypothesis}}{\mathrm{T}_{\{\mathtt{var}_1, \cdots, \mathtt{var}_n\}}(\emptyset[\mathtt{var}_i := \xi(exp_i)]_{i=1}^n, \mathrm{p}) \xrightarrow{a'} \equiv \mathrm{T}(\zeta, \mathrm{p}')}\ \text{T12}}{\mathrm{T}_{\{\mathtt{var}_1, \cdots, \mathtt{var}_n\}}(\emptyset, \mathrm{p}) \left[\mathrm{T}(\mathtt{var}_i) := t_{\mathrm{U}(\xi(exp_i))}\right]_{i=1}^n \xrightarrow{a'} \equiv \mathrm{T}(\zeta, \mathrm{p}')}\ \text{Lemma 4.6}}{\mathrm{X}(t_{\mathrm{U}(\xi(exp_1))}, \cdots, t_{\mathrm{U}(\xi(exp_n))}) \xrightarrow{a'} \equiv \mathrm{T}(\zeta, \mathrm{p}')}\ \textsc{Recursion 2}}{\mathrm{X}(\mathrm{T}_\xi(exp_1), \cdots, \mathrm{T}_\xi(exp_n)) \xrightarrow{a'} \equiv \mathrm{T}(\zeta, \mathrm{p}')}\ \text{Theorem 4.1, Lemma 4.3}}{\mathrm{T}_V(\xi, \mathrm{X}(exp_1, \cdots, exp_n)) \xrightarrow{a'} \equiv \mathrm{T}(\zeta, \mathrm{p}')}\ \text{T8}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{X}(exp_1, \cdots, exp_n)) \xrightarrow{a'} \equiv \mathrm{T}(\zeta, \mathrm{p}')}\ \text{Choose } V = \textsc{dom}(\xi)}{\mathrm{T}(\xi, \mathrm{X}(exp_1, \cdots, exp_n)) \xrightarrow{a'} \equiv \mathrm{T}(\zeta, \mathrm{p}')}\ \text{T12}$$

This proves this particular induction step.

Choice (T1-1): AWN defines the inference rule

$$\forall a \in \mathrm{Act}\ \frac{\xi, \mathrm{p} \xrightarrow{a} \zeta, \mathrm{p}'}{\xi, \mathrm{p} + \mathrm{q} \xrightarrow{a} \zeta, \mathrm{p}'}\ \textsc{Choice (T1-1)}$$

According to the induction hypothesis,

$$\exists (A_1, A_2) \in \mathscr{A} \,.\, a \in A_1 \wedge \xi, \mathrm{p} \xrightarrow{A_1} \zeta, \mathrm{p}' \wedge \mathrm{T}(\xi, \mathrm{p}) \xrightarrow{A_2} \equiv \mathrm{T}(\zeta, \mathrm{p}')$$

and so the following derivation can be made for all $a' \in A_2$:

$$\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\xi,\mathrm{p}) \xrightarrow{a'} \equiv \mathrm{T}(\zeta,\mathrm{p}')}\ \text{Induction hypothesis}}{\mathrm{T}(\xi,\mathrm{p})+\mathrm{T}(\xi,\mathrm{q}) \xrightarrow{a'} \equiv \mathrm{T}(\zeta,\mathrm{p}')}\ \textsc{Choice 2}}{\mathrm{T}(\xi,\mathrm{p}+\mathrm{q}) \xrightarrow{a'} \equiv \mathrm{T}(\zeta,\mathrm{p}')}\ \text{T9}$$

This proves this particular induction step.

Choice (T1-2): Similar to the the Lemma 4.7-proof for Choice (T1-1) (CHOICE 4 is used instead of CHOICE 2).

Parallel (T2-1): AWN defines the inference rule

$$\forall a \neq \mathbf{receive}(m)\ \dfrac{\mathrm{P} \xrightarrow{a} \mathrm{P}'}{\mathrm{P} \between \mathrm{Q} \xrightarrow{a} \mathrm{P}' \between \mathrm{Q}}\ \textsc{Parallel (T2-1)}$$

According to the induction hypothesis,

$$\exists (A_1,A_2) \in \mathscr{A}\ .\ a \in A_1 \wedge \mathrm{P} \xrightarrow{A_1} \mathrm{P}' \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_2} \equiv \mathrm{T}(\mathrm{P}')$$

Because $a \neq \mathbf{receive}$, Pair 2.10 from Table 2.10 cannot be among the pairs (A_1,A_2) that satisfy the induction hypothesis (of which there must be at least one). Furthermore, according to the AWN semantics a must be an action that can be performed at the sequential level, meaning that $a \in \{\tau, \mathbf{broadcast}, \mathbf{groupcast}, \mathbf{unicast}, \neg\mathbf{unicast}, \mathbf{send}, \mathbf{deliver}\}$. As a consequence,

$$\forall (A_1,A_2) \in \mathscr{A}\ .\ a \in A_1 \wedge \mathrm{P} \xrightarrow{A_1} \mathrm{P}' \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_2} \equiv \mathrm{T}(\mathrm{P}') \Rightarrow \forall a' \in A_2\ .\ \underline{a'} \in W$$
$$\text{where } W = \{\mathbf{t}, \mathbf{cast}, \neg\mathbf{uni}, \mathbf{send}, \mathbf{del}\}$$

Given this, the following derivation can be made in mCRL2 for all $a' \in A_2$:

$$\underline{a'} \in V \cup \{\tau\} \supseteq W\ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{P}) \xrightarrow{a'} \equiv \mathrm{T}(\mathrm{P}')}\ \text{Induction hypothesis}}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \xrightarrow{\{\mathbf{receive}\to\mathbf{r}\}\bullet a'} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}')}\ \textsc{Rename 2}}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \xrightarrow{a'} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}')}\ \underline{a'} \neq \mathbf{receive}\ \text{Apply } \{\mathbf{receive}\to\mathbf{r}\}\bullet}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{a'} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})}\ \textsc{Par 2}}{\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{\gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(a')} \equiv \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}))}\ \textsc{Comm 2}}{\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{a'} \equiv \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}))}\ \text{Apply } \gamma}{\nabla_V\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{a'} \equiv \nabla_V\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}))}\ \textsc{Allow 2}}{\mathrm{T}(\mathrm{P} \between \mathrm{Q}) \xrightarrow{a'} \equiv \mathrm{T}(\mathrm{P}' \between \mathrm{Q})}\ \text{T13}$$

This proves this particular induction step.

Parallel (T2-2): AWN defines the inference rule

$$\forall a \neq \mathbf{send}(m)\ \frac{\mathrm{Q} \xrightarrow{a} \mathrm{Q}'}{\mathrm{P} \langle\langle \mathrm{Q} \xrightarrow{a} \mathrm{P} \langle\langle \mathrm{Q}'}\ \textsc{Parallel (T2-2)}$$

According to the induction hypothesis,

$$\exists (A_1, A_2) \in \mathscr{A}\ .\ a \in A_1 \wedge \mathrm{Q} \xrightarrow{A_1} \mathrm{Q}' \wedge \mathrm{T}(\mathrm{Q}) \xrightarrow{A_2} \equiv \mathrm{T}(\mathrm{Q}')$$

Because $a \neq \mathbf{send}$, Pair 2.8 from Table 2.10 cannot be among the pairs (A_1, A_2) that satisfy the induction hypothesis (of which there must be at least one). Furthermore, according to the AWN semantics a must be an action that can be performed at the sequential level, meaning that $a \in \{\tau, \mathbf{broadcast}, \mathbf{groupcast}, \mathbf{unicast}, \neg\mathbf{unicast}, \mathbf{deliver}, \mathbf{receive}\}$. As a consequence,

$$\forall (A_1, A_2) \in \mathscr{A}\ .\ a \in A_1 \wedge \mathrm{P} \xrightarrow{A_1} \mathrm{P}' \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_2} \equiv \mathrm{T}(\mathrm{P}') \Rightarrow \forall a' \in A_2\ .\ \underline{a'} \in W$$
$$\text{where } W = \{\mathbf{t}, \mathbf{cast}, \neg\mathbf{uni}, \mathbf{del}, \mathbf{receive}\}$$

Given this, the following derivation can be made in mCRL2 for all $a' \in A_2$:

$$\underline{a'} \in V \cup \{\tau\} \supseteq W\ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{Q}) \xrightarrow{a'} \equiv \mathrm{T}(\mathrm{Q}')}\ \text{Induction hypothesis}}{\rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{\{\mathbf{send}\to\mathbf{s}\}\bullet a'} \equiv \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}')}\ \textsc{Rename 2}}{\rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{a'} \equiv \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}')}\ \underline{a'} \neq \mathbf{send}\ \text{Apply } \{\mathbf{send}\to\mathbf{s}\}\bullet}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{a'} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}')}\ \textsc{Par 5}}{\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{\gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(a')} \equiv \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}'))}\ \textsc{Comm 2}}{\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{a'} \equiv \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}'))}\ \text{Apply } \gamma}{\nabla_V\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{a'} \equiv \nabla_V\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}'))}\ \textsc{Allow 2}}{\mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q}) \xrightarrow{a'} \equiv \mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q}')}\ \textsc{T13}$$

This proves this particular induction step.

Parallel (T2-3): AWN defines the inference rule

$$\forall m \in \mathrm{MSG}\ \frac{\mathrm{P} \xrightarrow{\mathbf{receive}(m)} \mathrm{P}' \qquad \mathrm{Q} \xrightarrow{\mathbf{send}(m)} \mathrm{Q}'}{\mathrm{P} \langle\langle \mathrm{Q} \xrightarrow{\tau} \mathrm{P}' \langle\langle \mathrm{Q}'}\ \textsc{Parallel (T2-3)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\tau \in A_1 \wedge \mathrm{P} \langle\langle \mathrm{Q} \xrightarrow{A_1} \mathrm{P}' \langle\langle \mathrm{Q}'$, namely Pair 2.3 in Table 2.10. The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{P}' \langle\langle \mathrm{Q})$ for all $a \in A_2 = \{\, \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \,\}$ for some D, R where $R \subseteq D$.

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}', \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')$ for all $\hat{\mathtt{D}}, \hat{\mathtt{D}}' \in \mathrm{T}(\mathrm{Set(IP)})$ because Pair 2.10 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $\mathbf{receive}(m) \in B_1$. Similarly, $\mathrm{T}(\mathrm{Q}) \xrightarrow{\mathbf{send}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{Q}')$ because Pair 2.8 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $\mathbf{send}(m) \in B_1$ and the following condition holds:

$$\xi(ms) = m \overset{\text{Lemma 4.4}}{\Longleftrightarrow} \llbracket \mathrm{T}_\xi(ms) \rrbracket = \mathrm{U}(m)$$

This means that the following derivation can be made in mCRL2:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}', \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')}\ \text{Induction hypothesis}}{\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{receive}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')}\ \text{Choose } \hat{\mathtt{D}} = \hat{\mathtt{D}}' = \emptyset}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \xrightarrow{\{\mathbf{receive}\to\mathbf{r}\}\bullet\mathbf{receive}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}')}\ \textsc{Rename 2}}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{r}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}')}\ \text{Apply } \{\mathbf{receive}\to\mathbf{r}\}\bullet$$

$$\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{Q}) \xrightarrow{\mathbf{send}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{Q}')}\ \text{Induction hypothesis}}{\rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{\{\mathbf{send}\to\mathbf{s}\}\bullet\mathbf{send}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))} \equiv \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}')}\ \textsc{Rename 2}}{\rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{\mathbf{s}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))} \equiv \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}')}\ \text{Apply } \{\mathbf{send}\to\mathbf{s}\}\bullet$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{r}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}')}\ \textit{(see above)} \qquad \dfrac{}{\rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{\mathbf{s}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))} \equiv \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}')}\ \textit{(see above)}}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{\mathbf{r}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))|\mathbf{s}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}')}\ \textsc{Par 3}}{\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{\gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\mathbf{r}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))|\mathbf{s}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m)))} \equiv \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}'))}\ \textsc{Comm 2}}{\mathbf{t} \in V \cup \{\tau\}\ \dfrac{\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{\mathbf{rs}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))} \equiv \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}'))}{\nabla_V\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{\mathbf{t}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))} \equiv \nabla_V\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}'))}\ \textsc{Allow 2}}{\mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q}) \xrightarrow{\mathbf{t}(\llbracket\emptyset\rrbracket, \llbracket\emptyset\rrbracket, \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}' \langle\langle \mathrm{Q}')}\ \textsc{T13}$$

This proves this particular induction step.

Broadcast (T3): AWN defines the inference rule

$$\frac{\mathrm{P} \xrightarrow{\mathbf{broadcast}(m)} \mathrm{P}'}{ip : \mathrm{P} : R \xrightarrow{R:\mathbf{*cast}(m)} ip : \mathrm{P}' : R}\ \textsc{Broadcast (T3)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $R : \mathbf{*cast}(m) \in A_1 \wedge ip : \mathrm{P} : R \xrightarrow{A_1} ip : \mathrm{P}' : R$, namely Pair 2.11 in Table 2.10 (note that it is possible to choose $D = \mathscr{U}_{\mathrm{IP}}$). The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R)$ for all $a \in A_2 = \{\, \mathbf{starcast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m)) \,\}$.
In mCRL2, the following derivation can be made:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{\mathbf{cast}}(\mathscr{U}_{\mathrm{IP}}, t_{\mathrm{U}(R)}, t_{\mathrm{U}(m)}) \xrightarrow{[\![\overline{\mathbf{cast}}(\mathscr{U}_{\mathrm{IP}}, t_{\mathrm{U}(R)}, t_{\mathrm{U}(m)})]\!}} \checkmark}\ \textsc{Axiom}}{\overline{\mathbf{cast}}(\mathscr{U}_{\mathrm{IP}}, t_{\mathrm{U}(R)}, t_{\mathrm{U}(m)}) \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], [\![t_{\mathrm{U}(R)}]\!], [\![t_{\mathrm{U}(m)}]\!])} \checkmark}\ \text{Definition 2.1}}{\overline{\mathbf{cast}}(\mathscr{U}_{\mathrm{IP}}, t_{\mathrm{U}(R)}, t_{\mathrm{U}(m)}) \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \checkmark}\ \text{Definition of } t}{\overline{\mathbf{cast}}(\mathscr{U}_{\mathrm{IP}}, t_{\mathrm{U}(R)}, t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Seq 1}}{[\![t_{\mathrm{U}(R)} \cap \mathscr{U}_{\mathrm{IP}} = t_{\mathrm{U}(R)}]\!] = \mathit{true}}\ \textsc{Guard 2}}{(t_{\mathrm{U}(R)} \cap \mathscr{U}_{\mathrm{IP}} = t_{\mathrm{U}(R)}) \rightarrow \overline{\mathbf{cast}}(\mathscr{U}_{\mathrm{IP}}, t_{\mathrm{U}(R)}, t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Substitution}}{\big(\,(t_{\mathrm{U}(R)} \cap \mathtt{D} = \mathtt{D'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D}, \mathtt{D'}, \mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})\,\big)\,[\,\mathtt{D} := \mathscr{U}_{\mathrm{IP}}, \mathtt{D'} := t_{\mathrm{U}(R)}, \mathtt{msg} := t_{\mathrm{U}(m)}\,] \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Sum 2}\ \ (3 \text{ times})}{\textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (t_{\mathrm{U}(R)} \cap \mathtt{D} = \mathtt{D'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D}, \mathtt{D'}, \mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Choice 2}}{\textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (t_{\mathrm{U}(R)} \cap \mathtt{D} = \mathtt{D'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D}, \mathtt{D'}, \mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) + \mathrm{S}[\,\mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)}\,] \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Substitution}}{\text{Definition of G}\ \dfrac{\big(\,\textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (\mathtt{R} \cap \mathtt{D} = \mathtt{D'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D}, \mathtt{D'}, \mathtt{msg}).\mathrm{G}(\mathtt{ip}, \mathtt{R}) + \mathrm{S}\,\big)\,[\,\mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)}\,] \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}{\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Recursion 2}$$

where S is an expression equal to all summands of G except for the first one.

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')$ because Pair 2.4 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $\mathbf{broadcast}(m) \in B_1$. Combining this with the conclusion of the derivation above gives

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')}\ \text{Induction hypothesis} \qquad \dfrac{}{\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textit{(see above)}}{\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m)) | \overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Par 3}}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\gamma_C(\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m)) | \overline{\mathbf{cast}}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{Comm 2}}{\mathbf{starcast} \in V \cup \{\tau\}}\ \dfrac{\dfrac{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{starcast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}{\nabla_V \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{starcast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \equiv \nabla_V \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{Allow 2}}{\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{\mathbf{starcast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m))} \equiv \mathrm{T}(ip : \mathrm{P}' : R)}\ \text{T14}$$

So it is indeed the case that

$$\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R) \text{ for all } a \in A_2 = \{\, \mathbf{starcast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \mathrm{U}(R), \mathrm{U}(m)) \,\}$$

which finishes the induction step for the case of the BROADCAST (T3) inference rule.

Groupcast (T3): AWN defines the inference rule

$$\frac{\mathrm{P} \xrightarrow{\mathbf{groupcast}(D,m)} \mathrm{P}'}{ip:\mathrm{P}:R \xrightarrow{R\cap D:\mathbf{*cast}(m)} ip:\mathrm{P}':R}\ \textsc{Groupcast (T3)}$$

There exists an (A_1,A_2) in $\mathscr{A}$ such that $R\cap D:\mathbf{*cast}(m)\in A_1 \wedge ip:\mathrm{P}:R \xrightarrow{A_1} ip:\mathrm{P}':R$, namely Pair 2.11 in Table 2.10. The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}(ip:\mathrm{P}:R) \xrightarrow{a} \equiv \mathrm{T}(ip:\mathrm{P}':R)$ for all $a\in A_2 = \{\, \mathbf{starcast}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(\ exprm)) \,\}$.
In mCRL2, the following derivation can be made:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{\mathbf{cast}}(t_{\mathrm{U}(D)},t_{\mathrm{U}(R\cap D)},t_{\mathrm{U}(m)}) \xrightarrow{[\![\overline{\mathbf{cast}}(t_{\mathrm{U}(D)},t_{\mathrm{U}(R\cap D)},t_{\mathrm{U}(m)})]\!]} \checkmark}\ \textsc{Axiom}}{\overline{\mathbf{cast}}(t_{\mathrm{U}(D)},t_{\mathrm{U}(R\cap D)},t_{\mathrm{U}(m)}) \xrightarrow{\overline{\mathbf{cast}}([\![t_{\mathrm{U}(D)}]\!],[\![t_{\mathrm{U}(R\cap D)}]\!],[\![t_{\mathrm{U}(m)}]\!])} \checkmark}\ \text{Definition 2.1}}{\overline{\mathbf{cast}}(t_{\mathrm{U}(D)},t_{\mathrm{U}(R\cap D)},t_{\mathrm{U}(m)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \checkmark}\ \text{Definition of } t}{\overline{\mathbf{cast}}(t_{\mathrm{U}(D)},t_{\mathrm{U}(R\cap D)},t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \textsc{Seq 1}}{[\![t_{\mathrm{U}(R)}\cap t_{\mathrm{U}(D)} = t_{\mathrm{U}(R\cap D)}]\!] = \mathit{true}\ \dfrac{}{\overline{\mathbf{cast}}(t_{\mathrm{U}(D)},t_{\mathrm{U}(R\cap D)},t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \textsc{Guard 2}}{(t_{\mathrm{U}(R)}\cap t_{\mathrm{U}(D)} = t_{\mathrm{U}(R\cap D)}) \rightarrow \overline{\mathbf{cast}}(t_{\mathrm{U}(D)},t_{\mathrm{U}(R\cap D)},t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \text{Substitution}}{\big((t_{\mathrm{U}(R)}\cap \mathtt{D}=\mathtt{D'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D},\mathtt{D'},\mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})\big)\,[\,\mathtt{D}:=t_{\mathrm{U}(D)},\mathtt{D'}:=t_{\mathrm{U}(R\cap D)},\mathtt{msg}:=t_{\mathrm{U}(m)}\,] \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \text{Sum 2 (3 times)}}{\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})}(t_{\mathrm{U}(R)}\cap\mathtt{D}=\mathtt{D'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D},\mathtt{D'},\mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \textsc{Choice 2}}{\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})}(t_{\mathrm{U}(R)}\cap\mathtt{D}=\mathtt{D'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D},\mathtt{D'},\mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) + \mathrm{S}[\,\mathtt{ip}:=t_{\mathrm{U}(ip)},\mathtt{R}:=t_{\mathrm{U}(R)}\,] \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \text{Substitution}}{\text{Definition of G}\ \dfrac{\big(\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})}(\mathtt{R}\cap\mathtt{D}=\mathtt{D'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D},\mathtt{D'},\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R}) + \mathrm{S}\big)\,[\,\mathtt{ip}:=t_{\mathrm{U}(ip)},\mathtt{R}:=t_{\mathrm{U}(R)}\,] \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}{\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \textsc{Recursion 2}}$$

where S is an expression equal to all summands of G except for the first one.
From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{cast}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')$ because Pair 2.5 in Table 2.10 is the only $(B_1,B_2)\in\mathscr{A}$. $\mathbf{groupcast}(D,m)\in B_1$. Combining this with the conclusion of the derivation above gives

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{cast}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')}\ \text{Induction hypothesis} \quad \dfrac{}{\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \textit{(see above)}}{\mathrm{T}(\mathrm{P})\,||\,\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{cast}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))|\overline{\mathbf{cast}}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')\,||\,\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \text{Par 3}}{\Gamma_C(\mathrm{T}(\mathrm{P})\,||\,\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})) \xrightarrow{\gamma_C(\mathbf{cast}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))|\overline{\mathbf{cast}}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}')\,||\,\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}))}\ \text{Comm 2}}{\mathbf{starcast}\in V\cup\{\tau\}\ \dfrac{\Gamma_C(\mathrm{T}(\mathrm{P})\,||\,\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{starcast}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}')\,||\,\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}))}{\nabla_V\Gamma_C(\mathrm{T}(\mathrm{P})\,||\,\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{starcast}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \equiv \nabla_V\Gamma_C(\mathrm{T}(\mathrm{P}')\,||\,\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}))}\ \text{T14}}\ \text{Allow 2}}{\mathrm{T}(ip:\mathrm{P}:R) \xrightarrow{\mathbf{starcast}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m))} \equiv \mathrm{T}(ip:\mathrm{P}':R)}\ \text{Apply } \gamma_C$$

So it is indeed the case that

$$\mathrm{T}(ip:\mathrm{P}:R) \xrightarrow{a} \equiv \mathrm{T}(ip:\mathrm{P}':R) \text{ for all } a\in A_2 = \{\, \mathbf{starcast}(\mathrm{U}(D),\mathrm{U}(R\cap D),\mathrm{U}(m)) \,\}$$

which finishes the induction step for the case of the GROUPCAST (T3) inference rule.

Unicast (T3-1): AWN defines the inference rule

$$\frac{\mathrm{P} \xrightarrow{\mathbf{unicast}(dip,m)} \mathrm{P}' \qquad dip \in R}{ip : \mathrm{P} : R \xrightarrow{\{dip\}:\mathbf{*cast}(m)} ip : \mathrm{P}' : R}\ \textsc{Unicast (T3-1)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\{dip\} : \mathbf{*cast}(m) \in A_1 \wedge ip : \mathrm{P} : R \xrightarrow{A_1} ip : \mathrm{P}' : R$, namely Pair 2.11 in Table 2.10. The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R)$ for all $a \in A_2 = \{\, \mathbf{starcast}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m)) \,\}$.
In mCRL2, the following derivation can be made:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{\mathbf{cast}}(t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(m)}) \xrightarrow{[\overline{\mathbf{cast}}(t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(m)})]} \checkmark}\ \textsc{Axiom}}{\overline{\mathbf{cast}}(t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(m)}) \xrightarrow{\overline{\mathbf{cast}}([\![t_{\mathrm{U}(\{dip\})}]\!], [\![t_{\mathrm{U}(R \cap D)}]\!], [\![t_{\mathrm{U}(m)}]\!])} \checkmark}\ \text{Definition 2.1}}{\overline{\mathbf{cast}}(t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(m)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \checkmark}\ \text{Definition of } t}{\overline{\mathbf{cast}}(t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(m)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \checkmark}\ \textsc{Seq 1}}{\overline{\mathbf{cast}}(t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}$$

$$[\![t_{\mathrm{U}(R)} \cap t_{\mathrm{U}(\{dip\})} = t_{\mathrm{U}(\{dip\})}]\!] = true\ \dfrac{(t_{\mathrm{U}(R)} \cap t_{\mathrm{U}(\{dip\})} = t_{\mathrm{U}(\{dip\})}) \rightarrow \overline{\mathbf{cast}}(t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(\{dip\})}, t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}{(t_{\mathrm{U}(R)} \cap \mathtt{D}\mathtt{'} = \mathtt{D}\mathtt{'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D},\mathtt{D}\mathtt{'},\mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Guard 2}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(t_{\mathrm{U}(R)} \cap \mathtt{D} = \mathtt{D}\mathtt{'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D},\mathtt{D}\mathtt{'},\mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}{(\ (t_{\mathrm{U}(R)} \cap \mathtt{D} = \mathtt{D}\mathtt{'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D},\mathtt{D}\mathtt{'},\mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})\ )\ [\,\mathtt{D} := t_{\mathrm{U}(\{dip\})}, \mathtt{D}\mathtt{'} := t_{\mathrm{U}(\{dip\})}, \mathtt{msg} := t_{\mathrm{U}(m)}\,] \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Substitution}}{\sum_{\mathtt{D},\mathtt{D}\mathtt{'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (t_{\mathrm{U}(R)} \cap \mathtt{D} = \mathtt{D}\mathtt{'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D},\mathtt{D}\mathtt{'},\mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Sum 2}\ \ (2\text{ times})}{\sum_{\mathtt{D},\mathtt{D}\mathtt{'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (t_{\mathrm{U}(R)} \cap \mathtt{D} = \mathtt{D}\mathtt{'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D},\mathtt{D}\mathtt{'},\mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) + \mathrm{S}[\,\mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)}\,] \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Choice 2}}{(\ \sum_{\mathtt{D},\mathtt{D}\mathtt{'}:\mathrm{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (\mathtt{R} \cap \mathtt{D} = \mathtt{D}\mathtt{'}) \rightarrow \overline{\mathbf{cast}}(\mathtt{D},\mathtt{D}\mathtt{'},\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R}) + \mathrm{S}\ )\ [\,\mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)}\,] \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Substitution}}{\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Definition of G}\ \textsc{Recursion 2}$$

where S is an expression equal to all summands of G except for the first one.

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{cast}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')$ because Pair 2.6 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $\mathbf{unicast}(dip, m) \in B_1$. Combining this with the conclusion of the derivation above gives

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{cast}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')}\ \text{Induction hypothesis} \qquad \dfrac{}{\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textit{(see above)}}{\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{cast}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m)) | \overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Par 3}}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\gamma_C(\mathbf{cast}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m)) | \overline{\mathbf{cast}}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{Comm 2}}{\mathbf{starcast} \in V \cup \{\tau\}\ \dfrac{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{starcast}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}{\nabla_V \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{starcast}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \equiv \nabla_V \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{T14}}{\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{\mathbf{starcast}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m))} \equiv \mathrm{T}(ip : \mathrm{P}' : R)}\ \textsc{Allow 2}$$

So it is indeed the case that

$$\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R) \text{ for all } a \in A_2 = \{\, \mathbf{starcast}(\mathrm{U}(\{dip\}), \mathrm{U}(\{dip\}), \mathrm{U}(m)) \,\}$$

which finishes the induction step for the case of the UNICAST (T3-1) inference rule.

Unicast (T3-2): AWN defines the inference rule

$$\frac{\mathrm{P} \xrightarrow{\neg\mathbf{unicast}(dip,m)} \mathrm{P}' \qquad dip \notin R}{ip : \mathrm{P} : R \xrightarrow{\tau} ip : \mathrm{P}' : R}\ \textsc{Unicast}\ (\text{T3-2})$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\tau \in A_1 \wedge ip : \mathrm{P} : R \xrightarrow{A_1} ip : \mathrm{P}' : R$, namely Pair 2.3 in Table 2.10. The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R)$ for all $a \in A_2$ where

$$A_2 = \{\ \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m))\ \}$$

In mCRL2, the following derivation can be made:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{\neg\mathbf{uni}}(t_{\mathrm{U}(dip)},\emptyset,t_{\mathrm{U}(m)}) \xrightarrow{[\![\overline{\neg\mathbf{uni}}(t_{\mathrm{U}(dip)},\emptyset,t_{\mathrm{U}(m)})]\!]} \checkmark}\ \textsc{Axiom}}{\overline{\neg\mathbf{uni}}(t_{\mathrm{U}(dip)},\emptyset,t_{\mathrm{U}(m)}) \xrightarrow{\overline{\neg\mathbf{uni}}([\![t_{\mathrm{U}(dip)}]\!],[\![\emptyset]\!],[\![t_{\mathrm{U}(m)}]\!])} \checkmark}\ \text{Definition 2.1}}{\overline{\neg\mathbf{uni}}(t_{\mathrm{U}(dip)},\emptyset,t_{\mathrm{U}(m)}) \xrightarrow{\overline{\neg\mathbf{uni}}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \checkmark}\ \text{Definition of } t}{\overline{\neg\mathbf{uni}}(t_{\mathrm{U}(dip)},\emptyset,t_{\mathrm{U}(m)}) \xrightarrow{\overline{\neg\mathbf{uni}}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \checkmark}\ \textsc{Seq}\ 1}{\overline{\neg\mathbf{uni}}(t_{\mathrm{U}(dip)},\emptyset,t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) \xrightarrow{\overline{\neg\mathbf{uni}}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ [\![t_{\mathrm{U}(dip)} \notin t_{\mathrm{U}(R)}]\!] = \mathit{true}\ \textsc{Guard}\ 2}{(t_{\mathrm{U}(dip)} \notin t_{\mathrm{U}(R)}) \to \overline{\neg\mathbf{uni}}(t_{\mathrm{U}(dip)},\emptyset,t_{\mathrm{U}(m)}).\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) \xrightarrow{\overline{\neg\mathbf{uni}}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \text{Substitution}}{\big((\mathtt{d} \notin t_{\mathrm{U}(R)}) \to \overline{\neg\mathbf{uni}}(\mathtt{D},\emptyset,\mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})\big)[\,\mathtt{d} := t_{\mathrm{U}(\{dip\})}, \mathtt{msg} := t_{\mathrm{U}(m)}\,] \xrightarrow{\overline{\neg\mathbf{uni}}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \textsc{Sum}\ 2\ \ (2\text{ times})}{\sum_{\mathtt{d}:\mathrm{T}(\mathrm{IP}),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})}(\mathtt{d} \notin t_{\mathrm{U}(R)}) \to \overline{\neg\mathbf{uni}}(\mathtt{d},\emptyset,\mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) \xrightarrow{\overline{\neg\mathbf{uni}}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \textsc{Choice}\ 2}{\sum_{\mathtt{d}:\mathrm{T}(\mathrm{IP}),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})}(\mathtt{d} \notin t_{\mathrm{U}(R)}) \to \overline{\neg\mathbf{uni}}(\mathtt{d},\emptyset,\mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) + \mathrm{S}[\,\mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)}\,] \xrightarrow{\overline{\neg\mathbf{uni}}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \text{Substitution}}{\text{Definition of G}\ \dfrac{\big(\sum_{\mathtt{d}:\mathrm{T}(\mathrm{IP}),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})}(\mathtt{d} \notin \mathtt{R}) \to \overline{\neg\mathbf{uni}}(\mathtt{d},\emptyset,\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R}) + \mathrm{S}\big)[\,\mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)}\,] \xrightarrow{\overline{\neg\mathbf{uni}}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}{\mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) \xrightarrow{\overline{\neg\mathbf{uni}}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \textsc{Recursion}\ 2}$$

where S is an expression equal to all summands of G except for the second one.

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{P}) \xrightarrow{\neg\mathbf{uni}(\mathrm{U}(dip),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')$ because Pair 2.7 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $\neg\mathbf{unicast}(dip, m) \in B_1$. Combining this with the conclusion of the derivation above gives

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{P}) \xrightarrow{\neg\mathbf{uni}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')}\ \text{Induction hypothesis} \qquad \textit{(see above)}}{\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}) \xrightarrow{\neg\mathbf{uni}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))|\overline{\neg\mathbf{uni}}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})}\ \textsc{Par}\ 6}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})) \xrightarrow{\gamma_C(\neg\mathbf{uni}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))|\overline{\neg\mathbf{uni}}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}))}\ \textsc{Comm}\ 2}{\mathbf{t} \in V \cup \{\tau\}\ \dfrac{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{t}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}))}{\nabla_V\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{t}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \equiv \nabla_V\Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)},t_{\mathrm{U}(R)}))}\ \textsc{Allow}\ 2}{\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{\mathbf{t}(\mathrm{U}(dip),[\![\emptyset]\!],\mathrm{U}(m))} \equiv \mathrm{T}(ip : \mathrm{P}' : R)}\ \text{Apply } \gamma_C\ \ \text{T14}$$

So it is indeed the case that

$$\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R) \text{ for all } a \in A_2 = \{\ \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m))\ \}$$

which finishes the induction step for the case of the UNICAST (T3-2) inference rule.

Deliver (T3): AWN defines the inference rule

$$\frac{\mathrm{P} \xrightarrow{\mathbf{deliver}(d)} \mathrm{P}'}{ip : \mathrm{P} : R \xrightarrow{ip:\mathbf{deliver}(d)} ip : \mathrm{P}' : R}\ \textsc{Deliver (T3)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $ip : \mathbf{deliver}(d) \in A_1 \wedge ip : \mathrm{P} : R \xrightarrow{A_1} ip : \mathrm{P}' : R$, namely Pair 2.12 in Table 2.10. The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R)$ for all $a \in A_2 = \{\, \mathbf{deliver}(\mathrm{U}(ip), [\![\mathrm{T}_\xi(d)]\!]) \,\}$.
In mCRL2, the following derivation can be made:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{\mathbf{del}}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(d)}) \xrightarrow{[\![\overline{\mathbf{del}}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(d)})]\!]} \checkmark}\ \textsc{Axiom}}{\overline{\mathbf{del}}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(d)}) \xrightarrow{\overline{\mathbf{del}}([\![t_{\mathrm{U}(ip)}]\!], [\![t_{\mathrm{U}(d)}]\!])} \checkmark}\ \text{Definition 2.1}}{\overline{\mathbf{del}}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(d)}) \xrightarrow{\overline{\mathbf{del}}(\mathrm{U}(ip), \mathrm{U}(d))} \checkmark}\ \text{Definition of } t}{\overline{\mathbf{del}}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(d)}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{del}}(\mathrm{U}(ip), \mathrm{U}(d))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Seq 1}}{(\,\overline{\mathbf{del}}(t_{\mathrm{U}(ip)}, \mathtt{msg}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})\,)\,[\,\mathtt{data} := t_{\mathrm{U}(d)}\,] \xrightarrow{\overline{\mathbf{del}}(\mathrm{U}(ip), \mathrm{U}(d))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Substitution}}{\sum_{\mathtt{data}:\mathrm{T}(\mathrm{DATA})} \overline{\mathbf{del}}(\mathtt{d}, \mathtt{data}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{del}}(\mathrm{U}(ip), \mathrm{U}(d))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Sum 2}}{\sum_{\mathtt{data}:\mathrm{T}(\mathrm{DATA})} \overline{\mathbf{del}}(t_{\mathrm{U}(ip)}, \mathtt{data}).\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) + \mathrm{S}[\,\mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(R)}\,] \xrightarrow{\overline{\mathbf{del}}(\mathrm{U}(ip), \mathrm{U}(d))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Choice 2}}{(\,\sum_{\mathtt{data}:\mathrm{T}(\mathrm{DATA})} \overline{\mathbf{del}}(\mathtt{ip}, \mathtt{data}).\mathrm{G}(\mathtt{ip}, \mathtt{R}) + \mathrm{S}\,)\,[\,\mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{R} := t_{\mathrm{U}(\{ip\})}\,] \xrightarrow{\overline{\mathbf{del}}(\mathrm{U}(ip), \mathrm{U}(d))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Substitution}}{\mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{del}}(\mathrm{U}(ip), \mathrm{U}(d))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \text{Definition of G}\ \textsc{Recursion 2}$$

where S is an expression equal to all summands of G except for the third one.

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{del}(\mathrm{U}(ip), \mathrm{U}(d))} \equiv \mathrm{T}(\mathrm{P}')$ because Pair 2.9 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $\mathbf{deliver}(d) \in B_1$. Combining this with the conclusion of the derivation above gives

$$\mathbf{deliver} \in V \cup \{\tau\}\ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{del}(\mathrm{U}(ip), \mathrm{U}(d))} \equiv \mathrm{T}(\mathrm{P}')\ \text{Induction hypothesis} \qquad \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\overline{\mathbf{del}}(\mathrm{U}(ip), \mathrm{U}(d))} \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})\ \textit{(see above)}}{\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{del}(\mathrm{U}(ip), \mathrm{U}(d)) | \overline{\mathbf{del}}(\mathrm{U}(ip), \mathrm{U}(d))} \equiv \mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Par 3}}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\gamma_C(\mathbf{del}(\mathrm{U}(ip), \mathrm{U}(d)) | \overline{\mathbf{del}}(\mathrm{U}(ip), \mathrm{U}(d)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip), \mathrm{U}(d))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \text{Apply } \gamma_C}{\nabla_V \rho_{\{\neg \mathbf{unicast} \to \mathbf{t}\}} \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip), \mathrm{U}(d))} \equiv \nabla_V \rho_{\{\neg \mathbf{unicast} \to \mathbf{t}\}} \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{Allow 2}}{\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip), \mathrm{U}(d))} \equiv \mathrm{T}(ip : \mathrm{P}' : R)}\ \mathrm{T14}$$

So it is indeed the case that

$$\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R) \text{ for all } a \in A_2 = \{\, \mathbf{deliver}(\mathrm{U}(ip), [\![\mathrm{T}_\xi(d)]\!]) \,\}$$

which finishes the induction step for the case of the DELIVER (T3) inference rule.

Internal (T3): AWN defines the inference rule

$$\frac{\mathrm{P} \xrightarrow{\tau} \mathrm{P}'}{ip : \mathrm{P} : R \xrightarrow{\tau} ip : \mathrm{P}' : R}\ \textsc{Internal}\ (\mathrm{T3})$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\tau \in A_1 \wedge ip : \mathrm{P} : R \xrightarrow{A_1} ip : \mathrm{P}' : R$, namely Pair 2.3 in Table 2.10. The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R)$ for all $a \in A_2 = \{\, \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \,\}$ for some $D, R \in \mathrm{Set(IP)}$ where $R \subseteq D$ and some $m \in \mathrm{MSG}$.

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')$ for some $D, R \in$ Set(IP) where $R \subseteq D$ and some $m \in \mathrm{MSG}$ because Pair 2.3 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $\tau \in B_1$. This means that the following derivation can be made in mCRL2:

$$\mathbf{t} \in V \cup \{\tau\}\ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')}\ \text{Induction hypothesis}}{\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})}\ \textsc{Par}\ 2}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\gamma_C(\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{Comm}\ 2}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \text{Apply } \gamma_C}{\nabla_V\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \nabla_V\Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)}))}\ \textsc{Allow}\ 2}{\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(ip : \mathrm{P}' : R)}\ \textsc{T14}$$

So it is indeed the case that

$$\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R) \text{ for all } a \in A_2 = \{\, \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \,\}$$

which finishes the induction step for the case of the INTERNAL (T3) inference rule.

Arrive (T3-1): AWN defines the inference rule

$$\frac{\mathrm{P} \xrightarrow{\mathbf{receive}(m)} \mathrm{P}'}{ip : \mathrm{P} : R \xrightarrow{\{ip\}\neg\emptyset:\mathbf{arrive}(m)} ip : \mathrm{P}' : R}\ \textsc{Arrive}\ (\mathrm{T3\text{-}1})$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\{ip\}\neg\emptyset : \mathbf{arrive}(m) \in A_1 \wedge ip : \mathrm{P} : R \xrightarrow{A_1} ip : \mathrm{P}' : R$, namely Pair 2.13 in Table 2.10. The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R)$ for all $a \in A_2$ where

$$A_2 = \left\{ \mathbf{arrive}(\hat{\mathrm{D}}, \hat{\mathrm{D}}\text{'}, \mathrm{U}(m)) \;\middle|\; \begin{array}{c} \hat{\mathrm{D}}, \hat{\mathrm{D}}\text{'} \in \mathrm{T}(\mathrm{Set(IP)}) \\ \hat{\mathrm{D}}\text{'} \subseteq \hat{\mathrm{D}} \\ \{\mathrm{U}(ip)\} \subseteq \hat{\mathrm{D}}\text{'} \end{array} \right\}$$

Note that $\{\mathrm{U}(ip)\} \subseteq \hat{\mathrm{D}}\text{'} \Rightarrow [\![t_{\mathrm{U}(ip)} \in t_{\hat{\mathrm{D}}\text{'}}]\!] = \mathit{true}$.

In mCRL2, the following derivation can be made for all $\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'} \in \mathtt{T}(\mathrm{Set}(\mathrm{IP}))$ such that $\hat{\mathtt{D}}\mathtt{'} \subseteq \hat{\mathtt{D}} \wedge \mathtt{U}(ip) \in \hat{\mathtt{D}}\mathtt{'}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{\mathbf{receive}}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, , t_{\mathtt{U}(m)}) \xrightarrow{[\![\overline{\mathbf{receive}}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, , t_{\mathtt{U}(m)})]\!]} \checkmark}\ \text{Axiom}}{\overline{\mathbf{receive}}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, , t_{\mathtt{U}(m)}) \xrightarrow{\overline{\mathbf{receive}}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, [\![t_{\mathtt{U}(m)}]\!])} \checkmark}\ \text{Definition 2.1}}{\overline{\mathbf{receive}}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, , t_{\mathtt{U}(m)}) \xrightarrow{\overline{\mathbf{receive}}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \checkmark}\ \text{Definition of } t}{\overline{\mathbf{receive}}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, , t_{\mathtt{U}(m)}).\mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)}) \xrightarrow{\overline{\mathbf{receive}}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)})}\ \text{Seq 1}}{[\![t_{\mathtt{U}(ip)} \in t_{\hat{\mathtt{D}}},]\!] = \mathit{true}}\ \text{Guard 2}}{(t_{\mathtt{U}(ip)} \in t_{\hat{\mathtt{D}}},) \to \overline{\mathbf{receive}}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, , t_{\mathtt{U}(m)}).\mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)}) \xrightarrow{\overline{\mathbf{receive}}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)})}\ \text{Substitution}}{(\ (t_{\mathtt{U}(ip)} \in \mathtt{D}\mathtt{'}) \to \overline{\mathbf{receive}}(\mathtt{D}, \mathtt{D}\mathtt{'}, \mathtt{msg}).\mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)})\ )\ [\ \mathtt{D} := t_{\hat{\mathtt{D}}}, \mathtt{D}\mathtt{'} := t_{\hat{\mathtt{D}}}, , \mathtt{msg} := t_{\mathtt{U}(m)}\ ] \xrightarrow{\overline{\mathbf{receive}}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)})}\ \text{Sum 2 (3 times)}}{\textstyle\sum_{\mathtt{D},\mathtt{D}\mathtt{'}:\mathtt{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathtt{T}(\mathrm{MSG})} (t_{\mathtt{U}(ip)} \in \mathtt{D}\mathtt{'}) \to \overline{\mathbf{receive}}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, , \mathtt{msg}).\mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)}) \xrightarrow{\overline{\mathbf{receive}}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)})}\ \text{Choice 2}}{\textstyle\sum_{\mathtt{D},\mathtt{D}\mathtt{'}:\mathtt{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathtt{T}(\mathrm{MSG})} (t_{\mathtt{U}(ip)} \in \mathtt{D}\mathtt{'}) \to \overline{\mathbf{receive}}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, , \mathtt{msg}).\mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)}) + \mathrm{S}[\ \mathtt{ip} := t_{\mathtt{U}(ip)}, \mathtt{R} := t_{\mathtt{U}(R)}\ ] \xrightarrow{\overline{\mathbf{receive}}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)})}\ \text{Substitution}}{\text{Definition of G}\ \dfrac{(\ \textstyle\sum_{\mathtt{D},\mathtt{D}\mathtt{'}:\mathtt{T}(\mathrm{Set}(\mathrm{IP})),\mathtt{msg}:\mathtt{T}(\mathrm{MSG})} (\mathtt{ip} \in \mathtt{D}\mathtt{'}) \to \overline{\mathbf{receive}}(t_{\hat{\mathtt{D}}}, t_{\hat{\mathtt{D}}}, , \mathtt{msg}).\mathrm{G}(\mathtt{ip}, \mathtt{R}) + \mathrm{S}\ )\ [\ \mathtt{ip} := t_{\mathtt{U}(ip)}, \mathtt{R} := t_{\mathtt{U}(R)}\ ] \xrightarrow{\overline{\mathbf{receive}}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)})}{\mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)}) \xrightarrow{\overline{\mathbf{receive}}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)})}\ \text{Recursion 2}$$

where S is an expression equal to all summands of G except for the one containing $\overline{\mathbf{receive}}$. From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \equiv \mathrm{T}(\mathrm{P}')$ because Pair 2.10 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $\mathbf{receive}(m) \in B_1$. Combining this with the conclusion of the derivation above gives

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \equiv \mathrm{T}(\mathrm{P}')}\ \text{Induction hypothesis} \quad \dfrac{}{\mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)}) \xrightarrow{\overline{\mathbf{receive}}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \equiv \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)})}\ \textit{(see above)}}{\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m)) | \overline{\mathbf{receive}}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \equiv \mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)})}\ \text{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)})) \xrightarrow{\gamma_C(\mathbf{receive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m)) | \overline{\mathbf{receive}}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)}))}\ \text{Apply } \gamma_C}{\mathbf{arrive} \in V \cup \{\tau\}\ \dfrac{\Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)})) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \equiv \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)}))}{\nabla_V \Gamma_C(\mathrm{T}(\mathrm{P}) \,||\, \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)})) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \equiv \nabla_V \Gamma_C(\mathrm{T}(\mathrm{P}') \,||\, \mathrm{G}(t_{\mathtt{U}(ip)}, t_{\mathtt{U}(R)}))}\ \text{T14}}\ \text{Allow 2}}{\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m))} \equiv \mathrm{T}(ip : \mathrm{P}' : R)}$$

So it is indeed the case that

$$\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R) \text{ for all } a \in A_2 = \left\{ \mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'}, \mathtt{U}(m)) \;\middle|\; \begin{array}{c} \hat{\mathtt{D}}, \hat{\mathtt{D}}\mathtt{'} \in \mathtt{T}(\mathrm{Set}(\mathrm{IP})) \\ \hat{\mathtt{D}}\mathtt{'} \subseteq \hat{\mathtt{D}} \\ \{\mathtt{U}(ip)\} \subseteq \hat{\mathtt{D}}\mathtt{'} \end{array} \right\}$$

which finishes the induction step for the case of the ARRIVE (T3-1) inference rule.

Cast (T4-1): AWN defines the inference rule

$$H \subseteq R \wedge K \cap R = \emptyset \;\frac{\mathrm{M} \xrightarrow{R:\mathbf{*cast}(m)} \mathrm{M}' \qquad \mathrm{N} \xrightarrow{H\neg K:\mathbf{arrive}(m)} \mathrm{N}'}{\mathrm{M} \,||\, \mathrm{N} \xrightarrow{R:\mathbf{*cast}(m)} \mathrm{M}' \,||\, \mathrm{N}'}\ \textsc{Cast (T4-1)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $R : \mathbf{*cast}(m) \in A_1 \wedge \mathrm{M} \,||\, \mathrm{N} \xrightarrow{A_1} \mathrm{M}' \,||\, \mathrm{N}'$, namely Pair 2.11 in Table 2.10. The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}(\mathrm{M} \,||\, \mathrm{N}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{M}' \,||\, \mathrm{N}')$ for all $a \in \{\, \mathbf{starcast}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \,\}$ for some D with $R \subseteq D$.

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{starcast}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{M}')$ because Pair 2.11 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $R : \mathbf{*cast}(m) \in B_1$.

Similarly, $\mathrm{T}(\mathrm{N}) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{N}')$ for all $\hat{\mathtt{D}}' \subseteq \hat{\mathtt{D}}$ because Pair 2.13 is the only $(B_1, B_2) \in \mathscr{A}$. $H\neg K : \mathbf{arrive}(m) \in B_1$. This observation about $\mathrm{T}(\mathrm{M})$ can be refined as follows:

$$\cfrac{\cfrac{\cfrac{\cfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{N}')}\ \text{Induction hypothesis}}{H \subseteq R \wedge K \cap R = \emptyset}\ \text{Choose } t_{\hat{\mathtt{D}}'} = t_{\mathrm{U}(R)}}{R \subseteq D}\ \text{Choose } t_{\hat{\mathtt{D}}} = t_{\mathrm{U}(D)}}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{N}')}\ \text{Definition of } t \text{ (2 times)}$$

This means that the following derivation can be made in mCRL2:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{starcast}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{M}')}\ \text{Induction hypothesis} \qquad \cfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{N}')}\ \textit{(see above)}}{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N}) \xrightarrow{\mathbf{starcast}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))|\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}')}\ \text{Par 3}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_C(\mathbf{starcast}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))|\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \text{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{s}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \text{Apply } \gamma_C}{\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}(\mathbf{s}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m)))} \equiv \Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \text{Comm 2}}{\mathbf{s} \in V \cup \{\tau\}}\ \text{Apply } \gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}}{\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{s}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \text{Allow 2}}{\nabla_V\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{s}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \nabla_V\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \text{Rename 2}}{\rho_R\nabla_V\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{R\bullet\mathbf{s}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \rho_R\nabla_V\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \text{Apply } R\bullet}{\rho_R\nabla_V\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{starcast}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \rho_R\nabla_V\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \text{T15}}{\mathrm{T}(\mathrm{M} \,||\, \mathrm{N}) \xrightarrow{\mathbf{starcast}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{M}' \,||\, \mathrm{N}')}$$

Clearly, Pair 2.11 applies to the conclusion of this derivation, and thus the induction step of Lemma 4.7 for the case of Cast (T4-1) is established.

Cast (T4-2): Similar to the Lemma 4.7-proof for Cast (T4-1).

Cast (T4-3): AWN defines the inference rule

$$\frac{\mathrm{M} \xrightarrow{H\neg K:\mathbf{arrive}(m)} \mathrm{M}' \qquad \mathrm{N} \xrightarrow{H'\neg K':\mathbf{arrive}(m)} \mathrm{N}'}{\mathrm{M} \mid\mid \mathrm{N} \xrightarrow{(H\cup H')\neg(K\cup K'):\mathbf{arrive}(m)} \mathrm{M}' \mid\mid \mathrm{N}'} \textsc{Cast (T4-3)}$$

There exist an (A_1,A_2) in $\mathscr{A}$ such that $(H\cup H')\neg(K\cup K') : \mathbf{arrive}(m) \in A_1 \wedge \mathrm{M} \mid\mid \mathrm{N} \xrightarrow{A_1} \mathrm{M}' \mid\mid \mathrm{N}'$, namely Pair 2.13 in Table 2.10. The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{a}\equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N}')$ for all $a \in A_2$ where

$$A_2 = \left\{ \mathbf{arrive}(\hat{\mathrm{D}},\hat{\mathrm{D}}\text{'},\mathrm{U}(m)) \;\middle|\; \begin{array}{c} \hat{\mathrm{D}},\hat{\mathrm{D}}\text{'}\in\mathrm{T}(\mathrm{Set}(\mathrm{IP})) \\ \hat{\mathrm{D}}\text{'}\subseteq\hat{\mathrm{D}} \\ \mathrm{U}(H)\cup\mathrm{U}(H')\subseteq\hat{\mathrm{D}}\text{'} \\ (\mathrm{U}(K)\cup\mathrm{U}(K'))\cap\hat{\mathrm{D}}\text{'}=\emptyset \end{array} \right\}$$

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{arrive}(\hat{\mathrm{D}},\hat{\mathrm{D}}\text{'},\mathrm{U}(m))}\equiv \mathrm{T}(\mathrm{M}')$ for all $\hat{\mathrm{D}}\text{'} \subseteq \hat{\mathrm{D}}$ because Pair 2.13 in Table 2.10 is the only $(B_1,B_2) \in \mathscr{A}$. $H\neg K : \mathbf{arrive}(m) \in B_1$. For the same reason, $\mathrm{T}(\mathrm{N}) \xrightarrow{\mathbf{arrive}(\hat{\mathrm{D}},\hat{\mathrm{D}}\text{'},\mathrm{U}(m))}\equiv \mathrm{T}(\mathrm{N}')$ for all $\hat{\mathrm{D}}\text{'} \subseteq \hat{\mathrm{D}}$. This means that the following derivation can be made in mCRL2 for $R \subseteq D$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{arrive}(\hat{\mathrm{D}},\hat{\mathrm{D}}\text{'},\mathrm{U}(m))}\equiv \mathrm{T}(\mathrm{M}')}\text{Induction hypothesis}}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))}\equiv \mathrm{T}(\mathrm{M}')}\text{Choose } \hat{\mathrm{D}} = \mathrm{U}(D),\ \hat{\mathrm{D}}\text{'} = \mathrm{U}(R) \quad \dfrac{}{\mathrm{T}(\mathrm{N}) \xrightarrow{\mathbf{arrive}(\hat{\mathrm{D}},\hat{\mathrm{D}}\text{'},\mathrm{U}(m))}\equiv \mathrm{T}(\mathrm{N}')}\text{Induction hypothesis}}{\mathrm{T}(\mathrm{N}) \xrightarrow{\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))}\equiv \mathrm{T}(\mathrm{N}')}\text{Choose } \hat{\mathrm{D}} = \mathrm{U}(D),\ \hat{\mathrm{D}}\text{'} = \mathrm{U}(R)}{\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N}) \xrightarrow{\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))\mid\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))}\equiv \mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}')}\textsc{Par 3}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_C(\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))\mid\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m)))}\equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\textsc{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))\mid\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))}\equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\text{Apply } \gamma_C}{\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}(\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))\mid\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m)))}\equiv \Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\textsc{Comm 2}}{\mathbf{a} \in V\cup\{\tau\} \quad \Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{a}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))}\equiv \Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\textsc{Apply } \gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}}{\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{a}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))}\equiv \nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\textsc{Allow 2}}{\rho_R\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{R\bullet\mathbf{a}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))}\equiv \rho_R\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\textsc{Rename 2}}{\rho_R\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))}\equiv \rho_R\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\textsc{Apply } R\bullet}{\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{\mathbf{arrive}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))}\equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N}')}\textsc{T15}$$

where $\hat{\mathrm{D}}\text{'} \subseteq \hat{\mathrm{D}}$, $\mathrm{U}(H) \subseteq \hat{\mathrm{D}}\text{'}$, $\mathrm{U}(H') \subseteq \hat{\mathrm{D}}\text{'}$, $\mathrm{U}(K) \cap \hat{\mathrm{D}}\text{'} = \emptyset$ and $\mathrm{U}(K') \cap \hat{\mathrm{D}}\text{'} = \emptyset$ (according to the induction hypothesis). The side condition of Pair 2.13 holds for the conclusion of the derivation above:

$$\begin{aligned} \hat{\mathrm{D}}\text{'} \subseteq \hat{\mathrm{D}} &\Leftrightarrow \hat{\mathrm{D}}\text{'} \subseteq \hat{\mathrm{D}} \\ \mathrm{U}(H) \subseteq \hat{\mathrm{D}}\text{'} \wedge \mathrm{U}(H') \subseteq \hat{\mathrm{D}}\text{'} &\Leftrightarrow \mathrm{U}(H) \cup \mathrm{U}(H') \subseteq \hat{\mathrm{D}}\text{'} \\ \mathrm{U}(K) \cap \hat{\mathrm{D}}\text{'} = \emptyset \wedge \mathrm{U}(K') \cap \hat{\mathrm{D}}\text{'} = \emptyset &\Leftrightarrow (\mathrm{U}(K) \cup \mathrm{U}(K')) \cap \hat{\mathrm{D}}\text{'} = \emptyset \end{aligned}$$

This proves that

$$\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{a}\equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N}') \text{ for all } a \in \left\{ \mathbf{arrive}(\hat{\mathrm{D}},\hat{\mathrm{D}}\text{'},\mathrm{U}(m)) \;\middle|\; \begin{array}{c} \hat{\mathrm{D}},\hat{\mathrm{D}}\text{'}\in\mathrm{T}(\mathrm{Set}(\mathrm{IP})) \\ \hat{\mathrm{D}}\text{'}\subseteq\hat{\mathrm{D}} \\ \mathrm{U}(H)\cup\mathrm{U}(H')\subseteq\hat{\mathrm{D}}\text{'} \\ (\mathrm{U}(K)\cup\mathrm{U}(K'))\cap\hat{\mathrm{D}}\text{'}=\emptyset \end{array} \right\}$$

finishing the induction step for the case of the Cast (T4-3) inference rule.

Cast (T4-4): AWN defines the inference rule

$$\frac{\mathrm{M} \xrightarrow{R:\textbf{*cast}(m)} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{\tau} [\,\mathrm{M}'\,]}\ \textsc{Cast}\ (\text{T4-4})$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\tau \in A_1 \wedge [\,\mathrm{M}\,] \xrightarrow{A_1} [\,\mathrm{M}'\,]$, namely Pair 2.3 in Table 2.10. The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{a} \equiv \mathrm{T}([\,\mathrm{M}'\,])$ for all $a \in A_2 = \{\, \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \,\}$ for some $D, R \in \mathrm{Set(IP)}$ where $R \subseteq D$ and some $m \in \mathrm{MSG}$.

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{M}) \xrightarrow{\textbf{starcast}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{M}')$ for some $D \supseteq R$ because Pair 2.11 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $R : \textbf{*cast}(m) \in B_1$. This means that the following derivation can be made in mCRL2:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{\textbf{starcast}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{M}')}\ \text{Induction hypothesis}}{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H} \xrightarrow{\textbf{starcast}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{M}') \,||\, \mathrm{H}}\ \textsc{Par}\ 2}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\gamma_C(\textbf{starcast}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Comm}\ 2}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\textbf{starcast}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \gamma_C}{\rho_{\{\textbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{G}) \xrightarrow{\{\textbf{starcast}\to\mathbf{t}\}\bullet\textbf{starcast}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \rho_{\{\textbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{G})}\ \textsc{Rename}\ 2}{\mathbf{t} \in V \cup \{\tau\}\quad \dfrac{\rho_{\{\textbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{G}) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \rho_{\{\textbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{G})}{\nabla_V\rho_{\{\textbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{G}) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \nabla_V\rho_{\{\textbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{G})}\ \textsc{Allow}\ 2}{\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}([\,\mathrm{M}'\,])}\ \text{T16}}$$

So it is indeed the case that

$$\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{a} \equiv \mathrm{T}([\,\mathrm{M}'\,]) \text{ for all } a \in A_2 = \{\, \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \,\}$$

which finishes the induction step for the case of the CAST (T4-4) inference rule.

Deliver (T4-1): AWN defines the inference rule

$$\frac{\mathrm{M} \xrightarrow{ip:\mathbf{deliver}(d)} \mathrm{M}'}{\mathrm{M} \mid\mid \mathrm{N} \xrightarrow{ip:\mathbf{deliver}(d)} \mathrm{M}' \mid\mid \mathrm{N}}\ \textsc{Deliver (T4-1)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $ip : \mathbf{deliver}(d) \in A_1 \wedge \mathrm{M} \mid\mid \mathrm{N} \xrightarrow{A_1} \mathrm{M}' \mid\mid \mathrm{N}$, namely Pair 2.12 in Table 2.10. The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N})$ for all $a \in A_2 = \{\, \mathbf{deliver}(\mathrm{U}(ip), \mathrm{U}(d)) \,\}$.

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \mathrm{T}(\mathrm{M}')$ because Pair 2.12 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $ip : \mathbf{deliver}(d) \in B_1$. This means that the following derivation can be made in mCRL2:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \mathrm{T}(\mathrm{M}')}\ \text{Induction hypothesis}}{\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N}) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N})}\ \textsc{Par 2}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_C(\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \textsc{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \text{Apply } \gamma_C}{\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}(\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d)))} \equiv \Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \textsc{Comm 2}}{\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \text{Apply } \gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}}{\mathbf{deliver} \in V \cup \{\tau\}}\ \textsc{Allow 2}}{\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \textsc{Rename 2}}{\rho_R\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{R\bullet\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \rho_R\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \text{Apply } R\bullet}{\rho_R\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \rho_R\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \text{T15}}{\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N})}$$

So it is indeed the case that

$$\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N}) \text{ for all } a \in A_2 = \{\, \mathbf{deliver}(\mathrm{U}(ip), \mathrm{U}(d)) \,\}$$

which finishes the induction step for the case of the Deliver (T4-1) inference rule.

Deliver (T4-2): Similar to the Lemma 4.7-proof for Deliver (T4-1).

Deliver (T4-3): AWN defines the inference rule

$$\frac{\mathrm{M} \xrightarrow{ip:\mathbf{deliver}(d)} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{ip:\mathbf{deliver}(d)} [\,\mathrm{M}'\,]}\ \textsc{Deliver}\ (\text{T4-3})$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $ip : \mathbf{deliver}(d) \in A_1 \wedge \mathrm{M} \,||\, \mathrm{N} \xrightarrow{A_1} \mathrm{M}' \,||\, \mathrm{N}$, namely Pair 2.12 in Table 2.10. The induction step can therefore be proven for this particular case by finding a set of derivations in mCRL2 for $\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{a} \equiv \mathrm{T}([\,\mathrm{M}'\,])$ for all $a \in A_2 = \{\, \mathbf{deliver}(\mathrm{U}(ip), \mathrm{U}(d)) \,\}$.

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \mathrm{T}(\mathrm{M}')$ because Pair 2.12 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $ip : \mathbf{deliver}(d) \in B_1$. This means that the following derivation can be made in mCRL2:

$$\mathbf{deliver} \in V \cup \{\tau\}\ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\text{Induction hypothesis}}}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \mathrm{T}(\mathrm{M}')}\ \textsc{Par}\ 2}{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H} \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \mathrm{T}(\mathrm{M}') \,||\, \mathrm{H}}\ \textsc{Comm}\ 2}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \gamma_C}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Rename}\ 2}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\{\mathbf{starcast}\to\mathbf{t}\}\bullet\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \{\mathbf{starcast}\to\mathbf{t}\}\bullet$$

$$\dfrac{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}{\nabla_V\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \nabla_V\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Allow}\ 2$$

$$\dfrac{\nabla_V\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \nabla_V\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}{\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{\mathbf{deliver}(\mathrm{U}(ip),\mathrm{U}(d))} \equiv \mathrm{T}([\,\mathrm{M}'\,])}\ \text{T16}$$

So it is indeed the case that

$$\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{a} \equiv \mathrm{T}([\,\mathrm{M}'\,]) \text{ for all } a \in A_2 = \{\, \mathbf{deliver}(\mathrm{U}(ip), \mathrm{U}(d)) \,\}$$

which finishes the induction step for the case of the DELIVER (T4-3) inference rule.

Internal (T4-1): Similar to the Lemma 4.7-proof for Deliver (T4-1).

Internal (T4-2): Similar to the Lemma 4.7-proof for Deliver (T4-1).

Internal (T4-3): Similar to the Lemma 4.7-proof for Deliver (T4-3).

Connect (T4-1): AWN defines the inference rule

$$\frac{\mathrm{M} \xrightarrow{\textbf{connect}(ip,ip')} \mathrm{M}' \qquad \mathrm{N} \xrightarrow{\textbf{connect}(ip,ip')} \mathrm{N}'}{\mathrm{M} \,||\, \mathrm{N} \xrightarrow{\textbf{connect}(ip,ip')} \mathrm{M}' \,||\, \mathrm{N}'} \ \textsc{Connect (T4-1)}$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\textbf{connect}(ip, ip') \in A_1 \wedge \mathrm{M} \,||\, \mathrm{N} \xrightarrow{A_1} \mathrm{M}' \,||\, \mathrm{N}'$, namely Pair 2.14 in Table 2.10. For this particular case, the induction step can be proven by finding a set of derivations in mCRL2 for $\mathrm{T}(\mathrm{M} \,||\, \mathrm{N}) \xrightarrow{A_1}\equiv \mathrm{T}(\mathrm{M}' \,||\, \mathrm{N}')$ for all $a \in A_2 = \{\, \textbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip')) \,\}$.

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{M}) \xrightarrow{\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \mathrm{T}(\mathrm{M}')$ because Pair 2.14 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $\textbf{connect}(ip, ip') \in B_1$. For the same reason, $\mathrm{T}(\mathrm{N}) \xrightarrow{\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \mathrm{T}(\mathrm{N}')$. This means that the following derivation can be made in mCRL2:

$$\frac{\dfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \mathrm{T}(\mathrm{M}')}\ \text{Induction hypothesis} \qquad \dfrac{}{\mathrm{T}(\mathrm{N}) \xrightarrow{\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \mathrm{T}(\mathrm{N}')}\ \text{Induction hypothesis}}{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N}) \xrightarrow{\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))|\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}')}\ \textsc{Par 3}$$

$$\frac{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N}) \xrightarrow{\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))|\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}')}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_C(\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))|\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip')))}\equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \textsc{Comm 2}$$

$$\frac{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_C(\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))|\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip')))}\equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\textbf{c}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \text{Apply } \gamma_C$$

$$\frac{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\textbf{c}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}{\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}(\textbf{c}(\mathrm{U}(ip),\mathrm{U}(ip')))}\equiv \Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \textsc{Comm 2}$$

$$\frac{\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}(\textbf{c}(\mathrm{U}(ip),\mathrm{U}(ip')))}\equiv \Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}{\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\textbf{c}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \text{Apply } \gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}$$

$$\frac{\textbf{c} \in V \cup \{\tau\} \qquad \Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\textbf{c}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}{\nabla_V\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\textbf{c}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \nabla_V\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \textsc{Allow 2}$$

$$\frac{\nabla_V\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\textbf{c}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \nabla_V\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}{\rho_R\nabla_V\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{R\bullet\textbf{c}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \rho_R\nabla_V\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \textsc{Rename 2}$$

$$\frac{\rho_R\nabla_V\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{R\bullet\textbf{c}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \rho_R\nabla_V\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}{\rho_R\nabla_V\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \rho_R\nabla_V\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}\ \text{Apply } R\bullet$$

$$\frac{\rho_R\nabla_V\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \xrightarrow{\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \rho_R\nabla_V\Gamma_{\{\textbf{arrive}|\textbf{arrive}\to\textbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{T}(\mathrm{N}'))}{\mathrm{T}(\mathrm{M} \,||\, \mathrm{N}) \xrightarrow{\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \mathrm{T}(\mathrm{M}' \,||\, \mathrm{N}')}\ \textsc{T15}$$

So it is indeed the case that

$$\mathrm{T}(\mathrm{M} \,||\, \mathrm{N}) \xrightarrow{a}\equiv \mathrm{T}(\mathrm{M}' \,||\, \mathrm{N}) \text{ for all } a \in A_2 = \{\, \textbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip')) \,\}$$

which finishes the induction step for the case of the Connect (T4-1) inference rule.

Connect (T4-2): AWN defines the inference rule

$$\frac{\mathrm{M} \xrightarrow{\mathbf{connect}(ip,ip')} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{\mathbf{connect}(ip,ip')} [\,\mathrm{M}'\,]}\ \textsc{Connect}\ (\text{T4-2})$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $\mathbf{connect}(ip, ip') \in A_1 \wedge [\,\mathrm{M}\,] \xrightarrow{A_1} [\,\mathrm{M}'\,]$, namely Pair 2.14 in Table 2.10. For this particular case, the induction step can be proven by finding a set of derivations in mCRL2 for $\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{A_1} \equiv \mathrm{T}([\,\mathrm{M}'\,])$ for all $a \in A_2$ where $A_2 = \{\, \mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip')) \,\}$.

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))} \equiv \mathrm{T}(\mathrm{M}')$ because Pair 2.14 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $\mathbf{connect}(ip, ip') \in B_1$. This means that the following derivation can be made in mCRL2:

$$\mathbf{connect} \in V \cup \{\tau\}\ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))} \equiv \mathrm{T}(\mathrm{M}')}\ \text{Induction hypothesis}}{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H} \xrightarrow{\mathbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))} \equiv \mathrm{T}(\mathrm{M}') \,||\, \mathrm{H}}\ \textsc{Par}\ 2}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\gamma_C(\mathbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip')))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Comm}\ 2}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \gamma_C}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\{\mathbf{starcast}\to\mathbf{t}\}\bullet\mathbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Rename}\ 2}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \{\mathbf{starcast}\to\mathbf{t}\}\bullet}{\nabla_V\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))} \equiv \nabla_V\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Allow}\ 2}{\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{\mathbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip'))} \equiv \mathrm{T}([\,\mathrm{M}'\,])}\ \text{T16}$$

So it is indeed the case that

$$\mathrm{T}(\mathrm{M} \,||\, \mathrm{N}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{M}' \,||\, \mathrm{N}) \text{ for all } a \in A_2 = \{\, \mathbf{connect}(\mathrm{U}(ip), \mathrm{U}(ip')) \,\}$$

which finishes the induction step for the case of the Connect (T4-2) inference rule.

Disconnect (T4-1): Similar to the Lemma 4.7-proof for Connect (T4-1).

Disconnect (T4-2): Similar to the Lemma 4.7-proof for Connect (T4-2).

Newpkt (T4): AWN defines the inference rule

$$\frac{\mathrm{M} \xrightarrow{\{ip\}\neg K:\mathbf{arrive}(\mathtt{newpkt}(t_{\mathrm{U}(d)},t_{\mathrm{U}(dip)}))} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{ip:\mathbf{newpkt}(d,dip)} [\,\mathrm{M}'\,]}\ \textsc{Newpkt}\ (\text{T4})$$

There exists an (A_1, A_2) in $\mathscr{A}$ such that $ip : \mathbf{newpkt}(d, dip) \in A_1 \wedge [\ \mathrm{M}\ ] \xrightarrow{A_1} [\ \mathrm{M}'\ ]$, namely Pair 2.16 in Table 2.10. This base case can therefore be proven by finding a set of derivations in mCRL2 such that $\mathrm{T}([\ \mathrm{M}\ ]) \xrightarrow{a} \equiv \mathrm{T}([\ \mathrm{M}'\ ])$ for all $a \in A_2 = \{\ \mathbf{newpkt}(\{\mathrm{U}(ip)\}, \{\mathrm{U}(ip)\}, \mathtt{newpkt}(\mathrm{U}(d), \mathrm{U}(dip)))\ \}$.
In mCRL2, the following derivation can be made:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{\mathbf{newpkt}}(\{t_{\mathrm{U}(ip)}\},\{t_{\mathrm{U}(ip)}\},\mathtt{newpkt}(t_{\mathrm{U}(d)},t_{\mathrm{U}(dip)})) \xrightarrow{[\![\overline{\mathbf{newpkt}}(\{t_{\mathrm{U}(ip)}\},\{t_{\mathrm{U}(ip)}\},\mathtt{newpkt}(t_{\mathrm{U}(d)},t_{\mathrm{U}(dip)}))]\!]} \equiv \checkmark}\ \text{Axiom}}{\overline{\mathbf{newpkt}}(\{t_{\mathrm{U}(ip)}\},\{t_{\mathrm{U}(ip)}\},\mathtt{newpkt}(t_{\mathrm{U}(d)},t_{\mathrm{U}(dip)})) \xrightarrow{\overline{\mathbf{newpkt}}([\![\{t_{\mathrm{U}(ip)}\}]\!],[\![\{t_{\mathrm{U}(ip)}\}]\!],\mathtt{newpkt}([\![t_{\mathrm{U}(d)}]\!],[\![t_{\mathrm{U}(dip)}]\!]))} \equiv \checkmark}\ \text{Definition 2.1}}{\overline{\mathbf{newpkt}}(\{t_{\mathrm{U}(ip)}\},\{t_{\mathrm{U}(ip)}\},\mathtt{newpkt}(t_{\mathrm{U}(d)},t_{\mathrm{U}(dip)})) \xrightarrow{\overline{\mathbf{newpkt}}(\{[\![t_{\mathrm{U}(ip)}]\!]\},\{[\![t_{\mathrm{U}(ip)}]\!]\},\mathtt{newpkt}([\![t_{\mathrm{U}(d)}]\!],[\![t_{\mathrm{U}(dip)}]\!]))} \equiv \checkmark}\ \text{Definition of } t}{\overline{\mathbf{newpkt}}(\{t_{\mathrm{U}(ip)}\},\{t_{\mathrm{U}(ip)}\},\mathtt{newpkt}(t_{\mathrm{U}(d)},t_{\mathrm{U}(dip)})) \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \checkmark}\ \text{Seq 2}}{\overline{\mathbf{newpkt}}(\{t_{\mathrm{U}(ip)}\},\{t_{\mathrm{U}(ip)}\},\mathtt{newpkt}(t_{\mathrm{U}(d)},t_{\mathrm{U}(dip)})).\mathrm{H} \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{H}}\ \text{Substitution}}{(\overline{\mathbf{newpkt}}(\{\mathtt{ip}\},\{\mathtt{ip}\},\mathtt{newpkt}(\mathtt{data},\mathtt{dest})).\mathrm{H})\ [\mathtt{ip} := t_{\mathrm{U}(ip)}, \mathtt{data} := t_{\mathrm{U}(d)}, \mathtt{dest} := t_{\mathrm{U}(dip)}]\ \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{H}}\ \text{Sum 2}}{\sum_{\mathtt{ip}:\mathrm{T}(\mathrm{IP}),\mathtt{data}:\mathrm{T}(\mathrm{DATA}),\mathtt{dest}:\mathrm{T}(\mathrm{IP})} \overline{\mathbf{newpkt}}(\{\mathtt{ip}\},\{\mathtt{ip}\},\mathtt{newpkt}(\mathtt{data},\mathtt{dest})).\mathrm{H} \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{H}}\ \text{Substitution}}{(\sum_{\mathtt{ip}:\mathrm{T}(\mathrm{IP}),\mathtt{data}:\mathrm{T}(\mathrm{DATA}),\mathtt{dest}:\mathrm{T}(\mathrm{IP})} \overline{\mathbf{newpkt}}(\{\mathtt{ip}\},\{\mathtt{ip}\},\mathtt{newpkt}(\mathtt{data},\mathtt{dest})).\mathrm{H})[\,] \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{H}}\ \text{Recursion 2}}{\mathrm{H} \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{H}}\ \text{Definition of H}$$

From the induction hypothesis, it follows that $\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}},\hat{\mathtt{D}}\mathtt{'},\mathtt{newpkt}(t_{\mathrm{U}(d)},t_{\mathrm{U}(dip)}))} \equiv \mathrm{T}(\mathrm{M}')$ for all $\hat{\mathtt{D}}\mathtt{'} \subseteq \hat{\mathtt{D}}$ such that $\{\mathrm{U}(ip)\} \subseteq \hat{\mathtt{D}}\mathtt{'} \wedge \mathrm{U}(K) \cap \hat{\mathtt{D}}\mathtt{'} = \emptyset$ because Pair 2.13 in Table 2.10 is the only $(B_1, B_2) \in \mathscr{A}$. $\mathbf{arrive}(\mathtt{newpkt}(t_{\mathrm{U}(d)}, t_{\mathrm{U}(dip)})) \in B_1$. This observation about $\mathrm{T}(\mathrm{M})$ can be refined as follows:

$$\{\mathrm{U}(ip)\} \subseteq R \wedge K \cap R = \emptyset \quad \dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{arrive}(\hat{\mathtt{D}},\hat{\mathtt{D}}\mathtt{'},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{T}(\mathrm{M}')}\ \text{Induction hypothesis}}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{arrive}([\![\{t_{\mathrm{U}(ip)}\}]\!],[\![\{t_{\mathrm{U}(ip)}\}]\!],\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{T}(\mathrm{M}')}\ \text{Choose } \hat{\mathtt{D}} = \hat{\mathtt{D}}\mathtt{'} = \{\mathrm{U}(ip)\}}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{arrive}(\{[\![t_{\mathrm{U}(ip)}]\!]\},\{[\![t_{\mathrm{U}(ip)}]\!]\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{T}(\mathrm{M}')}\ \text{Definition of } t \text{ (2 times)}}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{arrive}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{T}(\mathrm{M}')}$$

This means that the following derivation can be made in mCRL2:

$$\mathbf{newpkt} \in V \cup \{\tau\} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{H} \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{H}}\ \textit{(see above)} \quad \dfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{arrive}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{T}(\mathrm{M}')}\ \textit{(see above)}}{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H} \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))|\mathbf{arrive}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{T}(\mathrm{M}') \,||\, \mathrm{H}}\ \text{Par 3}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\gamma_C(\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip))))|\mathbf{arrive}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\mathbf{newpkt}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \gamma_C}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\{\mathbf{starcast}\to\mathbf{t}\}\bullet\mathbf{newpkt}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Rename 2}}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\mathbf{newpkt}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \{\mathbf{starcast} \to \mathbf{t}\}\bullet}{\nabla_V \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\mathbf{newpkt}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \nabla_V \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Allow 2}}{\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{\mathbf{newpkt}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{T}([\,\mathrm{M}'\,])}\ \text{T16}$$

So it is indeed the case that

$$\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{T}(ip : \mathrm{P}' : R) \text{ for all } a \in A_2 = \{\ \mathbf{newpkt}(\{\mathrm{U}(ip)\}, \{\mathrm{U}(ip)\}, \mathtt{newpkt}(\mathrm{U}(d), \mathrm{U}(dip)))\ \}$$

which finishes the induction step for the case of the NEWPKT (T4) inference rule.

Appendix D

Complete proof of Lemma 4.8

Lemma: For T_V as defined by the rules T1 to T10 in Table 2.8 it holds that

$$\forall \mathrm{p}, a, \mathrm{Q}, \xi \,.\, \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{a} \mathrm{Q} \Rightarrow \exists (A_1, A_2) \in \mathscr{A}^{\sim}, \mathrm{q}, \sigma \,.\, a \in A_1$$
$$\wedge\, \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{q}) \wedge \xi, \mathrm{p} \xrightarrow{A_2} \xi[\sigma], \mathrm{q} \wedge \mathrm{Q} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{q})$$

Proof: The proof of Equation 2.19 is by structural induction over the rules T1 to T10 in Table 2.8. This is a sound approach because these rules yield all mCRL2 expressions that can result from a translation by T_V. For each translation rule, all representative derivations (see Definition 4.3) for expressions of the form $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{a} \mathrm{Q}$ are listed, and for each possible derivation Lemma 4.8 is proven.

Induction hypothesis: For all recursions $\mathrm{T}_W(\xi, \mathrm{p})$ that are part of a translation rule defining T_V, it holds that

$$\forall a, \mathrm{Q} \,.\, \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{a} \mathrm{Q} \Rightarrow \exists (A_1, A_2) \in \mathscr{A}^{\sim}, \mathrm{q}, \sigma \,.\, a \in A_1 \wedge$$
$$\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{q}) \wedge \xi, \mathrm{p} \xrightarrow{A_2} \xi[\sigma], \mathrm{q} \wedge \mathrm{Q} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{q})$$

1 Base cases

Show for translation rules T1 to T7 and T10 that for all a and Q such that $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \xrightarrow{a} \mathrm{Q}$ there is some $(A_1,A_2) \in \mathscr{A}^{\sim}$. $a \in A_1$ and some σ,q such that $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{q})$ and $\xi,\mathrm{p} \xrightarrow{a'} \xi[\sigma],\mathrm{q}$ can be derived for all $a' \in A_2$ and $\mathrm{Q} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{q})$.

Translation rule T1: The translation function T_V is partially defined by translation rule

$$\mathrm{T}_V(\xi,\mathbf{broadcast}(ms).\mathrm{p}) = \textstyle\sum_{\mathtt{D}:\mathrm{T}(\mathrm{Set(IP)})} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}},\mathtt{D},\mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi,\mathrm{p}) \text{ where } \mathtt{D} \notin \mathrm{T}(V) \quad \text{T1}$$

For expressions of the form $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathbf{broadcast}(ms).\mathrm{p}) \xrightarrow{a} \mathrm{Q}$, consider the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}},t_{\hat{\mathtt{D}}},\mathrm{T}_\xi(ms)) \xrightarrow{[\![\mathbf{cast}(\mathscr{U}_{\mathrm{IP}},t_{\hat{\mathtt{D}}},\mathrm{T}_\xi(ms))]\!]} \checkmark}\ \text{Axiom}}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}},t_{\hat{\mathtt{D}}},\mathrm{T}_\xi(ms)) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!],\hat{\mathtt{D}},[\![\mathrm{T}_\xi(ms)]\!])} \checkmark}\ \text{Definition 2.1}}{\mathbf{cast}(\mathscr{U}_{\mathrm{IP}},t_{\hat{\mathtt{D}}},\mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!],\hat{\mathtt{D}},[\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\ \text{Seq 1}}{(\mathbf{cast}(\mathscr{U}_{\mathrm{IP}},\mathtt{D},\mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}))[\mathtt{D} := t_{\hat{\mathtt{D}}}] \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!],\hat{\mathtt{D}},[\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\ \text{Substitution}}{\textstyle\sum_{\mathtt{D}:\mathrm{T}(\mathrm{Set(IP)})} \mathbf{cast}(\mathscr{U}_{\mathrm{IP}},\mathtt{D},\mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!],\hat{\mathtt{D}},[\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\ \text{Sum 2}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathbf{broadcast}(ms).\mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!],\hat{\mathtt{D}},[\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\ \text{T1}$$

This derivation is a representative derivation without alternatives (see Definition 4.3). It follows that $a \in \{\, \mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!],\hat{\mathtt{D}},[\![\mathrm{T}_\xi(ms)]\!]) \mid \hat{\mathtt{D}} \in \mathrm{T}(\mathrm{Set(IP)}) \,\}$ and $\mathrm{Q} = \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p})$ by choosing $\sigma = \emptyset$. There is exactly one pair $(A_1,A_2) \in \mathscr{A}^{\sim}$. $a \in A_1 \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathbf{broadcast}(ms).\mathrm{p}) \xrightarrow{A_1} \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p})$:

$$(\{\, \mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!],\hat{\mathtt{D}},[\![\mathrm{T}_\xi(ms)]\!]) \mid \hat{\mathtt{D}} \in \mathrm{T}(\mathrm{Set(IP)}) \,\}, \{\, \mathbf{broadcast}(\xi(ms)) \,\})$$

This pair satisfies the condition that $\xi,\mathbf{broadcast}(ms).\mathrm{p} \xrightarrow{a'} \xi[\sigma],\mathrm{p}$ can be derived for all $a' \in A_2$ (via AWN inference rule Broadcast (T1)), which is sufficient to prove this particular base case.

Translation rule T2: The translation function T_V is partially defined by translation rule

$$\mathrm{T}_V(\xi, \mathbf{groupcast}(\mathit{dests}, \mathit{ms}).\mathrm{p}) = \sum_{\mathtt{D}:\mathrm{T}(\mathrm{Set(IP)})} \mathbf{cast}(\mathrm{T}_\xi(\mathit{dests}), \mathtt{D}, \mathrm{T}_\xi(\mathit{ms})).\mathrm{T}_V(\xi, \mathrm{p}) \text{ where } \mathtt{D} \notin \mathrm{T}(V) \quad \text{T2}$$

For expressions of the form $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{groupcast}(\mathit{dests}, \mathit{ms}).\mathrm{p}) \xrightarrow{a} \mathrm{Q}$, consider the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{cast}(\mathrm{T}_\xi(\mathit{dests}), t_{\hat{\mathtt{D}}}, \mathrm{T}_\xi(\mathit{ms})) \xrightarrow{[\![\mathbf{cast}(\mathrm{T}_\xi(\mathit{dests}), t_{\hat{\mathtt{D}}}, \mathrm{T}_\xi(\mathit{ms}))]\!]} \checkmark}\,\textsc{Axiom}}{\mathbf{cast}(\mathrm{T}_\xi(\mathit{dests}), t_{\hat{\mathtt{D}}}, \mathrm{T}_\xi(\mathit{ms})) \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(\mathit{dests})]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(\mathit{ms})]\!])} \checkmark}\,\text{Definition 2.1}}{\mathbf{cast}(\mathrm{T}_\xi(\mathit{dests}), t_{\hat{\mathtt{D}}}, \mathrm{T}_\xi(\mathit{ms})).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(\mathit{dests})]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(\mathit{ms})]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\,\textsc{Seq 1}}{(\mathbf{cast}(\mathrm{T}_\xi(\mathit{dests}), \mathtt{D}, \mathrm{T}_\xi(\mathit{ms})).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}))[\mathtt{D} := t_{\hat{\mathtt{D}}}] \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(\mathit{dests})]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(\mathit{ms})]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\,\text{Substitution}}{\sum_{\mathtt{D}:\mathrm{T}(\mathrm{Set(IP)})} \mathbf{cast}(\mathrm{T}_\xi(\mathit{dests}), \mathtt{D}, \mathrm{T}_\xi(\mathit{ms})).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(\mathit{dests})]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(\mathit{ms})]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\,\textsc{Sum 2}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{groupcast}(\mathit{dests}, \mathit{ms}).\mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(\mathit{dests})]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(\mathit{ms})]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\,\text{T2}$$

This derivation is a representative derivation without alternatives (see Definition 4.3). It follows that $a \in \{\, \mathbf{cast}([\![\mathrm{T}_\xi(\mathit{dests})]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(\mathit{ms})]\!]) \mid \hat{\mathtt{D}} \in \mathrm{T}(\mathrm{Set(IP)}) \,\}$ and $\mathrm{Q} = \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{p})$ by choosing $\sigma = \emptyset$. There is exactly one pair $(A_1, A_2) \in \mathscr{A}^{\sim}$. $a \in A_1 \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{groupcast}(\mathit{dests}, \mathit{ms}).\mathrm{p}) \xrightarrow{A_1} \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{p})$:

$$(\{\, \mathbf{cast}([\![\mathrm{T}_\xi(\mathit{dests})]\!], \hat{\mathtt{D}}, [\![\mathrm{T}_\xi(\mathit{ms})]\!]) \mid \hat{\mathtt{D}} \in \mathrm{T}(\mathrm{Set(IP)}) \,\}, \{\, \mathbf{groupcast}(\xi(\mathit{dests}), \xi(\mathit{ms})) \,\})$$

This pair satisfies the condition that $\xi, \mathbf{groupcast}(\mathit{dests}, \mathit{ms}).\mathrm{p} \xrightarrow{a'} \equiv \xi[\sigma], \mathrm{p}$ can be derived for all $a' \in A_2$ (via AWN inference rule GROUPCAST (T1)), which is sufficient to prove this particular base case.

Translation rule T3: The translation function T_V is partially defined by translation rule

$$\mathrm{T}_V(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q}) = \mathbf{cast}(\{\mathrm{T}_\xi(dest)\}, \{\mathrm{T}_\xi(dest)\}, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p}) + \neg\mathbf{uni}(\{\mathrm{T}_\xi(dest)\}, \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{q}) \quad \text{T3}$$

For expressions of the form $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q}) \xrightarrow{a} \mathrm{Q}$, consider the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{cast}(\mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(ms)) \xrightarrow{[\![\mathbf{cast}(\mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(ms))]\!]} \checkmark}\,\textsc{Axiom}}{\mathbf{cast}(\mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(ms)) \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(ms)]\!])} \checkmark}\,\text{Definition 2.1}}{\mathbf{cast}(\mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\,\textsc{Seq 1}}{\mathbf{cast}(\mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) + \neg\mathbf{uni}(\mathrm{T}_\xi(\{dest\}), \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q}) \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\,\textsc{Choice 2}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q}) \xrightarrow{\mathbf{cast}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})}\,\text{T3}$$

This representative derivation has one alternative where $a \in \{\, \neg\mathbf{uni}([\![\mathrm{T}_\xi(dest)]\!], [\![\mathrm{T}_\xi(ms)]\!]) \,\}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg\mathbf{uni}(\mathrm{T}_\xi(\{dest\}), \emptyset, \mathrm{T}_\xi(ms)) \xrightarrow{[\![\neg\mathbf{uni}(\mathrm{T}_\xi(\{dest\}), \emptyset, \mathrm{T}_\xi(ms))]\!]} \checkmark}\,\textsc{Axiom}}{\neg\mathbf{uni}(\mathrm{T}_\xi(\{dest\}), \emptyset, \mathrm{T}_\xi(ms)) \xrightarrow{\neg\mathbf{uni}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!])} \checkmark}\,\text{Definition 2.1}}{\neg\mathbf{uni}(\mathrm{T}_\xi(\{dest\}), \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q}) \xrightarrow{\neg\mathbf{uni}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q})}\,\textsc{Seq 1}}{\mathbf{cast}(\mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(\{dest\}), \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) + \neg\mathbf{uni}(\mathrm{T}_\xi(\{dest\}), \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q}) \xrightarrow{\neg\mathbf{uni}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q})}\,\textsc{Choice 4}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q}) \xrightarrow{\neg\mathbf{uni}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q})}\,\text{T3}$$

These derivations are the only representative derivations (see Definition 4.3) for an expression of the form $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q}) \xrightarrow{a} \mathrm{Q}$. It follows that there are exactly two cases that must be distinguished:

- Let $a \in \{\, \mathbf{cast}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(ms)]\!]) \,\}$ and $\mathrm{Q} = \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q}) \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{q})$ by choosing $\sigma = \emptyset$. There is one pair $(A_1, A_2) \in \mathscr{A}^{\sim}$. $a \in A_1 \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{p})$:

  $$(\{\, \mathbf{cast}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(ms)]\!]) \,\}, \{\, \mathbf{unicast}(\xi(dest), \xi(ms)) \,\})$$

  This pair satisfies the condition that $\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q} \xrightarrow{a'} \xi[\sigma], \mathrm{p}$ can be derived for all $a' \in A_2$ (via AWN inference rule UNICAST (T1-1)).
- Let $a \in \{\, \neg\mathbf{uni}([\![\mathrm{T}_\xi(dest)]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!]) \,\}$ and $\mathrm{Q} = \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q}) \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{q})$ by choosing $\sigma = \emptyset$. There is one pair $(A_1, A_2) \in \mathscr{A}^{\sim}$. $a \in A_1 \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q}) \xrightarrow{A_1} \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{q})$:

  $$(\{\, \neg\mathbf{uni}([\![\mathrm{T}_\xi(dest)]\!], [\![\mathrm{T}_\xi(ms)]\!]) \,\}, \{\, \neg\mathbf{unicast}(\xi(dest), \xi(ms)) \,\})$$

  This pair satisfies the condition that $\xi, \mathbf{unicast}(dest, ms).\mathrm{p} \blacktriangleright \mathrm{q} \xrightarrow{a'} \xi[\sigma], \mathrm{q}$ can be derived for all $a' \in A_2$ (via AWN inference rule UNICAST (T1-2)).

By proving both cases this particular base case has been established.

Translation rule T4: The translation function T_V is partially defined by translation rule

$$\mathrm{T}_V(\xi, \mathbf{send}(\mathrm{T}_\xi(ms)).\mathrm{p}) = \mathbf{send}(\emptyset, \emptyset, \mathrm{T}_\xi(ms)).\mathrm{T}_V(\xi, \mathrm{p}) \quad \text{T4}$$

For expressions of the form $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{send}(\mathrm{T}_\xi(ms)).\mathrm{p}) \xrightarrow{a} \mathrm{Q}$, consider the following derivation:

$$\dfrac{\dfrac{\dfrac{}{\mathbf{send}(\emptyset,\emptyset,\mathrm{T}_\xi(dest)) \xrightarrow{[\![\mathbf{send}(\emptyset,\emptyset,\mathrm{T}_\xi(ms))]\!]} \checkmark}\ \text{Axiom}}{\mathbf{send}(\emptyset,\emptyset,\mathrm{T}_\xi(dest)) \xrightarrow{\mathbf{send}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathrm{T}_\xi(ms)]\!])} \checkmark}\ \text{Definition 2.1}}{\mathbf{send}(\emptyset,\emptyset,\mathrm{T}_\xi(dest)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{q}) \xrightarrow{\mathbf{send}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\ \text{Seq 1}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{send}(ms).\mathrm{p}) \xrightarrow{\mathbf{send}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathrm{T}_\xi(ms)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\ \text{T4}$$

This derivation is a representative derivation without alternatives (see Definition 4.3). It follows that $a \in \{\, \mathbf{send}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathrm{T}_\xi(ms)]\!]) \,\}$ and $\mathrm{Q} = \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p})$ by choosing $\sigma = \emptyset$. There is one pair $(A_1, A_2) \in \mathscr{A}^{\sim}$. $a \in A_1 \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{send}(\mathrm{T}_\xi(ms)).\mathrm{p}) \xrightarrow{A_1} \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p})$:

$$(\{\, \mathbf{send}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathrm{T}_\xi(ms)]\!]) \,\}, \{\, \mathbf{send}(\xi(ms)) \,\})$$

This pair satisfies the condition that $\xi, \mathbf{send}(\mathrm{T}_\xi(ms)).\mathrm{p} \xrightarrow{a'} \xi[\sigma],\mathrm{p}$ can be derived for all $a' \in A_2$ (via AWN inference rule Send (T1)), which is sufficient to prove this particular base case.

Translation rule T5: The translation function T_V is partially defined by translation rule

$$\mathrm{T}_V(\xi, \mathbf{deliver}(data).\mathrm{p}) = \sum_{\mathtt{ip}:\mathtt{T}(\mathrm{IP})} \mathbf{del}(\mathtt{ip}, \mathtt{T}_\xi(data)).\mathrm{T}_V(\xi,\mathrm{p}) \text{ where } \mathtt{ip} \notin \mathtt{T}(V) \quad \text{T5}$$

For expressions of the form $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{deliver}(data).\mathrm{p}) \xrightarrow{a} \mathrm{Q}$, consider the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{del}(t_{\hat{\imath}}, \mathtt{T}_\xi(data)) \xrightarrow{[\![\mathbf{del}(t_{\hat{\imath}}, \mathtt{T}_\xi(data))]\!]} \checkmark}\,\text{Axiom}}{\mathbf{del}(t_{\hat{\imath}}, \mathtt{T}_\xi(data)) \xrightarrow{\mathbf{del}(\hat{\imath}, [\![\mathtt{T}_\xi(data)]\!])} \checkmark}\,\text{Definition 2.1}}{\mathbf{del}(t_{\hat{\imath}}, \mathtt{T}_\xi(data)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{q}) \xrightarrow{\mathbf{del}(\hat{\imath}, [\![\mathtt{T}_\xi(data)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\,\text{Seq 1}}{\mathbf{del}(\mathtt{ip}, \mathtt{T}_\xi(data)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{q})\,[\mathtt{ip} := t_{\hat{\imath}}] \xrightarrow{\mathbf{del}(\hat{\imath}, [\![\mathtt{T}_\xi(data)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\,\text{Substitution}}{\sum_{\mathtt{ip}:\mathtt{T}(\mathrm{IP})} \mathbf{del}(\mathtt{ip}, \mathtt{T}_\xi(data)).\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{q}) \xrightarrow{\mathbf{del}(\hat{\imath}, [\![\mathtt{T}_\xi(data)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\,\text{Sum 2}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{deliver}(ms).\mathrm{p}) \xrightarrow{\mathbf{del}(\hat{\imath}, [\![\mathtt{T}_\xi(data)]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p})}\,\text{T5}$$

This derivation is a representative derivation without alternatives (see Definition 4.3). It follows that $a \in \left\{ \mathbf{del}(\hat{\imath}, [\![\mathtt{T}_\xi(data)]\!]) \;\middle|\; \hat{\imath} \in \mathtt{T}(\mathrm{IP}) \right\}$ and $\mathrm{Q} = \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p})$ by choosing $\sigma = \emptyset$. There is one pair $(A_1, A_2) \in \mathscr{A}^{\sim}$. $a \in A_1 \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{deliver}(data).\mathrm{p}) \xrightarrow{A_1} \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p})$:

$$(\left\{ \mathbf{del}(\hat{\imath}, [\![\mathtt{T}_\xi(data)]\!]) \;\middle|\; \hat{\imath} \in \mathtt{T}(\mathrm{IP}) \right\}, \{ \mathbf{deliver}(\xi(data)) \})$$

This pair satisfies the condition that $\xi, \mathbf{deliver}(data).\mathrm{p} \xrightarrow{a'} \xi[\sigma],\mathrm{p}$ can be derived for all $a' \in A_2$ (via AWN inference rule Deliver (T1)), which is sufficient to prove this particular base case.

Translation rule T6: The translation function T_V is partially defined by translation rule

$$\mathrm{T}_V(\xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p}) = \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set(IP)}),\mathrm{T}(\mathtt{msg}):\mathrm{T}(\mathrm{MSG})} \mathbf{receive}(\mathtt{D},\mathtt{D'},\mathrm{T}(\mathtt{msg})).\mathrm{T}_{V\cup\{\mathtt{msg}\}}(\xi^{\setminus\mathtt{msg}},\mathrm{p}) \text{ where } \mathtt{D},\mathtt{D'} \notin \mathrm{T}(V) \quad \text{T6}$$

For expressions of the form $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p}) \xrightarrow{a} \mathrm{Q}$, consider the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{receive}(t_{\hat{\mathtt{D}}},t_{\hat{\mathtt{D}}'},t_{\mathrm{U}(m)}) \xrightarrow{[\![\mathbf{receive}(t_{\hat{\mathtt{D}}},t_{\hat{\mathtt{D}}'},t_{\mathrm{U}(m)})]\!]} \checkmark}\ \text{Axiom}}{\mathbf{receive}(t_{\hat{\mathtt{D}}},t_{\hat{\mathtt{D}}'},t_{\mathrm{U}(m)}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',[\![t_{\mathrm{U}(m)}]\!])} \checkmark}\ \text{Definition 2.1}}{\mathbf{receive}(t_{\hat{\mathtt{D}}},t_{\hat{\mathtt{D}}'},t_{\mathrm{U}(m)}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',\mathrm{U}(m))} \checkmark}\ \text{Definition of } t}{\mathbf{receive}(t_{\hat{\mathtt{D}}},t_{\hat{\mathtt{D}}'},t_{\mathrm{U}(m)}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg}:=m],\mathrm{p}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',\mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg}:=m],\mathrm{p})}\ \text{Seq 1}}{\mathbf{receive}(t_{\mathtt{D}},t_{\mathtt{D}'},t_{\mathrm{U}(m)}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg}:=m],\mathrm{p}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',\mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg}:=m],\mathrm{p})}}{\mathbf{receive}(t_{\mathtt{D}},t_{\mathtt{D}'},t_{\mathrm{U}(m)}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi^{\setminus\mathtt{msg}}[\mathtt{msg}:=m],\mathrm{p}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',\mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg}:=m],\mathrm{p})}\ \mathtt{msg}\notin\mathrm{DOM}(\xi^{\setminus\mathtt{msg}})\ \text{Lemma 4.6}}{\Big(\mathbf{receive}(t_{\mathtt{D}},t_{\mathtt{D}'},\mathrm{T}(\mathtt{msg})).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi^{\setminus\mathtt{msg}},\mathrm{p})\Big)[\mathrm{T}(\mathtt{msg}):=t_{\mathrm{U}(m)}] \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',\mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg}:=m],\mathrm{p})}\ \text{Substitution}}{\Big(\mathbf{receive}(\mathtt{D},\mathtt{D'},\mathrm{T}(\mathtt{msg})).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi^{\setminus\mathtt{msg}},\mathrm{p})\Big)[\mathtt{D}:=t_{\hat{\mathtt{D}}},\mathtt{D'}:=t_{\hat{\mathtt{D}}'},\mathrm{T}(\mathtt{msg}):=t_{\mathrm{U}(m)}] \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',\mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg}:=m],\mathrm{p})}\ \text{Sum 2 (3 times)}}{\textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set(IP)}),\mathrm{T}(\mathtt{msg}):\mathrm{T}(\mathrm{MSG})} \mathbf{receive}(\mathtt{D},\mathtt{D'},\mathrm{T}(\mathtt{msg})).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi^{\setminus\mathtt{msg}},\mathrm{p}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',\mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg}:=m],\mathrm{p})}\ \text{T6}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',\mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{msg}\}}(\xi[\mathtt{msg}:=m],\mathrm{p})}$$

$$\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p}) \xrightarrow{\mathbf{receive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',\mathrm{U}(m))} \mathrm{T}_{\mathrm{DOM}(\xi[\mathtt{msg}:=m])}(\xi[\mathtt{msg}:=m],\mathrm{p})$$

This derivation is a representative derivation without alternatives (see Definition 4.3). It follows that $a = \mathbf{receive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',\mathrm{U}(m))$ and $\mathrm{Q} = \mathrm{T}_{\mathrm{DOM}(\xi[\mathtt{msg}:=m])}(\xi[\mathtt{msg} := m],\mathrm{p}) \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p})$ by choosing $\sigma = [\mathtt{msg} := m]$. There is only one candidate for $(A_1,A_2) \in \mathscr{A}^{\sim}.\ a \in A_1 \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p}) \xrightarrow{A_1} \mathrm{Q}$, namely

$$(\{\, \mathbf{receive}(\hat{\mathtt{D}},\hat{\mathtt{D}}',\mathrm{U}(m)) \mid \hat{\mathtt{D}},\hat{\mathtt{D}}' \in \mathrm{T}(\mathrm{Set(IP)}) \,\},\{\, \mathbf{receive}(m) \,\})$$

This candidate satisfies the condition that $\xi, \mathbf{receive}(\mathtt{msg}).\mathrm{p} \xrightarrow{a'} \xi[\sigma],\mathrm{p}$ can be derived for all $a' \in A_2$ (via AWN inference rule Receive (T1)), which is sufficient to prove this particular base case.

Translation rule T7: The translation function T_V is partially defined by translation rule

$$\mathrm{T}_V(\xi, [\![\mathtt{var} := exp]\!]\ \mathrm{p}) = \textstyle\sum_{\mathtt{tmp}:\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathtt{tmp} = \mathrm{T}_\xi(exp)) \to \sum_{\mathrm{T}(\mathtt{var}):\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathrm{T}(\mathtt{var}) = \mathtt{tmp}) \to \quad \text{T7}$$
$$\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{V\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}},\mathrm{p}) \text{ where } \mathtt{tmp} \notin \mathrm{T}(V)$$

Let t_c and t_d be two mCRL2 expressions such that $[\![t_c = t_d]\!] = \mathit{true}$.
For expressions of the form $\mathrm{T}_V(\xi, [\![\mathtt{var} := exp]\!]\ \mathrm{p}) \xrightarrow{a} \mathrm{Q}$, consider the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}) \xrightarrow{[\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy})]} \checkmark}\,\text{Axiom}}{\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \checkmark}\,\text{Definition 2.1}}{\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := b],\mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := b],\mathrm{p})}\,\text{Seq 1}}{\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}}[\mathtt{var} := b],\mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := b],\mathrm{p})}\,\text{Guard 2}}{[\![t_c = t_d]\!] = \mathit{true}}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(t_c = t_d) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}}[\mathtt{var} := b],\mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := b],\mathrm{p})}\,\text{Guard 2}}{(t_c = t_d) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}},\mathrm{p})\,[\mathrm{T}(\mathtt{var}) := t_{\mathrm{U}(b)}] \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := b],\mathrm{p})}\,\text{Lemma 4.6}}{\mathtt{var} \notin \mathrm{DOM}(\xi^{\setminus\mathtt{var}})}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(t_c = t_d) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}},\mathrm{p})\,[\mathrm{T}(\mathtt{var}) := t_c] \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := b],\mathrm{p})}\,\text{Choose } c = \mathrm{U}(b)}{\mathrm{T}(\mathtt{var}) \notin \mathrm{FV}(t_c = t_d)}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Big((\mathrm{T}(\mathtt{var}) = t_d) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}},\mathrm{p})\Big)\,[\mathrm{T}(\mathtt{var}) := t_c] \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := b],\mathrm{p})}\,\text{Sum 2}}{\sum_{\mathrm{T}(\mathtt{var}):\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathrm{T}(\mathtt{var}) = t_d) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}},\mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := b],\mathrm{p})}\,\text{Guard 2}}{[\![t_d = \mathrm{T}_\xi(exp)]\!] = \mathit{true}}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(t_d = \mathrm{T}_\xi(exp)) \to \sum_{\mathrm{T}(\mathtt{var}):\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathrm{T}(\mathtt{var}) = t_d) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}},\mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := b],\mathrm{p})}\,\text{Sum 2}}{\Big((\mathtt{tmp} = \mathrm{T}_\xi(exp)) \to \sum_{\mathrm{T}(\mathtt{var}):\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathrm{T}(\mathtt{var}) = \mathtt{tmp}) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}},\mathrm{p})\Big)\,[\mathtt{tmp} := t_d] \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := b],\mathrm{p})}\,\text{Sum 2}}{\sum_{\mathtt{tmp}:\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathtt{tmp} = \mathrm{T}_\xi(exp)) \to \sum_{\mathrm{T}(\mathtt{var}):\mathrm{sort}(\mathrm{T}(\mathtt{var}))}(\mathrm{T}(\mathtt{var}) = \mathtt{tmp}) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi^{\setminus\mathtt{var}},\mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := b],\mathrm{p})}\,\text{T7}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, [\![\mathtt{var} := exp]\!]\mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathtt{var}\}}(\xi[\mathtt{var} := b],\mathrm{p})}$$
$$\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, [\![\mathtt{var} := exp]\!]\mathrm{p}) \xrightarrow{\mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])} \mathrm{T}_{\mathrm{DOM}(\xi[\mathtt{var}:=b])}(\xi[\mathtt{var} := b],\mathrm{p})$$

This derivation is a representative derivation without alternatives (see Definition 4.3). Furthermore, $[\![t_c = t_d]\!] = \mathit{true} \wedge [\![t_{\mathrm{U}(b)} = \mathrm{T}_\xi(exp)]\!] = \mathit{true} \Rightarrow [\![t_{\mathrm{U}(b)}]\!] = [\![\mathrm{T}_\xi(exp)]\!]$ – note that the derivation is impossible without these premises! – which means that $b = \xi(exp)$ according to Lemma 4.3. It follows that $a = \mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])$ and $\mathrm{Q} = \mathrm{T}_{\mathrm{DOM}(\xi[\mathtt{var}:=\xi(exp)])}(\xi[\mathtt{var} := \xi(exp)],\mathrm{p}) \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p})$ by choosing $\sigma = [\mathtt{var} := \xi(exp)]$. There is only one candidate for $(A_1,A_2) \in \mathscr{A}\ .\ a \in A_1 \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, [\![\mathtt{var} := exp]\!]\ \mathrm{p}) \xrightarrow{A_1} \mathrm{Q}$, namely

$$(\{\, \mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m)) \,\},\{\, \tau \,\})$$

This candidate satisfies the condition that $\xi, [\![\mathtt{var} := exp]\!]\ \mathrm{p} \xrightarrow{a'} \xi[\sigma],\mathrm{p}$ can be derived for all $a' \in A_2$ (via AWN inference rule ASSIGNMENT (T1)), which is sufficient to prove this particular base case.

Translation rule T10: The translation function T_V is partially defined by translation rule

$$\mathrm{T}_V(\xi,[\phi]\mathrm{p}) = \sum_{\mathrm{T}(\mathrm{Fv}(\phi)\setminus V)} \mathrm{T}_\xi(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{V\cup\mathrm{Fv}(\phi)}(\xi,\mathrm{p}) \quad \text{T10}$$

Now consider the following derivation for expressions of the form $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,[\phi]\mathrm{p}) \xrightarrow{a} \mathrm{Q}$, where $\sigma = [\mathrm{q}_1 := e_1,\cdots,\mathrm{q}_n := e_n]$ such that $\zeta = \xi[\sigma]$ and $\mathrm{DOM}(\sigma) = \mathrm{Fv}(\phi)\setminus\mathrm{DOM}(\xi)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}) \xrightarrow{[\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy})]} \checkmark}\text{Axiom}}{\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \checkmark}\text{Definition 2.1}}{\mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\zeta,\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\zeta,\mathrm{p})}\text{Seq 1}}{[\![\mathrm{T}_\zeta(\phi)]\!] = \mathit{true} \quad \mathrm{T}_\zeta(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\zeta,\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\zeta,\mathrm{p})}\text{Guard 2}}{\mathrm{T}_\zeta(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\zeta,\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\zeta,\mathrm{p})}\text{Definition of } \zeta \text{ (3 times)}}{\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}\cap\mathrm{DOM}(\xi) = \emptyset \quad \mathrm{T}_{\xi[\mathrm{q}_i:=e_1,\cdots,\mathrm{q}_n:=e_n]}(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\xi[\mathrm{q}_1 := e_1,\cdots,\mathrm{q}_n := e_n],\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}\text{Lemma 4.5}}{\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}\cap\mathrm{DOM}(\xi) = \emptyset \quad \mathrm{T}_\xi(\phi)[\mathrm{T}(\mathrm{q}_1) := t_{\mathrm{U}(e_1)},\cdots,\mathrm{T}(\mathrm{q}_n) := t_{\mathrm{U}(e_n)}] \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\xi[\mathrm{q}_1 := e_1,\cdots,\mathrm{q}_n := e_n],\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}\text{Lemma 4.6}}{\mathrm{T}_\xi(\phi)[\mathrm{T}(\mathrm{q}_1) := t_{\mathrm{U}(e_1)},\cdots,\mathrm{T}(\mathrm{q}_n) := t_{\mathrm{U}(e_n)}] \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\xi,\mathrm{p})[\mathrm{T}(\mathrm{q}_1) := t_{\mathrm{U}(e_1)},\cdots,\mathrm{T}(\mathrm{q}_n) := t_{\mathrm{U}(e_n)}] \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}\text{Choose } t_{d_i} = t_{\mathrm{U}(e_i)}}{\mathrm{T}_\xi(\phi)[\mathrm{T}(\mathrm{q}_1) := t_{d_1},\cdots,\mathrm{T}(\mathrm{q}_n) := t_{d_n}] \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\xi,\mathrm{p})[\mathrm{T}(\mathrm{q}_1) := t_{d_1},\cdots,\mathrm{T}(\mathrm{q}_n) := t_{d_n}] \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\left(\mathrm{T}_\xi(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\xi,\mathrm{p})\right)[\mathrm{T}(\mathrm{q}_1) := t_{d_1},\cdots,\mathrm{T}(\mathrm{q}_n) := t_{d_n}] \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}{\sum_{\{\mathrm{T}(\mathrm{q}_1),\cdots,\mathrm{T}(\mathrm{q}_n)\}} \mathrm{T}_\xi(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\xi,\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}\text{Sum 2 } (n \text{ times})}{\sum_{\mathrm{T}(\{\mathrm{q}_1,\cdots,\mathrm{q}_n\})} \mathrm{T}_\xi(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\{\mathrm{q}_1,\cdots,\mathrm{q}_n\}}(\xi,\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}\text{Definition of } \mathrm{DOM}(\sigma) \text{ (2 times)}}{\sum_{\mathrm{T}(\mathrm{Fv}(\phi)\setminus\mathrm{DOM}(\xi))} \mathrm{T}_\xi(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup(\mathrm{Fv}(\phi)\setminus\mathrm{DOM}(\xi))}(\xi,\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}$$

$$\dfrac{\sum_{\mathrm{T}(\mathrm{Fv}(\phi)\setminus\mathrm{DOM}(\xi))} \mathrm{T}_\xi(\phi) \to \mathbf{t}(\emptyset,\emptyset,\mathtt{msg}_{\,dummy}).\mathrm{T}_{\mathrm{DOM}(\xi)\cup\mathrm{Fv}(\phi)}(\xi,\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,[\phi]\mathrm{p}) \xrightarrow{\mathbf{t}([\emptyset],[\emptyset],[\mathtt{msg}_{\,dummy}])} \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p})}\text{T10}$$

This derivation is a representative derivation without alternatives (see Definition 4.3). It follows that $a = \mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!])$ and $\mathrm{Q} = \mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta,\mathrm{p}) \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p})$ because $\zeta = \xi[\sigma]$. There is only one candidate for $(A_1,A_2) \in \mathscr{A}^{\sim}.\, a \in A_1 \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,[\phi]\mathrm{p}) \xrightarrow{A_1} \mathrm{Q}$, namely

$$(\{\, \mathbf{t}([\![\emptyset]\!],[\![\emptyset]\!],[\![\mathtt{msg}_{\,dummy}]\!]) \,\},\{\, \tau \,\})$$

Note that $[\![\mathrm{T}_\zeta(\phi)]\!] = \mathit{true}$ if and only if $\zeta(\phi) = \mathit{true}$ according to Lemma 4.4. This candidate therefore satisfies the condition that $\xi,[\phi]\mathrm{p} \xrightarrow{a'} \xi[\sigma],\mathrm{p}$ can be derived for all $a' \in A_2$ (via AWN inference rule GUARD (T1)), which is sufficient to prove this particular base case.

2 Induction step

Given only its side conditions and the induction hypothesis, show for translation rules T8 and T9 that for all a and Q such that $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \xrightarrow{a} \mathrm{Q}$ there is some $(A_1,A_2) \in \mathscr{A}^{\sim} \,.\, a \in A_1$ and some σ,q such that $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\mathrm{DOM}(\xi[\sigma]),\mathrm{q})$ and $\mathrm{p} \xrightarrow{a'} \mathrm{q}$ can be derived for all $a' \in A_2$ and $\mathrm{Q} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\mathrm{DOM}(\xi[\sigma]),\mathrm{q})$.

Translation rule T8: The translation function T_V is partially defined by translation rule

$$\mathrm{T}_V(\xi,\mathrm{X}(exp_1,\cdots,exp_n)) = \mathrm{X}(\mathrm{T}_\xi(exp_1),\cdots,\mathrm{T}_\xi(exp_n)) \quad \text{T8}$$
$$\text{where } \mathrm{T}(\mathrm{X}(\mathtt{var}_1,\cdots,\mathtt{var}_n) \stackrel{\text{def}}{=} \mathrm{p}) = \mathrm{X}(\mathrm{T}(\mathtt{var}_1) : \mathrm{sort}(\mathrm{T}_\xi(exp_1)),\cdots,\mathrm{T}(\mathtt{var}_n) : \mathrm{sort}(\mathrm{T}_\xi(exp_n))) \stackrel{\text{def}}{=} \mathrm{T}_{\{\mathtt{var}_1,\cdots,\mathtt{var}_n\}}(\emptyset,\mathrm{p})$$

The induction hypothesis states that recursions T_V occurring in $\mathrm{X}(\mathrm{T}_\xi(exp_1),\cdots,\mathrm{T}_\xi(exp_n))$ are related. In particular, the induction hypothesis is used here to relate arbitrary instantiations of process calls:

$$\begin{aligned}
&\forall a,\mathrm{Q} \,.\, \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)],\mathrm{p}) \xrightarrow{a} \mathrm{Q} \Rightarrow \exists (A_1,A_2) \in \mathscr{A}^{\sim},\mathrm{q},\sigma \,.\, a \in A_1 \\
&\quad \wedge \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)],\mathrm{p}) \xrightarrow{A_1} \equiv \\
&\qquad\qquad \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup\mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q}) \\
&\quad \wedge \emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)],\mathrm{p} \xrightarrow{A_2} \emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q} \\
&\quad \wedge \mathrm{Q} \equiv \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup\mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q})
\end{aligned}$$

The following derivation in AWN can be based on the fourth line of the instantiated induction hypothesis:

$$\dfrac{\dfrac{}{\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)],\mathrm{p} \xrightarrow{a'} \emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q}}\,\text{Induction hypothesis} \quad \text{Definition of X}}{\xi,\mathrm{X}(exp_1,\cdots,exp_n) \xrightarrow{a'} \emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q}}\,\text{RECURSION (T1)}$$

This derivation holds for all $a' \in A_2$. Similarly, in mCRL2, it happens to be the case that

$$\text{Definition of X}\;\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)],\mathrm{p}) \xrightarrow{a} \equiv \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup\mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q})}\,\text{Induction hypothesis}}{\mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}}(\emptyset,\mathrm{p})[\mathrm{T}(\mathtt{d}_1) := t_{\mathrm{U}(\xi(exp_1))},\cdots,\mathrm{T}(\mathtt{d}_n) := t_{\mathrm{U}(\xi(exp_n))}] \xrightarrow{a} \equiv \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup\mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q})}\,\text{Lemma 4.6}}{\mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}}(\emptyset,\mathrm{p})[\mathrm{T}(\mathtt{d}_1) := \mathrm{T}_\xi(exp_1),\cdots,\mathrm{T}(\mathtt{d}_n) := \mathrm{T}_\xi(exp_n)] \xrightarrow{a} \equiv \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup\mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q})}\,\begin{matrix}\text{Theorem 4.1}\\ \text{Lemma 4.3}\end{matrix}}{\mathrm{X}(\mathrm{T}_\xi(exp_1),\cdots,\mathrm{T}_\xi(exp_n)) \xrightarrow{a} \equiv \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup\mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q})}\,\text{RECURSION 2}}{\mathrm{T}_V(\xi,\mathrm{X}(exp_1,\cdots,exp_n)) \xrightarrow{a} \equiv \mathrm{T}_{\{\mathtt{d}_1,\cdots,\mathtt{d}_n\}\cup\mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1),\cdots,\mathtt{d}_n := \xi(exp_n)][\sigma],\mathrm{q})}\,\text{T8}$$

for all $a \in A_1$. This derivation is a representative derivation without alternatives (see Definition 4.3).

Because the induction hypothesis is used to express the relationship between *arbitrary* instantiations of process calls, the first line of this derivation plus the inference rules of mCRL2 are sufficient to generate all possible behavior of an mCRL2 process call. The actions

available to $\mathrm{T}_V(\xi, \mathrm{X}(exp_1, \cdots, exp_n))$ in mCRL2 can therefore be mimicked by AWN, and so both derivations can be combined into an new equation matching the induction hypothesis:

$$\begin{aligned}
&\forall a, \mathrm{Q} \,.\, \mathrm{T}_V(\xi, \mathrm{X}(exp_1, \cdots, exp_n)) \xrightarrow{a} \mathrm{Q} \Rightarrow \exists (A_1, A_2) \in \mathscr{A}^{\sim}, \mathrm{q}, \sigma \,.\, a \in A_1 \\
&\qquad \wedge \mathrm{T}_V(\xi, \mathrm{X}(exp_1, \cdots, exp_n)) \xrightarrow{A_1} \equiv \\
&\qquad\qquad \mathrm{T}_{\{\mathtt{d}_1, \cdots, \mathtt{d}_n\} \cup \mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1), \cdots, \mathtt{d}_n := \xi(exp_n)][\sigma], \mathrm{q}) \\
&\qquad \wedge \emptyset[\mathtt{d}_1 := \xi(exp_1), \cdots, \mathtt{d}_n := \xi(exp_n)], \mathrm{p} \xrightarrow{A_2} \emptyset[\mathtt{d}_1 := \xi(exp_1), \cdots, \mathtt{d}_n := \xi(exp_n)][\sigma], \mathrm{q} \\
&\qquad \wedge \mathrm{Q} \equiv \mathrm{T}_{\{\mathtt{d}_1, \cdots, \mathtt{d}_n\} \cup \mathrm{DOM}(\sigma)}(\emptyset[\mathtt{d}_1 := \xi(exp_1), \cdots, \mathtt{d}_n := \xi(exp_n)][\sigma], \mathrm{q})
\end{aligned}$$

This confirms the induction step of the proof of Lemma 4.8.

Translation rule T9: The translation function $\mathrm{T}_{\mathrm{DOM}(\xi)}$ is partially defined by translation rule

$$\mathrm{T}_V(\xi, \mathrm{p} + \mathrm{q}) = \mathrm{T}_V(\xi, \mathrm{p}) + \mathrm{T}_V(\xi, \mathrm{q}) \quad \text{T9}$$

Suppose that $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p} + \mathrm{q}) \xrightarrow{a} \mathrm{P}'$ for some a and P'. According to the semantics of mCRL2 (in particular inference rules CHOICE 2 and CHOICE 4) this can only be the case if $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{a} \mathrm{P}'$ or $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q}) \xrightarrow{a} \mathrm{P}'$ or both:

$$\frac{\dfrac{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{a} \mathrm{P}'}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) + \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q}) \xrightarrow{a} \mathrm{P}'}\ \text{CHOICE 2}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p} + \mathrm{q}) \xrightarrow{a} \mathrm{P}'}\ \text{T9}$$

$$\frac{\dfrac{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q}) \xrightarrow{a} \mathrm{P}'}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) + \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q}) \xrightarrow{a} \mathrm{P}'}\ \text{CHOICE 4}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p} + \mathrm{q}) \xrightarrow{a} \mathrm{P}'}\ \text{T9}$$

That is,

$$\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p} + \mathrm{q}) \xrightarrow{a} \mathrm{P}' \Rightarrow \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{a} \mathrm{P}' \vee \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{q}) \xrightarrow{a} \mathrm{P}' \tag{D.1}$$

According to the induction hypothesis, $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \xrightarrow{a} \mathrm{P}'$ implies that there must exist some $(A_1,A_2) \in \mathscr{A}^{\sim}$ and σ,p' such that $a \in A_1 \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p}') \wedge \xi,\mathrm{p} \xrightarrow{A_2} \xi[\sigma],\mathrm{p}' \wedge \mathrm{P}' \equiv \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}')$, which makes exactly the following derivations possible (in mCRL2 and AWN, respectively):

$$\dfrac{\dfrac{\dfrac{}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p}')}\ \text{Induction hypothesis}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) + \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{q}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p}')}\ \textsc{Choice 2}}{\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}+\mathrm{q}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p}')}\ \textsc{T9}$$

$$\dfrac{\dfrac{}{\xi,\mathrm{p} \xrightarrow{A_2} \xi[\sigma],\mathrm{p}'}\ \text{Induction hypothesis}}{\xi,\mathrm{p}+\mathrm{q} \xrightarrow{A_2} \xi[\sigma],\mathrm{p}'}\ \textsc{Choice (T1-1)}$$

and therefore

$$\begin{aligned}
&\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \xrightarrow{a} \equiv \mathrm{P}' \Rightarrow \exists (A_1,A_2) \in \mathscr{A}^{\sim},\mathrm{p}' \,.\, a \in A_1 \\
&\quad \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}+\mathrm{q}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p}') \\
&\quad \wedge \xi,\mathrm{p}+\mathrm{q} \xrightarrow{A_2} \xi[\sigma],\mathrm{p}' \wedge \mathrm{P}' \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p}')
\end{aligned}$$

Similar reasoning can be applied to $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}) \xrightarrow{a} \equiv \mathrm{P}'$, resulting in

$$\begin{aligned}
&\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{q}) \xrightarrow{a} \equiv \mathrm{P}' \Rightarrow \exists (A_1,A_2) \in \mathscr{A}^{\sim},\mathrm{p}' \,.\, a \in A_1 \\
&\quad \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}+\mathrm{q}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p}') \\
&\quad \wedge \xi,\mathrm{p}+\mathrm{q} \xrightarrow{A_2} \xi[\sigma],\mathrm{p}' \wedge \mathrm{P}' \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p}')
\end{aligned}$$

Combining both of those results via Equation D.1, it then becomes clear that

$$\begin{aligned}
&\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}+\mathrm{q}) \xrightarrow{a} \equiv \mathrm{P}' \Rightarrow \exists (A_1,A_2) \in \mathscr{A}^{\sim},\mathrm{p}' \,.\, a \in A_1 \\
&\quad \wedge \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi,\mathrm{p}+\mathrm{q}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p}') \\
&\quad \wedge \xi,\mathrm{p}+\mathrm{q} \xrightarrow{A_2} \xi[\sigma],\mathrm{p}' \wedge \mathrm{P}' \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma],\mathrm{p}')
\end{aligned}$$

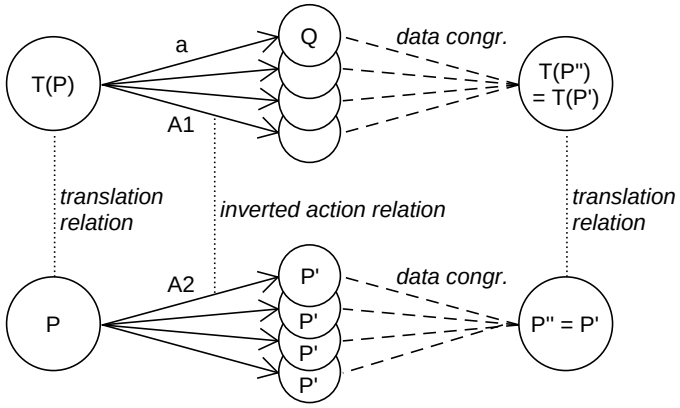
which is sufficient to prove this particular base case.

Appendix E

Complete proof of Lemma 4.9

Lemma: Let $\widetilde{\mathrm{T}}^{\smile}$ be the converse of translation relation $\widetilde{\mathrm{T}}$ from Equation 2.1 and let $\mathscr{A}^{\smile}$ be the converse of relation $\mathscr{A}$ from Table 2.10. Then $\widetilde{\mathrm{T}}^{\smile}$ is a strong $\mathscr{A}^{\smile}$-warped simulation of mCRL2 expressions $\mathrm{T}(\mathrm{P})$ by AWN expressions P for all AWN expressions P.

Proof: For all actions that $\mathrm{T}(\mathrm{P})$ can do, it must be shown that the resulting state is data congruent with $\mathrm{T}(\mathrm{P}')$. A depiction of the requirements for the proof in this direction is given below:



Note that data congruence is implied by the assumption that $\mathrm{P}'' = \mathrm{P}'$. In combination with Lemma 3.1, this means that for Definition 4.11 to apply it is sufficient to prove that

$$\forall \mathrm{P}, a, \mathrm{Q}' \,.\, \mathrm{T}(\mathrm{P}) \xrightarrow{a} \mathrm{Q} \Rightarrow \exists (A_1, A_2) \in \mathscr{A}^{\smile}, \mathrm{P}' \,.\, a \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}') \wedge \mathrm{P} \xrightarrow{A_2} \mathrm{P}' \wedge \mathrm{Q} \equiv \mathrm{T}(\mathrm{P}')$$

The proof of Equation 2.20 is provided by structural induction over the translation rules of mCRL2, which is possible because these yield all possible mCRL2 expressions that can result from a translation by T.

Induction hypothesis: For all recursions $\mathrm{T}(\mathrm{P})$ that are part of a translation rule defining T, it holds that

$$\forall a, \mathrm{Q} \,.\, \mathrm{T}(\mathrm{P}) \xrightarrow{a} \mathrm{Q} \Rightarrow \exists (A_1, A_2) \in \mathscr{A}^{\smile}, \mathrm{P}' \,.\, a \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}') \wedge \mathrm{P} \xrightarrow{A_2} \mathrm{P}' \wedge \mathrm{Q} \equiv \mathrm{T}(\mathrm{P}')$$

1 Base cases

Show for translation rules T11 and T12 that for all a and Q such that $\mathrm{T}(\mathrm{P}) \xrightarrow{a} \mathrm{Q}$ there is some $(A_1, A_2) \in \mathscr{A}^{\sim}$. $a \in A_1$ and some P' such that $\mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}')$ and $\mathrm{P} \xrightarrow{a'} \mathrm{P}$ can be derived for all $a' \in A_2$ and $\mathrm{Q} \equiv \mathrm{T}(\mathrm{P}')$.

Translation rule T11: This translation rule is an exceptional case because a process definition by itself cannot do any transitions, and providing a proof for Lemma 4.9 is therefore nonsensical. Related to T11 is process recursion, for which a proof *does* make sense; see the Lemma 4.8-proof for Translation rule T8.

Translation rule T12: The translation function T is partially defined by translation rule T12:

$$\mathrm{T}(\xi, \mathrm{p}) = \mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p})$$

Suppose that $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{a} \mathrm{Q}$. Then, according to Lemma 4.8, there exists some $(A_1, A_2) \in \mathscr{A}^{\sim}$. $a \in A_1$ and some σ, p' such that $\mathrm{T}_{\mathrm{DOM}(\xi)}(\xi, \mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{p}')$ and $\xi, \mathrm{p} \xrightarrow{a'} \xi[\sigma], \mathrm{p}'$ for all $a' \in A_2$ and $\mathrm{Q} \equiv \mathrm{T}_{\mathrm{DOM}(\xi[\sigma])}(\xi[\sigma], \mathrm{p}')$.
Define $\zeta = \xi[\sigma]$. Q is also data congruent with $\mathrm{T}_{\mathrm{DOM}(\zeta)}(\zeta, \mathrm{p}')$ which, as stated by rule T12, equals $\mathrm{T}(\zeta, \mathrm{p}')$. As a result, there must exist some $(A_1, A_2) \in \mathscr{A}^{\sim}$. $a \in A_1$ and some p' so that $\mathrm{T}(\xi, \mathrm{p}) \xrightarrow{A_1} \equiv \mathrm{T}(\zeta, \mathrm{p}')$ and $\xi, \mathrm{p} \xrightarrow{a'} \zeta, \mathrm{p}'$ for all $a' \in A_2$ and $\mathrm{Q} \equiv \mathrm{T}(\zeta, \mathrm{p}')$, proving this particular base case.

2 Induction step

Given only their side conditions and the induction hypothesis, show for translation rules T13 to T16 that for all a and Q such that $\mathrm{T}(\mathrm{P}) \xrightarrow{a} \mathrm{Q}$ there is some $(A_1, A_2) \in \mathscr{A}^{\sim}$. $a \in A_1$ and some P' such that $\mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}')$ and $\mathrm{P} \xrightarrow{a'} \mathrm{P}'$ can be derived for all $a' \in A_2$ and $\mathrm{Q} \equiv \mathrm{T}(\mathrm{P}')$.

Translation rule T13: The translation function T is partially defined by translation rule

$$\mathrm{T}(\mathrm{P} \langle\!\langle \mathrm{Q}) = \nabla_V \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \quad \text{T13}$$
$$\text{where } V = \{\mathbf{t}, \mathbf{cast}, \neg\mathbf{uni}, \mathbf{send}, \mathbf{del}, \mathbf{receive}\}$$

Consider Pairs 2.10 and 2.8 from Table 2.10:

$$(\{\, \mathbf{receive}(m) \,\}, \{\, \mathbf{receive}(\hat{\mathrm{D}}, \hat{\mathrm{D}}', \mathrm{U}(m)) \mid \hat{\mathrm{D}}, \hat{\mathrm{D}}' \in \mathrm{T}(\mathrm{Set}(\mathrm{IP})) \,\})$$
$$(\{\, \mathbf{send}(\xi(ms)) \,\}, \{\, \mathbf{send}([\![\emptyset]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!]) \,\})$$

Let R_1 and S_1 be defined as the second members of these pairs; that is, let

$$R_1 \stackrel{\text{def}}{=} \{\, \mathbf{receive}(\hat{\mathrm{D}}, \hat{\mathrm{D}}', \mathrm{U}(m)) \mid \hat{\mathrm{D}}, \hat{\mathrm{D}}' \in \mathrm{T}(\mathrm{Set}(\mathrm{IP})) \,\}$$
$$S_1 \stackrel{\text{def}}{=} \{\, \mathbf{send}([\![\emptyset]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!]) \,\}$$

for some m, $\xi(ms) \in \mathrm{MSG}$.
The following cases are distinguished:

1: $\mathrm{T}(\mathrm{P})$ does a transition $a \in R_1$, $\mathrm{T}(\mathrm{Q})$ does a transition $b \in S_1$, and $\mathrm{U}(m) = [\![\mathrm{T}_\xi(ms)]\!]$.
2: $\mathrm{T}(\mathrm{P})$ does a transition $a \in R_1$, $\mathrm{T}(\mathrm{Q})$ does a transition $b \in S_1$, and $\mathrm{U}(m) \neq [\![\mathrm{T}_\xi(ms)]\!]$.
3: $\mathrm{T}(\mathrm{P})$ does a transition a, $\mathrm{T}(\mathrm{Q})$ does a transition b, and $\underline{a} \neq \mathbf{receive}$ or $\underline{b} \neq \mathbf{send}$ or both.
4: $\mathrm{T}(\mathrm{P})$ does a transition a where $\underline{a} = \mathbf{receive}$ and $\mathrm{T}(\mathrm{Q})$ does not do a transition.
5: $\mathrm{T}(\mathrm{P})$ does a transition a where $\underline{a} \neq \mathbf{receive}$ and $\mathrm{T}(\mathrm{Q})$ does not do a transition.
6: $\mathrm{T}(\mathrm{P})$ does not do a transition and $\mathrm{T}(\mathrm{Q})$ does a transition b where $\underline{b} = \mathbf{send}$.
7: $\mathrm{T}(\mathrm{P})$ does not do a transition and $\mathrm{T}(\mathrm{Q})$ does a transition b where $\underline{b} \neq \mathbf{send}$.

Note that these cases cover all combinations of behavior of $\mathrm{T}(\mathrm{P})$ and $\mathrm{T}(\mathrm{Q})$.
The proof is provided below for each of the cases:

1: Suppose that $\mathrm{T}(\mathrm{P})$ does a transition $a \in R_1$, $\mathrm{T}(\mathrm{Q})$ does a transition $b \in S_1$, and $\mathrm{U}(m) = [\![\mathrm{T}_\xi(ms)]\!]$. Following the induction hypothesis and eliminating pairs from the action relation $\mathscr{A}$ in Table 2.10 that do not match the transition labels from R_1 or S_1, it can first be concluded that

$$\mathrm{T}(\mathrm{P}) \xrightarrow{R_1} \equiv \mathrm{T}(\mathrm{P}') \wedge \mathrm{P} \xrightarrow{\mathbf{receive}(m)} \mathrm{P}' \wedge \mathrm{T}(\mathrm{Q}) \xrightarrow{S_1} \equiv \mathrm{T}(\mathrm{Q}') \wedge \mathrm{Q} \xrightarrow{\mathbf{send}(\xi(ms))} \mathrm{Q}'$$

Because $\mathrm{U}(m) = [\![\mathrm{T}_\xi(ms)]\!] \stackrel{\text{Lemma 4.4}}{\Longleftrightarrow} m = \xi(ms)$ this becomes

$$\mathrm{T}(\mathrm{P}) \xrightarrow{R_1} \equiv \mathrm{T}(\mathrm{P}') \wedge \mathrm{P} \xrightarrow{\mathbf{receive}(m)} \mathrm{P}' \wedge \mathrm{T}(\mathrm{Q}) \xrightarrow{S_1} \equiv \mathrm{T}(\mathrm{Q}') \wedge \mathrm{Q} \xrightarrow{\mathbf{send}(m)} \mathrm{Q}'$$

The mCRL2 derivation in the Lemma 4.7-proof for Parallel (T2-3) shows how the first and third conjunct are sufficient to prove that $\mathrm{T}(\mathrm{P} \mathbin{\langle\langle} \mathrm{Q}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{P}' \mathbin{\langle\langle} \mathrm{Q}')$ for all $a \in \{\, \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \,\}$ for some D, R where $R \subseteq D$.
On the AWN side, the second and fourth conjunct can be used as premises in

$$\forall m \in \mathrm{MSG}\ \frac{\mathrm{P} \xrightarrow{\mathbf{receive}(m)} \mathrm{P}' \qquad \mathrm{Q} \xrightarrow{\mathbf{send}(m)} \mathrm{Q}'}{\mathrm{P} \mathbin{\langle\langle} \mathrm{Q} \xrightarrow{\tau} \mathrm{P}' \mathbin{\langle\langle} \mathrm{Q}'}\ \textsc{Parallel (T2-3)}$$

This case is proven if there exists a pair $(A_1, A_2) \in \mathscr{A}^{\sim}$ that satisfies $\mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \in A_1 \wedge \mathrm{T}(\mathrm{P} \mathbin{\langle\langle} \mathrm{Q}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}' \mathbin{\langle\langle} \mathrm{Q}') \wedge \mathrm{P} \mathbin{\langle\langle} \mathrm{Q} \xrightarrow{A_2} \mathrm{P}' \mathbin{\langle\langle} \mathrm{Q}'$, and the converse of Pair 2.3 satisfies this requirement.

2: Suppose that $\mathrm{T}(\mathrm{P})$ does a transition $a \in R_1$, $\mathrm{T}(\mathrm{Q})$ does a transition $b \in S_1$, and $\mathrm{U}(m) \neq [\![\mathrm{T}_\xi(ms)]\!]$. Following the induction hypothesis and eliminating pairs from the action relation $\mathscr{A}$ in Table 2.10 that do not match the transition labels from R_1 or S_1, it can first be concluded that

$$\mathrm{T}(\mathrm{P}) \xrightarrow{R_1} \equiv \mathrm{T}(\mathrm{P}') \wedge \mathrm{P} \xrightarrow{\mathbf{receive}(m)} \mathrm{P}' \wedge \mathrm{T}(\mathrm{Q}) \xrightarrow{S_1} \equiv \mathrm{T}(\mathrm{Q}') \wedge \mathrm{Q} \xrightarrow{\mathbf{send}(\xi(ms))} \mathrm{Q}'$$

where $\mathrm{U}(m) \neq [\![\mathrm{T}_\xi(ms)]\!] \overset{\text{Lemma 4.4}}{\Longleftrightarrow} m \neq \xi(ms)$.
The mCRL2 derivation below shows where the attempt to generate behavior for $\mathrm{T}(\mathrm{P} \mathbin{\langle\langle} \mathrm{Q})$ fails:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{receive}(\hat{\mathbb{D}}, \hat{\mathbb{D}}', \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')}\ \text{Induction hypothesis}}{\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{receive}([\![\emptyset]\!], [\![\emptyset]\!], \mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{P}')}\ \text{Choose } \hat{\mathbb{D}} = \hat{\mathbb{D}}' = \emptyset}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \xrightarrow{\{\mathbf{receive}\to\mathbf{r}\}\bullet\mathbf{receive}([\![\emptyset]\!], [\![\emptyset]\!], \mathrm{U}(m))} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}')}\ \textsc{Rename 2}}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{r}([\![\emptyset]\!], [\![\emptyset]\!], \mathrm{U}(m))} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}')}\ \text{Apply } \{\mathbf{receive}\to\mathbf{r}\}\bullet$$

$$\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{Q}) \xrightarrow{\mathbf{send}([\![\emptyset]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!])} \equiv \mathrm{T}(\mathrm{Q}')}\ \text{Induction hypothesis}}{\rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{\{\mathbf{send}\to\mathbf{s}\}\bullet\mathbf{send}([\![\emptyset]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!])} \equiv \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}')}\ \textsc{Rename 2}}{\rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{\mathbf{s}([\![\emptyset]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!])} \equiv \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}')}\ \text{Apply } \{\mathbf{send}\to\mathbf{s}\}\bullet$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \xrightarrow{\mathbf{r}([\![\emptyset]\!], [\![\emptyset]\!], \mathrm{U}(m))} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}')}\ \textit{(see above)} \qquad \dfrac{}{\rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{\mathbf{s}([\![\emptyset]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!])} \equiv \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}')}\ \textit{(see above)}}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \mid\mid \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{\mathbf{r}([\![\emptyset]\!], [\![\emptyset]\!], \mathrm{U}(m)) \mid \mathbf{s}([\![\emptyset]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!])} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \mid\mid \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}')}\ \textsc{Par 3}}{\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \mid\mid \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{\gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\mathbf{r}([\![\emptyset]\!], [\![\emptyset]\!], \mathrm{U}(m)) \mid \mathbf{s}([\![\emptyset]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!]))} \equiv \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \mid\mid \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}'))}\ \textsc{Comm 2}}{\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \mid\mid \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{\mathbf{r}([\![\emptyset]\!], [\![\emptyset]\!], \mathrm{U}(m)) \mid \mathbf{s}([\![\emptyset]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!])} \equiv \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \mid\mid \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}'))}\ \text{Apply } \gamma$$

The ALLOW 2 operator cannot be applied next (like in the previous case) because $\mathbf{r}|\mathbf{s} \notin V \cup \{\tau\}$. Since this means that $\mathrm{T}(\mathrm{M} \mathbin{\langle\langle} \mathrm{N})$ cannot do a transition in mCRL2 under the circumstances specified in this particular case, there is no behavior for AWN to mimic.

3: Suppose that $\mathrm{T}(\mathrm{P})$ does a transition a, $\mathrm{T}(\mathrm{Q})$ does a transition b, and $\underline{a} \neq \mathbf{receive}$ or $\underline{b} \neq \mathbf{send}$ or both.
This situation fails to generate behavior for $\mathrm{T}(\mathrm{P} \lll \mathrm{Q})$ because no derivation similar to the one in the Lemma 4.7-proof for Parallel (T2-3) is possible:
   a. The a action may be able to be converted to an $\mathbf{r}$ action and the b action may be able to be converted to an $\mathbf{s}$ action, but never both at the same time;
   b. The $a|b$ action produced by $\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \mid\mid \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})$ is (without arguments) never equal to $\mathbf{r}|\mathbf{s}$ (that is, $\underline{a|b} \neq \mathbf{r}|\mathbf{s}$);
   c. Because there is never an $\mathbf{r}|\mathbf{s}$ action, the communication operator $\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}$ does not generate an $\mathbf{t}$ action but leaves $a|b$ intact;
   d. Among others, the allow operator ∇_V blocks all multi-actions, and so $a|b$ is blocked.
Since $\mathrm{T}(\mathrm{P}) \lll \mathrm{T}(\mathrm{Q})$ cannot do a transition in mCRL2 under the circumstances specified in this particular case, there is no behavior for AWN to mimic.

4: Suppose that $\mathrm{T}(\mathrm{P})$ does a transition a with $\underline{a} = \mathbf{receive}$ and $\mathrm{T}(\mathrm{Q})$ does not do a transition.
This situation fails to generate behavior for $\mathrm{T}(\mathrm{P} \lll \mathrm{Q})$ because no derivation similar to the one in the Lemma 4.7-proof for Parallel (T2-3) is possible:
   a. The a action of $\mathrm{T}(\mathrm{P})$ is renamed to an $\mathbf{r}$ action by the renaming operator $\rho_{\{\mathbf{receive}\to\mathbf{r}\}}$;
   b. Because $\mathbf{r} \neq \mathbf{r}|\mathbf{s}$, the communication operator $\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}$ does not generate an $\mathbf{t}$ action but leaves the $\mathbf{r}$ action intact;
   c. The allow operator ∇_V cannot be applied because $\mathbf{r} \notin V \cup \{\tau\}$.
Since $\mathrm{T}(\mathrm{P}) \lll \mathrm{T}(\mathrm{Q})$ cannot do a transition in mCRL2 under the circumstances specified in this particular case, there is no behavior for AWN to mimic.

5: Suppose that $\mathrm{T}(\mathrm{P})$ does a transition a with $\underline{a} \neq \mathbf{receive}$ and $\mathrm{T}(\mathrm{Q})$ does not do a transition.
The induction hypothesis states that

$$\exists (A_1, A_2) \in \mathscr{A}^{\sim} \,.\, a \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}') \wedge \mathrm{P} \xrightarrow{A_2} \mathrm{P}'$$

Define $\mathscr{A}'$ as a subset of $\mathscr{A}^{\sim}$ that contains the possible values of $(A_1, A_2) \in \mathscr{A}^{\sim}$ where A_1 are actions that $\mathrm{T}(\mathrm{P} \lll \mathrm{Q})$ can perform in mCRL2 and where A_2 are the actions that AWN should use to mimic the actions in A_1 in order for this case to be proven:

$$\mathscr{A}' \stackrel{\text{def}}{=} \left\{ (A_1, A_2) \;\middle|\; (A_1, A_2) \in \mathscr{A}^{\sim}, \quad \mathrm{T}(\mathrm{P} \lll \mathrm{Q}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}' \lll \mathrm{Q}) \right\}$$

The following derivation is possible in mCRL2 for all $a \in A_1$ such that $(A_1, A_2) \in \mathscr{A}'$:

$$\underline{a} \in V \cup \{\tau\} \;\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{P}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{P}')}\ \text{Induction hypothesis}}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \xrightarrow{\{\mathbf{receive}\to\mathbf{r}\}\bullet a} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}')}\ \text{RENAME 2}}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \xrightarrow{a} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}')}\ \text{Apply } \{\mathbf{receive}\to\mathbf{r}\}\bullet}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{a} \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})}\ \text{PAR 2}}{\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{\gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(a)} \equiv \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}))}\ \text{COMM 2}}{\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{a} \equiv \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}))}\ \text{Apply } \gamma}{\nabla_V \rho_{\{\mathbf{t}\to\mathbf{t}\}}\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{a} \equiv \nabla_V \rho_{\{\mathbf{t}\to\mathbf{t}\}}\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}))}\ \text{ALLOW 2}}{\mathrm{T}(\mathrm{P} \between \mathrm{Q}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{P}' \between \mathrm{Q})}\ \text{T13}$$

The derivation above is valid only for $a \in A_1$ such that $\underline{a} \in V \cup \{\tau\}$. Under the constraints of this case, the derivation is also a representative derivation without alternatives (see Definition 4.3). Consequently, $\mathscr{A}'$ is as follows:
{

$$(\{\, \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \,\}, \{\, \tau \,\}),$$
$$(\{\, \mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathrm{D}}, [\![\mathrm{T}_\xi(ms)]\!]) \mid \hat{\mathrm{D}} \in \mathrm{T}(\mathrm{Set(IP)}) \,\}, \{\, \mathbf{broadcast}(\xi(ms)) \,\}),$$
$$(\{\, \mathbf{cast}([\![\mathrm{T}_\xi(dests)]\!], \hat{\mathrm{D}}, [\![\mathrm{T}_\xi(ms)]\!]) \mid \hat{\mathrm{D}} \in \mathrm{T}(\mathrm{Set(IP)}) \,\}, \{\, \mathbf{groupcast}(\xi(dests), \xi(ms)) \,\}),$$
$$(\{\, \mathbf{cast}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(ms)]\!]) \,\}, \{\, \mathbf{unicast}(\xi(dest), \xi(ms)) \,\}),$$
$$(\{\, \neg\mathbf{uni}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(ms)]\!]) \,\}, \{\, \neg\mathbf{unicast}(\xi(dest), \xi(ms)) \,\}),$$
$$(\{\, \mathbf{send}([\![\emptyset]\!], [\![\emptyset]\!], [\![\mathrm{T}_\xi(ms)]\!]) \,\}, \{\, \mathbf{send}(\xi(ms)) \,\}),$$
$$(\{\, \mathbf{del}(\hat{\imath}, [\![\mathrm{T}_\xi(data)]\!]) \mid \hat{\imath} \in \mathrm{T}(\mathrm{IP}) \,\}, \{\, \mathbf{deliver}(\xi(data)) \,\})$$

$\mid \quad m, \xi(ms) \in \mathrm{MSG}; \quad dest \in \mathrm{IP}; \quad \xi(data) \in \mathrm{DATA}; \quad dests, R, D \in \mathrm{Set(IP)}; \quad R \subseteq D \quad \}$

For all $(A_1, A_2) \in \mathscr{A}'$ the actions A_1 in mCRL2 can be mimicked by the actions A_2 in AWN by means of the inference rule

$$\forall a' \neq \mathbf{receive}(m)\ \dfrac{\mathrm{P} \xrightarrow{a'} \mathrm{P}'}{\mathrm{P} \between \mathrm{Q} \xrightarrow{a'} \mathrm{P}' \between \mathrm{Q}}\ \text{PARALLEL (T2-1)}$$

which is valid for all $a' \in A_2$. Therefore the induction hypothesis holds for this case.

6: Suppose that $\mathrm{T}(\mathrm{P})$ does not do a transition and $\mathrm{T}(\mathrm{Q})$ does a transition b with $\underline{b} = \mathbf{send}$. This situation fails to generate behavior for $\mathrm{T}(\mathrm{P} \between \mathrm{Q})$ because no derivation similar to the one in the Lemma 4.7-proof for Parallel (T2-3) is possible:
   a. The a action of $\mathrm{T}(\mathrm{P})$ is renamed to an $\mathbf{s}$ action by the renaming operator $\rho_{\{\mathbf{send}\to\mathbf{s}\}}$;
   b. Because $\mathbf{s} \neq \mathbf{r}|\mathbf{s}$, the communication operator $\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}$ does not generate an $\mathbf{t}$ action but leaves the $\mathbf{s}$ action intact;
   c. The allow operator ∇_V cannot be applied because $\mathbf{s} \notin V \cup \{\tau\}$.

   Since $\mathrm{T}(\mathrm{P}) \between \mathrm{T}(\mathrm{Q})$ cannot do a transition in mCRL2 under the circumstances specified in this particular case, there is no behavior for AWN to mimic.

7: Suppose that $\mathrm{T}(\mathrm{P})$ does not do a transition and $\mathrm{T}(\mathrm{Q})$ does a transition b with $\underline{b} \neq \mathbf{send}$. The induction hypothesis states that

$$\exists (B_1, B_2) \in \mathscr{A}^{\sim}\,.\, b \in B_1 \wedge \mathrm{T}(\mathrm{Q}) \xrightarrow{B_1} \equiv \mathrm{T}(\mathrm{Q}') \wedge \mathrm{Q} \xrightarrow{B_2} \mathrm{Q}'$$

Define $\mathscr{A}'$ as a subset of $\mathscr{A}^{\sim}$ that contains the possible values of $(B_1, B_2) \in \mathscr{A}^{\sim}$ where B_1 are actions that $\mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q})$ can perform in mCRL2 and where B_2 are the actions that AWN should use to mimic the actions in B_1 in order for this case to be proven:

$$\mathscr{A}' \stackrel{\text{def}}{=} \left\{ (B_1, B_2) \;\middle|\; (B_1, B_2) \in \mathscr{A}^{\sim}, \quad \mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q}) \xrightarrow{B_1} \mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q}') \right\}$$

The following derivation is possible in mCRL2 for all $b \in B_1$ such that $(B_1, B_2) \in \mathscr{A}'$:

$$\underline{b} \in V \cup \{\tau\} \;\; \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{}\text{ Induction hypothesis}}{\mathrm{T}(\mathrm{Q}) \xrightarrow{b} \equiv \mathrm{T}(\mathrm{Q}')}\text{ Rename 2}}{\rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{\{\mathbf{send}\to\mathbf{s}\}\bullet b} \equiv \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}')}\text{ Apply } \{\mathbf{send}\to\mathbf{s}\}\bullet}{\rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{b} \equiv \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}')}\text{ Par 5}}{\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}) \xrightarrow{b} \equiv \rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})}\text{ Comm 2}}{\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{\gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(b)} \equiv \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}))}\text{ Apply } \gamma}{\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{b} \equiv \Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}))}\text{ Allow 2}}{\nabla_V \rho_{\{\mathbf{t}\to\mathbf{t}\}}\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}) \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q})) \xrightarrow{b} \equiv \nabla_V \rho_{\{\mathbf{t}\to\mathbf{t}\}}\Gamma_{\{\mathbf{r}|\mathbf{s}\to\mathbf{t}\}}(\rho_{\{\mathbf{receive}\to\mathbf{r}\}}\mathrm{T}(\mathrm{P}') \,||\, \rho_{\{\mathbf{send}\to\mathbf{s}\}}\mathrm{T}(\mathrm{Q}))}\text{ T13}}{\mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q}) \xrightarrow{b} \equiv \mathrm{T}(\mathrm{P}' \langle\langle \mathrm{Q})}$$

The derivation above is valid only for $b \in B_1$ such that $\underline{b} \in V \cup \{\tau\}$. Under the constraints of this case, the derivation is also a representative derivation without alternatives (see Definition 4.3). Consequently, $\mathscr{A}'$ is as follows:
{

$$\begin{gathered}
(\{\, \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \,\}, \{\, \tau \,\}), \\
(\{\, \mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathrm{D}}, [\![\mathrm{T}_\xi(ms)]\!]) \mid \hat{\mathrm{D}} \in \mathrm{T}(\mathrm{Set(IP)}) \,\}, \{\, \mathbf{broadcast}(\xi(ms)) \,\}), \\
(\{\, \mathbf{cast}([\![\mathrm{T}_\xi(dests)]\!], \hat{\mathrm{D}}, [\![\mathrm{T}_\xi(ms)]\!]) \mid \hat{\mathrm{D}} \in \mathrm{T}(\mathrm{Set(IP)}) \,\}, \{\, \mathbf{groupcast}(\xi(dests), \xi(ms)) \,\}), \\
(\{\, \mathbf{cast}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(ms)]\!]) \,\}, \{\, \mathbf{unicast}(\xi(dest), \xi(ms)) \,\}), \\
(\{\, \neg\mathbf{uni}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(ms)]\!]) \,\}, \{\, \neg\mathbf{unicast}(\xi(dest), \xi(ms)) \,\}), \\
(\{\, \mathbf{receive}(\hat{\mathrm{D}}, \hat{\mathrm{D}}\text{'}, \mathrm{U}(m)) \mid \hat{\mathrm{D}}, \hat{\mathrm{D}}\text{'} \in \mathrm{T}(\mathrm{Set(IP)}) \,\}, \{\, \mathbf{receive}(m) \,\}), \\
(\{\, \mathbf{del}(\hat{\mathrm{i}}, [\![\mathrm{T}_\xi(data)]\!]) \mid \hat{\mathrm{i}} \in \mathrm{T}(\mathrm{IP}) \,\}, \{\, \mathbf{deliver}(\xi(data)) \,\})
\end{gathered}$$

$\mid \quad m, \xi(ms) \in \mathrm{MSG}; dest \in \mathrm{IP}; \quad \xi(data) \in \mathrm{DATA}; \quad dests, R, D \in \mathrm{Set(IP)}; \quad R \subseteq D \quad \}$

For all $(B_1, B_2) \in \mathscr{A}'$ the actions B_1 in mCRL2 can be mimicked by the actions B_2 in AWN by means of the inference rule

$$\forall b' \neq \mathbf{send}(m) \;\; \frac{\mathrm{Q} \xrightarrow{b'} \mathrm{Q}'}{\mathrm{P} \langle\langle \mathrm{Q} \xrightarrow{b'} \mathrm{P} \langle\langle \mathrm{Q}'} \text{ Parallel (T2-2)}$$

which is valid for all $b' \in B_2$. Therefore the induction hypothesis holds for this case.
The derivations in mCRL2 from the cases listed above are representative derivations that collectively have no alternatives (see Definition 4.3): there are no other derivations with different or differently ordered one-way steps that yield different behavior of process expression $\mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q})$. Therefore the induction hypothesis generally holds for this expression.

Translation rule T14: The translation function T is partially defined by translation rule

$$\begin{aligned}
&\mathrm{T}(ip:\mathrm{P}:R) = \nabla_V \Gamma_C(\mathrm{T}(\mathrm{P}) \mid\mid \mathrm{G}(t_{\mathrm{U}(ip)}, t_{\mathrm{U}(R)})) \quad \text{T14}\\
&\quad \text{where } V = \{\mathbf{t}, \mathbf{starcast}, \mathbf{arrive}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}\\
&\quad \text{where } C = \{\mathbf{cast}|\overline{\mathbf{cast}} \to \mathbf{starcast}, \neg\mathbf{uni}|\overline{\neg\mathbf{uni}} \to \mathbf{t}, \mathbf{del}|\overline{\mathbf{del}} \to \mathbf{deliver}, \mathbf{receive}|\overline{\mathbf{receive}} \to \mathbf{arrive}\}\\
&\text{where } \mathrm{G}(\mathtt{ip},\mathtt{R}) = \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set(IP)}),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (\mathtt{R}\cap\mathtt{D} = \mathtt{D'}) \to \overline{\mathbf{cast}}(\mathtt{D},\mathtt{D'},\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R})\\
&\qquad + \textstyle\sum_{\mathtt{d}:\mathrm{T}(\mathrm{IP}),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (\mathtt{d} \notin \mathtt{R}) \to \overline{\neg\mathbf{uni}}(\{\mathtt{d}\},\emptyset,\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R})\\
&\qquad + \textstyle\sum_{\mathtt{data}:\mathrm{DATA}} \overline{\mathbf{del}}(\mathtt{ip},\mathtt{data}).\mathrm{G}(\mathtt{ip},\mathtt{R})\\
&\qquad + \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set(IP)}),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (\mathtt{ip} \in \mathtt{D'}) \to \overline{\mathbf{receive}}(\mathtt{D},\mathtt{D'},\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R})\\
&\qquad + \textstyle\sum_{\mathtt{D},\mathtt{D'}:\mathrm{T}(\mathrm{Set(IP)}),\mathtt{msg}:\mathrm{T}(\mathrm{MSG})} (\mathtt{ip} \notin \mathtt{D'}) \to \mathbf{arrive}(\mathtt{D},\mathtt{D'},\mathtt{msg}).\mathrm{G}(\mathtt{ip},\mathtt{R})\\
&\qquad + \textstyle\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{connect}(\mathtt{ip},\mathtt{ip'}).\mathrm{G}(\mathtt{ip},\mathtt{R}\cup\{\mathtt{ip'}\})\\
&\qquad + \textstyle\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{connect}(\mathtt{ip'},\mathtt{ip}).\mathrm{G}(\mathtt{ip},\mathtt{R}\cup\{\mathtt{ip'}\})\\
&\qquad + \textstyle\sum_{\mathtt{ip'},\mathtt{ip''}:\mathrm{T}(\mathrm{IP})} (\mathtt{ip} \notin \{\mathtt{ip'},\mathtt{ip''}\}) \to \mathbf{connect}(\mathtt{ip'},\mathtt{ip''}).\mathrm{G}(\mathtt{ip},\mathtt{R})\\
&\qquad + \textstyle\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{disconnect}(\mathtt{ip},\mathtt{ip'}).\mathrm{G}(\mathtt{ip},\mathtt{R}\setminus\{\mathtt{ip'}\})\\
&\qquad + \textstyle\sum_{\mathtt{ip'}:\mathrm{T}(\mathrm{IP})} \mathbf{disconnect}(\mathtt{ip'},\mathtt{ip}).\mathrm{G}(\mathtt{ip},\mathtt{R}\setminus\{\mathtt{ip'}\})\\
&\qquad + \textstyle\sum_{\mathtt{ip'},\mathtt{ip''}:\mathrm{T}(\mathrm{IP})} (\mathtt{ip} \notin \{\mathtt{ip'},\mathtt{ip''}\}) \to \mathbf{disconnect}(\mathtt{ip'},\mathtt{ip''}).\mathrm{G}(\mathtt{ip},\mathtt{R})
\end{aligned}$$

Process expressions $\mathrm{T}(ip:\mathrm{P}:R)$ produced by this translation rule have by design a ∇_V operator on the outside. As a consequence of mCRL2 inference rule ALLOW 2, if $\mathrm{T}(ip:\mathrm{P}:R) \xrightarrow{a} \equiv \mathrm{Q}$ then $a \in V \cup \{\tau\} = \{\tau, \mathbf{t}, \mathbf{starcast}, \mathbf{arrive}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}$. The induction hypothesis must be proven for each of these actions:

1: Proof for τ actions.
There is no scenario in which τ actions can be produced by a $\mathrm{T}(ip:\mathrm{P}:R)$ expression, and therefore the induction hypothesis is established automatically.

2: Proof for $\mathbf{t}$ actions.
There are exactly two possible sources for $\mathbf{t}$ actions in T14: synchronizing a $\neg\mathbf{uni}$ action and $\overline{\neg\mathbf{uni}}$ action into a $\mathbf{t}$ action or leaving a $\mathbf{t}$ action performed by $\mathrm{T}(\mathrm{P})$ unchanged and unblocked. Pair 2.3 is the only pair in $\mathscr{A}$ that matches the $\mathbf{t}$, and therefore it must be proven that AWN can always mimic a $\mathbf{t}$ with a τ.
A case distinction is made based on the source:

- The first source corresponds with the derivation found in the Lemma 4.7-proof for Unicast (T3-2), which is a representative derivation without alternatives (see Definition 4.3): there are no other derivations that produce a conclusion of the form $\mathrm{T}(ip:\mathrm{P}:R)$ in which a a $\neg\mathbf{uni}$ action and $\overline{\neg\mathbf{uni}}$ action are synchronized into a $\mathbf{t}$ action.
  The induction hypothesis states that under these circumstances there must be some $(A_1,A_2) \in \mathscr{A}^{\sim}.\ \mathbf{t} \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}')$ such that $\mathrm{P} \xrightarrow{a'} \mathrm{P}'$ for all $a' \in A_2$. Clearly,

$$(A_1,A_2) = (\{\ \neg\mathbf{uni}([\![\mathrm{T}_\xi(dest)]\!], [\![\mathrm{T}_\xi(ms)]\!])\ \}, \{\ \neg\mathbf{unicast}(\xi(dest), \xi(ms))\ \})$$

and therefore $\mathrm{P} \xrightarrow{\neg\mathbf{unicast}(dip,m)} \mathrm{P}'$. Additionally, from the assumption that $[\![t_{\mathrm{U}(dip)} \notin t_{\mathrm{U}(R)}]\!] = \mathit{true}$ in the derivation follows that $dip \notin R$. It then becomes clear that inference rule UNICAST (T3-2) can be used to reach the conclusion that $ip : \mathrm{P} : R \xrightarrow{\tau} ip : \mathrm{P}' : R$.

- The second source for **t** corresponds with the derivation found in the Lemma 4.7-proof for Internal (T3), which is a representative derivation without alternatives (see Definition 4.3): there are no other derivations that produce a conclusion of the form $\mathrm{T}(ip : \mathrm{P} : R)$ in which a **t** action performed by $\mathrm{T}(\mathrm{P})$ is left unchanged and unblocked.
  The induction hypothesis states that under these circumstances there must be some $(A_1, A_2) \in \mathscr{A}^{\sim}$. $\mathbf{t} \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}')$ such that $\mathrm{P} \xrightarrow{a'} \mathrm{P}'$ for all $a' \in A_2$. Clearly,

  $$(\{\, \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \,\}, \{\, \tau \,\})$$

  and therefore $\mathrm{P} \xrightarrow{\tau} \mathrm{P}'$. Inference rule INTERNAL (T3) can then be used to reach the conclusion that $ip : \mathrm{P} : R \xrightarrow{\tau} ip : \mathrm{P}' : R$.

Having covered all possibilities for $\mathrm{T}(ip : \mathrm{P} : R)$ to do a **t** action, Lemma 4.9 has been proven for this particular case.

3: Proof for **starcast** actions.
In order for $\mathrm{T}(ip : \mathrm{P} : R)$ to produce a **starcast** action, $\mathrm{T}(\mathrm{P})$ must produce a **cast** action that synchronizes with a $\overline{\mathbf{cast}}$ action of G. The induction hypothesis states that – if that is the case – there must be some $(A_1, A_2) \in \mathscr{A}^{\sim}$. $\mathbf{cast}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}')$ such that $\mathrm{P} \xrightarrow{a'} \mathrm{P}'$ for all $a' \in A_2$. These are the candidates for (A_1, A_2):

$$(\{\, \mathbf{cast}([\![\mathscr{U}_{\mathrm{IP}}]\!], \hat{\mathrm{D}}, [\![\mathrm{T}_\xi(ms)]\!]) \mid \hat{\mathrm{D}} \in \mathrm{T}(\mathrm{Set}(\mathrm{IP})) \,\}, \{\, \mathbf{broadcast}(\xi(ms)) \,\})$$
$$(\{\, \mathbf{cast}([\![\mathrm{T}_\xi(dests)]\!], \hat{\mathrm{D}}, [\![\mathrm{T}_\xi(ms)]\!]) \mid \hat{\mathrm{D}} \in \mathrm{T}(\mathrm{Set}(\mathrm{IP})) \,\}, \{\, \mathbf{groupcast}(\xi(dests), \xi(ms)) \,\})$$
$$(\{\, \mathbf{cast}([\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(\{dest\})]\!], [\![\mathrm{T}_\xi(ms)]\!]) \,\}, \{\, \mathbf{unicast}(\xi(dest), \xi(ms)) \,\})$$

In other words, one of the following premises must hold:

$$\mathrm{P} \xrightarrow{\mathbf{broadcast}(\xi(ms))} \mathrm{P}' \text{ or } \mathrm{P} \xrightarrow{\mathbf{groupcast}(\xi(dests), \xi(ms))} \mathrm{P}' \text{ or } \mathrm{P} \xrightarrow{\mathbf{unicast}(\xi(dest), \xi(ms))} \mathrm{P}'$$

A case distinction is made for these premises:

- Suppose that $\mathrm{P} \xrightarrow{\mathbf{broadcast}(\xi(ms))} \mathrm{P}'$. This leads to a derivation that can be found in the Lemma 4.7-proof for Broadcast (T3). Since no other derivations that are based on the premise *and* that lead to a conclusion of the form $\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{Q}$ are possible (the derivation is a representative derivation without alternatives, see Definition 4.3), AWN must be able to do the transition

  $$ip : \mathrm{P} : R \xrightarrow{R:\mathbf{*cast}(\xi(ms))} ip : \mathrm{P}' : R$$

for some *ip* in order to mimic mCRL2. Indeed, applying inference rule BROADCAST (T3) produces the required conclusion.

- Suppose that $\mathrm{P} \xrightarrow{\mathbf{groupcast}(\xi(dests),\xi(ms))} \mathrm{P}'$. This leads to a derivation that can be found in the Lemma 4.7-proof for Groupcast (T3). Since no other derivations that are based on the premise *and* that lead to a conclusion of the form $\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{Q}$ are possible (the derivation is a representative derivation without alternatives, see Definition 4.3), AWN must be able to do the transition

  $$ip : \mathrm{P} : R \xrightarrow{R \cap D:\mathbf{*cast}(\xi(ms))} ip : \mathrm{P}' : R$$

  for some *ip* in order to mimic mCRL2. Indeed, applying inference rule GROUPCAST (T3) produces the required conclusion.
- Suppose that $\mathrm{P} \xrightarrow{\mathbf{unicast}(\xi(dest),\xi(ms))} \mathrm{P}'$. This leads to a derivation that can be found in the Lemma 4.7-proof for Unicast (T3-1). Since no other derivations that are based on the premise *and* that lead to a conclusion of the form $\mathrm{T}(ip : \mathrm{P} : R) \xrightarrow{a} \equiv \mathrm{Q}$ are possible (the derivation is a representative derivation without alternatives, see Definition 4.3), AWN must be able to do the transition

  $$ip : \mathrm{P} : R \xrightarrow{\{dip\}:\mathbf{*cast}(\xi(ms))} ip : \mathrm{P}' : R$$

  for some *ip* in order to mimic mCRL2. Indeed, applying inference rule UNICAST (T3-1) produces the required conclusion, but only under the condition that $dip \in R$. This follows from $[\![t_{\mathrm{U}(R)} \cap t_{\mathrm{U}(\{dip\})} = t_{\mathrm{U}(\{dip\})}]\!] = true$, an assumption that is necessary for the derivation in the Lemma 4.7-proof for Unicast (T3-1).

Having covered all possibilities for $\mathrm{T}(ip : \mathrm{P} : R)$ to do a **starcast** action, Lemma 4.9 has been proven for this particular case.

4: Proof for **arrive** actions.
There are exactly two possible sources for **arrive** actions in T14: letting G generate an **arrive** action on its own or converting a **receive** action performed by $\mathrm{T}(\mathrm{P})$ through synchronization with G.
A case distinction is made based on the source:

- The first source corresponds with the derivation found in the Lemma 4.7-proof for Arrive (T3-2), which is a representative derivation without alternatives (see Definition 4.3): there are no other derivations that produce a conclusion of the form $\mathrm{T}(ip : \mathrm{P} : R)$ in which G generates an **arrive** action on its own.
  AWN can copy the behavior via inference rule ARRIVE (T3-2); Lemma 4.9 applies because

  $$\left(\left\{ \mathbf{arrive}(\hat{\mathrm{D}}, \hat{\mathrm{D}}', \mathrm{U}(m)) \;\middle|\; \begin{array}{c} \hat{\mathrm{D}},\hat{\mathrm{D}}' \in \mathrm{T}(\mathrm{Set}(\mathrm{IP})) \\ \hat{\mathrm{D}}' \subseteq \hat{\mathrm{D}} \\ \mathrm{U}(H) \subseteq \hat{\mathrm{D}}' \\ \mathrm{U}(K) \cap \hat{\mathrm{D}}' = \emptyset \end{array} \right\}, \{\, H \neg K : \mathbf{arrive}(m) \,\}\right)$$

can be chosen as $(A_1,A_2) \in \mathscr{A}^{\sim}$. $\mathbf{arrive}(\hat{\mathtt{D}},\hat{\mathtt{D}}\mathtt{'},\mathrm{U}(m)) \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}')$ such that $\mathrm{P} \xrightarrow{a'} \mathrm{P}'$ for all $a' \in A_2$.

- The second source for **arrive** corresponds with the derivation found in the Lemma 4.7-proof for Arrive (T3-1), which is a representative derivation without alternatives (see Definition 4.3): there are no other derivations that produce a conclusion of the form $\mathrm{T}(ip : \mathrm{P} : R)$ in which a **receive** action is converted through synchronization with G.
  In this particular case, the induction hypothesis states that there must be some $(A_1,A_2) \in \mathscr{A}^{\sim}$. $\mathbf{receive}([\![\hat{\mathtt{D}}]\!],[\![\hat{\mathtt{D}}\mathtt{'}]\!],\mathrm{U}(m)) \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}')$ such that $\mathrm{P} \xrightarrow{a'} \mathrm{P}'$ for all $a' \in A_2$. Clearly,

$$(\{\ \mathbf{receive}(\hat{\mathtt{D}},\hat{\mathtt{D}}\mathtt{'},\mathrm{U}(m)) \mid \hat{\mathtt{D}},\hat{\mathtt{D}}\mathtt{'} \in \mathrm{T}(\mathrm{Set}(\mathrm{IP}))\ \},\{\ \mathbf{receive}(m)\ \})$$

  and therefore $\mathrm{P} \xrightarrow{\mathbf{receive}(m)} \mathrm{P}'$. Inference rule ARRIVE (T3-1) can then be used to reach the conclusion that $ip : \mathrm{P} : R \xrightarrow{\{ip\}\neg\emptyset:\mathbf{arrive}(m)} ip : \mathrm{P}' : R$.

Having covered all possibilities for $\mathrm{T}(ip : \mathrm{P} : R)$ to do an **arrive** action, Lemma 4.9 has been proven for this particular case.

5: Proof for **deliver** actions.
In order for $\mathrm{T}(ip : \mathrm{P} : R)$ to produce a **deliver** action, $\mathrm{T}(\mathrm{P})$ must produce a **del** action that synchronizes with a $\overline{\mathbf{del}}$ action of G. This corresponds with the derivation found in the Lemma 4.7-proof for Deliver (T3), which is a representative derivation without alternatives (see Definition 4.3): there are no other derivations that produce a conclusion of the form $\mathrm{T}(ip : \mathrm{P} : R)$ in which **del** and $\overline{\mathbf{del}}$ are synchronized.
The induction hypothesis states that under these circumstances there must be some $(A_1,A_2) \in \mathscr{A}^{\sim}$. $\mathbf{del}(\mathrm{U}(ip),\mathrm{U}(d)) \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}')$ such that $\mathrm{P} \xrightarrow{a'} \mathrm{P}'$ for all $a' \in A_2$. There is only one candidate for (A_1,A_2), namely

$$(\{\ \mathbf{del}(\hat{\mathtt{i}},[\![\mathrm{T}_{\xi}(data)]\!]) \mid \hat{\mathtt{i}} \in \mathrm{T}(\mathrm{IP})\ \},\{\ \mathbf{deliver}(\xi(data))\ \})$$

It follows that $\mathrm{P} \xrightarrow{\mathbf{deliver}(\xi(data))} \mathrm{P}'$, which can serve as the premise for inference rule DELIVER (T3) to prove that $ip : \mathrm{P} : R \xrightarrow{ip:\mathbf{deliver}(\xi(data))} ip : \mathrm{P}' : R$. This covers all possibilities for $\mathrm{T}(ip : \mathrm{P} : R)$ to do a **deliver** action, thus proving Lemma 4.9 for this particular case.

6: Proof for **connect** actions.
Process expressions $\mathrm{T}(ip : \mathrm{P} : R)$ can only produce **connect** actions if they are generated by G such as in the derivation found in the Lemma 4.7-proof for Connect (T3-1). This derivation as well as the comparable derivations from the proofs for CONNECT (T3-2) and CONNECT (T3-3) are representative derivations without alternatives (see Definition 4.3): there are no other derivations that produce a conclusion of the form $\mathrm{T}(ip : \mathrm{P} : R)$ in which G generates a **connect** action.

AWN can copy the behavior of mCRL2 via inference rules CONNECT (T3-1), CONNECT (T3-2), and CONNECT (T3-3); Lemma 4.9 applies because

$$(\{\ \textbf{connect}(\mathrm{U}(ip'),\mathrm{U}(ip''))\ \},\{\ \textbf{connect}(ip',ip'')\ \})$$

can be chosen as $(A_1,A_2) \in \mathscr{A}^{\sim}$. $\textbf{connect}(\mathrm{U}(ip),\mathrm{U}(ip')) \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}')$ such that $\mathrm{P} \xrightarrow{a'} \mathrm{P}'$ for all $a' \in A_2$.
This covers all possibilities for $\mathrm{T}(ip : \mathrm{P} : R)$ to do a **connect** action, thus proving Lemma 4.9 for this particular case.

7: Proof for **disconnect** actions.
This proof is analogous to the proof for **connect** actions.

Translation rule T15: The translation function T is partially defined by translation rule

$$\begin{aligned}
&\mathrm{T}(\mathrm{M} \,||\, \mathrm{N}) = \rho_R \nabla_V \Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}} \Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{T}(\mathrm{N})) \quad \text{T15}\\
&\quad \text{where } R = \{\mathbf{a}\to\mathbf{arrive}, \mathbf{c}\to\mathbf{connect}, \mathbf{d}\to\mathbf{disconnect}, \mathbf{s}\to\mathbf{starcast}\}\\
&\quad \text{where } V = \{\mathbf{a},\mathbf{c},\mathbf{d},\mathbf{deliver},\mathbf{s},\mathbf{t}\}\\
&\quad \text{where } C = \{\mathbf{starcast}|\mathbf{arrive}\to\mathbf{s}, \mathbf{connect}|\mathbf{connect}\to\mathbf{c}, \mathbf{disconnect}|\mathbf{disconnect}\to\mathbf{d}\}
\end{aligned}$$

Consider several pairs from Table 2.10:

$$\begin{aligned}
&(\{\,\tau\,\}, \{\,\mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m))\,\})\\
&(\{\,R : \mathbf{*cast}(m)\,\}, \{\,\mathbf{starcast}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m))\,\})\\
&(\{\,ip : \mathbf{deliver}(d)\,\}, \{\,\mathbf{deliver}(\mathrm{U}(ip), \mathrm{U}(d))\,\})\\
&(\{\,H\neg K : \mathbf{arrive}(m)\,\}, \left\{ \mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}', \mathrm{U}(m)) \;\middle|\; \begin{smallmatrix} \hat{\mathtt{D}},\hat{\mathtt{D}}'\in\mathrm{T}(\mathrm{Set(IP)}) \\ \hat{\mathtt{D}}'\subseteq\hat{\mathtt{D}} \\ \mathrm{U}(H)\subseteq\hat{\mathtt{D}}' \\ \mathrm{U}(K)\cap\hat{\mathtt{D}}'=\emptyset \end{smallmatrix} \right\})\\
&(\{\,\mathbf{connect}(ip', ip'')\,\}, \{\,\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))\,\})\\
&(\{\,\mathbf{disconnect}(ip', ip'')\,\}, \{\,\mathbf{disconnect}(\mathrm{U}(ip'), \mathrm{U}(ip''))\,\})
\end{aligned}$$

Let T_1, S_1, L_1, A_1, C_1, and D_1 be defined as the second members of these pairs; that is, let

$$\begin{aligned}
T_1 &\stackrel{\text{def}}{=} \{\,\mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m))\,\})\\
S_1 &\stackrel{\text{def}}{=} \{\,\mathbf{starcast}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m))\,\}\\
L_1 &\stackrel{\text{def}}{=} \{\,\mathbf{deliver}(\mathrm{U}(ip), \mathrm{U}(d))\,\}\\
A_1 &\stackrel{\text{def}}{=} \left\{ \mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}', \mathrm{U}(m)) \;\middle|\; \begin{smallmatrix} \hat{\mathtt{D}},\hat{\mathtt{D}}'\in\mathrm{T}(\mathrm{Set(IP)}) \\ \hat{\mathtt{D}}'\subseteq\hat{\mathtt{D}} \\ \mathrm{U}(H)\subseteq\hat{\mathtt{D}}' \\ \mathrm{U}(K)\cap\hat{\mathtt{D}}'=\emptyset \end{smallmatrix} \right\})\\
C_1 &\stackrel{\text{def}}{=} \{\,\mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip''))\,\}\\
D_1 &\stackrel{\text{def}}{=} \{\,\mathbf{disconnect}(\mathrm{U}(ip'), \mathrm{U}(ip''))\,\}
\end{aligned}$$

for some $d \in \mathrm{DATA}$, $ip, ip', ip'' \in \mathrm{IP}$, $D, R, H, K \in \mathrm{Set(IP)}$, and $m \in \mathrm{MSG}$.
The following cases are distinguished:

1: $\mathrm{T}(\mathrm{M})$ or $\mathrm{T}(\mathrm{N})$ does a transition $a \in T_1$; the other process does not do a transition.
2: $\mathrm{T}(\mathrm{M})$ or $\mathrm{T}(\mathrm{N})$ does a transition $a \in L_1$; the other process does not do a transition.
3: $\mathrm{T}(\mathrm{M})$ or $\mathrm{T}(\mathrm{N})$ does a transition a where $\underline{a} \notin \{\mathbf{t}, \mathbf{deliver}\}$; the other process does not do a transition.
4: $\mathrm{T}(\mathrm{M})$ and $\mathrm{T}(\mathrm{N})$ both do the same transition $a \in A_1$.
5: $\mathrm{T}(\mathrm{M})$ and $\mathrm{T}(\mathrm{N})$ both do the same transition $a \in C_1$.
6: $\mathrm{T}(\mathrm{M})$ and $\mathrm{T}(\mathrm{N})$ both do the same transition $a \in D_1$.

7: $\mathrm{T}(\mathrm{M})$ and $\mathrm{T}(\mathrm{N})$ both do the same transition a where a must satisfy the requirement that $\underline{a} \notin \{\mathbf{arrive}, \mathbf{connect}, \mathbf{disconnect}\}$.
8: $\mathrm{T}(\mathrm{M})$ does a transition $a \in S_1$ and $\mathrm{T}(\mathrm{N})$ does a transition $b \in A_1$, or vice versa.
9: $\mathrm{T}(\mathrm{M})$ does a transition $a \in S_1$ and $\mathrm{T}(\mathrm{N})$ does a transition $b \notin A_1$ where $\underline{b} = \mathbf{arrive}$, or vice versa.
10: $\mathrm{T}(\mathrm{M})$ does a transition a and $\mathrm{T}(\mathrm{N})$ does a transition $b \neq a$ where a and b must satisfy the requirement that $\{\underline{a}, \underline{b}\} \neq \{\mathbf{starcast}, \mathbf{arrive}\}$.

Note that these cases cover all combinations of behavior of $\mathrm{T}(\mathrm{M})$ and $\mathrm{T}(\mathrm{N})$.
The proof is provided below for each of the cases:

1: Suppose that $\mathrm{T}(\mathrm{M})$ or $\mathrm{T}(\mathrm{N})$ does a transition $a \in T_1$; the other process does not do a transition. Following the induction hypothesis and eliminating pairs from the action relation $\mathscr{A}$ in Table 2.10 that do not match the transition labels from T_1, it can first be concluded that

$$\mathrm{T}(\mathrm{M}) \xrightarrow{T_1} \equiv \mathrm{T}(\mathrm{M}') \wedge \mathrm{M} \xrightarrow{\tau} \mathrm{M}'$$

The mCRL2 derivation below shows how the first conjunct is sufficient to prove that $\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{T_1} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N})$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{M}')}\ \text{Induction hypothesis}}{\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N}) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N})}\ \textsc{Par 2}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_C(\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \textsc{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \text{Apply } \gamma_C}{\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}(\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m)))} \equiv \Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \textsc{Comm 2}}{\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \text{Apply } \gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}}{\tau \in V \cup \{\tau\}\quad \nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \textsc{Allow 2}}{\rho_R\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{R\bullet\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \rho_R\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \textsc{Rename 2}}{\rho_R\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \rho_R\nabla_V\Gamma_{\{\mathbf{arrive}\mid\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}))}\ \text{Apply } R\bullet}{\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N})}\ \text{T15}$$

On the AWN side, the second conjunct can be used as premise in

$$\frac{\mathrm{M} \xrightarrow{\tau} \mathrm{M}'}{\mathrm{M} \mid\mid \mathrm{N} \xrightarrow{\tau} \mathrm{M}' \mid\mid \mathrm{N}}\ \textsc{Internal (T4-1)}$$

This case is proven if there exists a pair $(A_1, A_2) \in \mathscr{A}^{\sim}$ that satisfies $\mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \in A_1 \wedge \mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N}) \wedge \mathrm{M} \mid\mid \mathrm{N} \xrightarrow{A_2} \mathrm{M}' \mid\mid \mathrm{N}$, and the converse of Pair 2.3 satisfies this requirement.
The proof for when N does the $\mathbf{t}$ action is similar.

2: Suppose that $\mathrm{T}(\mathrm{M})$ or $\mathrm{T}(\mathrm{N})$ does a transition $a \in L_1$; the other process does not do a transition. Following the induction hypothesis and eliminating pairs from the action

relation $\mathscr{A}$ in Table 2.10 that do not match the transition labels from L_1, it can first be concluded that

$$\mathrm{T}(\mathrm{M}) \xrightarrow{L_1} \equiv \mathrm{T}(\mathrm{M}') \wedge \mathrm{M} \xrightarrow{ip:\mathbf{deliver}(d)} \mathrm{M}'$$

The mCRL2 derivation in the Lemma 4.7-proof for Deliver (T4-1) shows how the first conjunct is sufficient to prove that $\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{L_1} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N})$.
On the AWN side, the second conjunct can be used as premise in

$$\frac{\mathrm{M} \xrightarrow{ip:\mathbf{deliver}(d)} \mathrm{M}'}{\mathrm{M} \mid\mid \mathrm{N} \xrightarrow{ip:\mathbf{deliver}(d)} \mathrm{M}' \mid\mid \mathrm{N}}\ \textsc{Deliver (T4-1)}$$

This case is proven if there exists a pair $(A_1, A_2) \in \mathscr{A}^{\sim}$ that satisfies $a \in A_1 \wedge \mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N}) \wedge \mathrm{M} \mid\mid \mathrm{N} \xrightarrow{A_2} \mathrm{M}' \mid\mid \mathrm{N}$, and the converse of Pair 2.12 satisfies this requirement.
The proof for when N does the **deliver** action is similar.

3: Suppose that $\mathrm{T}(\mathrm{M})$ or $\mathrm{T}(\mathrm{N})$ does a transition a where $\underline{a} \notin \{\mathbf{t}, \mathbf{deliver}\}$; the other process does not do a transition.
This situation fails to generate behavior for $\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N})$ because no derivation similar to the one in the Lemma 4.7-proof for Deliver (T4-1) is possible:

a. The condition $\underline{a} \notin \{\mathbf{t}, \mathbf{deliver}\}$ implies that

$$\underline{a} \in \{\mathbf{cast}, \neg\mathbf{uni}, \mathbf{receive}, \mathbf{send}, \mathbf{del}, \mathbf{starcast}, \mathbf{arrive}, \mathbf{connect}, \mathbf{disconnect}, \mathbf{newpkt}\}$$

because these are all action labels from the right-hand side of the action relation $\mathscr{A}$;

b. $\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})$ does not produce multi-actions because only one of $\mathrm{T}(\mathrm{M})$ and $\mathrm{T}(\mathrm{N})$ does a transition;

c. The communication operator Γ_C has no effect because $\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})$ does not produce multi-actions, and therefore no **c**, **d**, or **s** actions are generated;

d. The communication operator $\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}$ has no effect because $\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N}))$ does not produce multi-actions, and therefore no **a** action is generated;

e. The allow operator ∇_V blocks all actions except **a**, **c**, **d**, **deliver**, **s**, or **t**, none of which can be produced by $\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N}))$.

Since $\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})$ cannot do a transition in mCRL2 under the circumstances specified in this particular case, there is no behavior for AWN to mimic.

4: Suppose that $\mathrm{T}(\mathrm{M})$ and $\mathrm{T}(\mathrm{N})$ both do the same transition $a \in A_1$. Following the induction hypothesis and eliminating pairs from the action relation $\mathscr{A}$ in Table 2.10 that do not match the transition labels from A_1, it can first be concluded that

$$\mathrm{T}(\mathrm{M}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{M}') \wedge \mathrm{M} \xrightarrow{H\neg K:\mathbf{arrive}(m)} \mathrm{M}' \wedge \mathrm{T}(\mathrm{N}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{N}') \wedge \mathrm{N} \xrightarrow{H'\neg K':\mathbf{arrive}(m)} \mathrm{N}'$$

The mCRL2 derivation in the Lemma 4.7-proof for Cast (T4-3) shows how the first and third conjunct are sufficient to prove that $\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N})$.
On the AWN side, the second and fourth conjunct can be used as premise in

$$\frac{\mathrm{M} \xrightarrow{H\neg K:\mathbf{arrive}(m)} \mathrm{M}' \qquad \mathrm{N} \xrightarrow{H'\neg K':\mathbf{arrive}(m)} \mathrm{N}'}{\mathrm{M} \mid\mid \mathrm{N} \xrightarrow{(H\cup H')\neg(K\cup K'):\mathbf{arrive}(m)} \mathrm{M}' \mid\mid \mathrm{N}'} \textsc{Cast (T4-3)}$$

This case is proven if there exists a pair $(A_1, A_2) \in \mathscr{A}^{\sim}$ that satisfies $a \in A_1 \wedge \mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N}) \wedge \mathrm{M} \mid\mid \mathrm{N} \xrightarrow{A_2} \mathrm{M}' \mid\mid \mathrm{N}$, and the converse of Pair 2.13 satisfies this requirement.

5: Suppose that $\mathrm{T}(\mathrm{M})$ and $\mathrm{T}(\mathrm{N})$ both do the same transition $a \in C_1$. Following the induction hypothesis and eliminating pairs from the action relation $\mathscr{A}$ in Table 2.10 that do not match the transition labels from C_1, it can first be concluded that

$$\mathrm{T}(\mathrm{M}) \xrightarrow{C_1} \equiv \mathrm{T}(\mathrm{M}') \wedge \mathrm{M} \xrightarrow{\mathbf{connect}(ip',ip'')} \mathrm{M}' \wedge \mathrm{T}(\mathrm{N}) \xrightarrow{C_1} \equiv \mathrm{T}(\mathrm{N}') \wedge \mathrm{N} \xrightarrow{\mathbf{connect}(ip',ip'')} \mathrm{N}'$$

The mCRL2 derivation in the Lemma 4.7-proof for Connect (T4-1) shows how the first and third conjunct are sufficient to prove that $\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{C_1} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N})$.
On the AWN side, the second and fourth conjunct can be used as premise in

$$\frac{\mathrm{M} \xrightarrow{\mathbf{connect}(ip',ip'')} \mathrm{M}' \qquad \mathrm{N} \xrightarrow{\mathbf{connect}(ip',ip'')} \mathrm{N}'}{\mathrm{M} \mid\mid \mathrm{N} \xrightarrow{\mathbf{connect}(ip',ip'')} \mathrm{M}' \mid\mid \mathrm{N}'} \textsc{Connect (T4-1)}$$

This case is proven if there exists a pair $(A_1, A_2) \in \mathscr{A}^{\sim}$ that satisfies $a \in A_1 \wedge \mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N}) \wedge \mathrm{M} \mid\mid \mathrm{N} \xrightarrow{A_2} \mathrm{M}' \mid\mid \mathrm{N}$, and the converse of Pair 2.14 satisfies this requirement.

6: Suppose that $\mathrm{T}(\mathrm{M})$ and $\mathrm{T}(\mathrm{N})$ both do the same transition $a \in D_1$. Following the induction hypothesis and eliminating pairs from the action relation $\mathscr{A}$ in Table 2.10 that do not match the transition labels from D_1, it can first be concluded that

$$\mathrm{T}(\mathrm{M}) \xrightarrow{D_1} \equiv \mathrm{T}(\mathrm{M}') \wedge \mathrm{M} \xrightarrow{\mathbf{disconnect}(ip',ip'')} \mathrm{M}' \wedge \mathrm{T}(\mathrm{N}) \xrightarrow{D_1} \equiv \mathrm{T}(\mathrm{N}') \wedge \mathrm{N} \xrightarrow{\mathbf{disconnect}(ip',ip'')} \mathrm{N}'$$

The mCRL2 derivation below shows how the first and third conjunct are sufficient to prove that $\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{D_1} \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N})$:

$$\dfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{disconnect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \mathrm{T}(\mathrm{M}')}\ \text{Induction hypothesis} \qquad \dfrac{}{\mathrm{T}(\mathrm{N}) \xrightarrow{\mathbf{disconnect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \mathrm{T}(\mathrm{N}')}\ \text{Induction hypothesis}$$

$$\dfrac{\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{disconnect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \mathrm{T}(\mathrm{M}') \qquad \mathrm{T}(\mathrm{N}) \xrightarrow{\mathbf{disconnect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \mathrm{T}(\mathrm{N}')}{\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N}) \xrightarrow{\mathbf{disconnect}(\mathrm{U}(ip),\mathrm{U}(ip'))|\mathbf{disconnect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}')}\ \textsc{Par 3}$$

$$\dfrac{\cdots}{\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_C(\mathbf{disconnect}(\mathrm{U}(ip),\mathrm{U}(ip'))|\mathbf{disconnect}(\mathrm{U}(ip),\mathrm{U}(ip')))}\equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \textsc{Comm 2}$$

$$\dfrac{\cdots}{\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{d}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \text{Apply } \gamma_C$$

$$\dfrac{\cdots}{\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}(\mathbf{d}(\mathrm{U}(ip),\mathrm{U}(ip')))}\equiv \Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \textsc{Comm 2}$$

$$\dfrac{\cdots}{\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{d}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \text{Apply } \gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}$$

$$\dfrac{\mathbf{d} \in V \cup \{\tau\} \qquad \Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{d}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}{\nabla_V\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{d}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \nabla_V\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \textsc{Allow 2}$$

$$\dfrac{\cdots}{\rho_R\nabla_V\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{R\bullet\mathbf{d}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \rho_R\nabla_V\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \textsc{Rename 2}$$

$$\dfrac{\cdots}{\rho_R\nabla_V\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\mathbf{disconnect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \rho_R\nabla_V\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \text{Apply } R\bullet$$

$$\dfrac{\cdots}{\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{\mathbf{disconnect}(\mathrm{U}(ip),\mathrm{U}(ip'))}\equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N}')}\ \text{T15}$$

On the AWN side, the second and fourth conjunct can be used as premise in

$$\dfrac{\mathrm{M} \xrightarrow{\mathbf{disconnect}(ip',ip'')} \mathrm{M}' \qquad \mathrm{N} \xrightarrow{\mathbf{disconnect}(ip',ip'')} \mathrm{N}'}{\mathrm{M} \mid\mid \mathrm{N} \xrightarrow{\mathbf{disconnect}(ip',ip'')} \mathrm{M}' \mid\mid \mathrm{N}'}\ \textsc{Disconnect (T4-1)}$$

This case is proven if there exists a pair $(A_1, A_2) \in \mathscr{A}^{\smile}$ that satisfies $a \in A_1 \wedge \mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{A_1}\equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N}) \wedge \mathrm{M} \mid\mid \mathrm{N} \xrightarrow{A_2} \mathrm{M}' \mid\mid \mathrm{N}$, and the converse of Pair 2.15 satisfies this requirement.

7: Suppose that $\mathrm{T}(\mathrm{M})$ and $\mathrm{T}(\mathrm{N})$ both do the same transition a where $\underline{a} \notin \{\mathbf{arrive}, \mathbf{connect}, \mathbf{disconnect}\}$. The mCRL2 derivation below shows where the attempt to generate behavior for $\mathrm{T}(\mathrm{P} \langle\!\langle\, \mathrm{Q})$ under these circumstances fails:

$$\dfrac{\dfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{a}\equiv \mathrm{T}(\mathrm{M}')}\ \text{Induction hypothesis} \qquad \dfrac{}{\mathrm{T}(\mathrm{N}) \xrightarrow{a}\equiv \mathrm{T}(\mathrm{N}')}\ \text{Induction hypothesis}}{\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N}) \xrightarrow{a|a}\equiv \mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}')}\ \textsc{Par 3}$$

$$\dfrac{\cdots}{\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_C(a|a)}\equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \textsc{Comm 2}$$

$$\dfrac{\cdots}{\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{a|a}\equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \text{Apply } \gamma_C$$

$$\dfrac{\cdots}{\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}(a|a)}\equiv \Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \textsc{Comm 2}$$

$$\dfrac{\cdots}{\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{a|a}\equiv \Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \text{Apply } \gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}$$

The Allow 2 operator cannot be applied next (like in Case 6) because $\underline{a|a} \notin V \cup \{\tau\}$. Since this means that $\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N})$ cannot do a transition in mCRL2 under the circumstances specified in this particular case, there is no behavior for AWN to mimic.

8: Suppose that $\mathrm{T}(\mathrm{M})$ does a transition $a \in S_1$ and $\mathrm{T}(\mathrm{N})$ does a transition $b \in A_1$, or vice versa. Following the induction hypothesis and eliminating pairs from the action relation $\mathscr{A}$ in Table 2.10 that do not match the transition labels from S_1 or A_1, it can first be concluded that

$$\mathrm{T}(\mathrm{M}) \xrightarrow{S_1}\equiv \mathrm{T}(\mathrm{M}') \wedge \mathrm{M} \xrightarrow{R:^*\mathbf{cast}(m)} \mathrm{M}' \wedge \mathrm{T}(\mathrm{N}) \xrightarrow{A_1}\equiv \mathrm{T}(\mathrm{N}') \wedge \mathrm{N} \xrightarrow{H\neg K:\mathbf{arrive}(m)} \mathrm{N}'$$

The mCRL2 derivation in the Lemma 4.7-proof for Cast (T4-1) shows how the first and third conjunct are sufficient to prove that $\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{S_1} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N})$.
On the AWN side, the second and fourth conjunct can be used as premise in

$$H \subseteq R \wedge K \cap R = \emptyset \;\frac{\mathrm{M} \xrightarrow{R:*\mathbf{cast}(m)} \mathrm{M}' \qquad \mathrm{N} \xrightarrow{H \neg K:\mathbf{arrive}(m)} \mathrm{N}'}{\mathrm{M} \mid\mid \mathrm{N} \xrightarrow{R:*\mathbf{cast}(m)} \mathrm{M}' \mid\mid \mathrm{N}'}\ \textsc{Cast (T4-1)}$$

This case is proven if there exists a pair $(A_1, A_2) \in \mathscr{A}^{\sim}$ that satisfies $a \in A_1 \wedge \mathrm{T}(\mathrm{M} \mid\mid \mathrm{N}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{M}' \mid\mid \mathrm{N}) \wedge \mathrm{M} \mid\mid \mathrm{N} \xrightarrow{A_2} \mathrm{M}' \mid\mid \mathrm{N}$, and the converse of Pair 2.11 satisfies this requirement.
The proof for when $\mathrm{T}(\mathrm{M})$ does the **arrive** action and $\mathrm{T}(\mathrm{N})$ does the **starcast** action is similar.

9: Suppose that $\mathrm{T}(\mathrm{M})$ does a transition $a \in S_1$ and $\mathrm{T}(\mathrm{N})$ does a transition $b \notin A_1$ where $\underline{b} = \mathbf{arrive}$, or vice versa. The mCRL2 derivation below shows where the attempt to generate behavior for $\mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q})$ under these circumstances fails:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{M}')}\ \text{Induction hypothesis} \quad \dfrac{}{\mathrm{T}(\mathrm{N}) \xrightarrow{b} \equiv \mathrm{T}(\mathrm{N}')}\ \text{Induction hypothesis}}{\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N}) \xrightarrow{a|b} \equiv \mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}')}\ \textsc{Par 3}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_C(a|b)} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \text{Apply } \gamma_C}{\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{a|b} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \textsc{Comm 2}}{\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{\gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}(a|b)} \equiv \Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))}\ \text{Apply } \gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}$$
$$\Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \mid\mid \mathrm{T}(\mathrm{N})) \xrightarrow{a|b} \equiv \Gamma_{\{\mathbf{arrive}|\mathbf{arrive}\to\mathbf{a}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \mid\mid \mathrm{T}(\mathrm{N}'))$$

The ALLOW 2 operator cannot be applied next (like in Case 6) because $\underline{a|b} \notin V \cup \{\tau\}$. Note that the communication operator Γ_C has no effect because a and b have different arguments.
In conclusion, $\mathrm{T}([\,\mathrm{M}\,])$ cannot do a transition in mCRL2 under the circumstances specified in this particular case, and therefore there is no behavior for AWN to mimic.

10: Suppose that $\mathrm{T}(\mathrm{M})$ does a transition a and $\mathrm{T}(\mathrm{N})$ does a transition $b \neq a$ where $\{\underline{a}, \underline{b}\} \neq \{\mathbf{starcast}, \mathbf{arrive}\}$. Generating behavior for $\mathrm{T}(\mathrm{P} \langle\langle \mathrm{Q})$ under these circumstances fails for a similar reason as in Case 9: the communication operator Γ_C has no effect because $a \neq b$ (rather than that just their arguments are different) and therefore it is not possible to eventually apply the ALLOW 2 operator.

The derivations in mCRL2 from the cases listed above are representative derivations that collectively have no alternatives (see Definition 4.3): there are no other derivations with different or differently ordered one-way steps that yield different behavior of process expression $\mathrm{T}(\mathrm{M} \mid\mid \mathrm{N})$. Therefore the induction hypothesis generally holds for this expression.

Translation rule T16: The translation function T is partially defined by translation rule

$$\mathrm{T}([\,\mathrm{M}\,]) = \nabla_V \rho_{\{\mathbf{starcast}\to\mathbf{t}\}} \Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \quad \text{T16}$$
$$\text{where } V = \{\mathbf{t}, \mathbf{newpkt}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}$$
$$\text{where } C = \{\overline{\mathbf{newpkt}} | \mathbf{arrive} \to \mathbf{newpkt}\}$$
$$\text{where } \mathrm{H} = \textstyle\sum_{\mathtt{ip}:\mathrm{T}(\mathrm{IP}),\mathtt{data}:\mathrm{T}(\mathrm{DATA}),\mathtt{dest}:\mathrm{T}(\mathrm{IP})} \overline{\mathbf{newpkt}}(\{\mathtt{ip}\},\{\mathtt{ip}\},\mathtt{newpkt}(\mathtt{data},\mathtt{dest})).\mathrm{H}$$

Consider two pairs from Table 2.10:

$$(\{\, H\neg K : \mathbf{arrive}(m) \,\}, \left\{ \mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}', \mathrm{U}(m)) \;\middle|\; \begin{array}{c} \hat{\mathtt{D}},\hat{\mathtt{D}}'\in\mathrm{T}(\mathrm{Set(IP)}) \\ \hat{\mathtt{D}}'\subseteq\hat{\mathtt{D}} \\ \mathrm{U}(H)\subseteq\hat{\mathtt{D}}' \\ \mathrm{U}(K)\cap\hat{\mathtt{D}}'=\emptyset \end{array} \right\}),$$
$$(\{\, ip : \mathbf{newpkt}(d, dip) \,\}, \{\, \mathbf{newpkt}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip))) \,\})$$

Let A_1 and N_1 be defined as the second members of these pairs; that is, let

$$A_1 \stackrel{\text{def}}{=} \left\{ \mathbf{arrive}(\hat{\mathtt{D}}, \hat{\mathtt{D}}', \mathrm{U}(m)) \;\middle|\; \begin{array}{c} \hat{\mathtt{D}},\hat{\mathtt{D}}'\in\mathrm{T}(\mathrm{Set(IP)}) \\ \hat{\mathtt{D}}'\subseteq\hat{\mathtt{D}} \\ \mathrm{U}(H)\subseteq\hat{\mathtt{D}}' \\ \mathrm{U}(K)\cap\hat{\mathtt{D}}'=\emptyset \end{array} \right\}$$
$$N_1 \stackrel{\text{def}}{=} \{\, \mathbf{newpkt}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip))) \,\}$$

and let

$$\overline{N}_1 \stackrel{\text{def}}{=} \{\, \overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip))) \,\}$$

for some $d \in \mathrm{DATA}$, $ip \in \mathrm{IP}$, $H, K \in \mathrm{Set(IP)}$, and $m \in \mathrm{MSG}$.
The following cases are distinguished:

1: $\mathrm{T}(\mathrm{M})$ does a transition $a \in A_1$ and H does a transition $b \in \overline{N}_1$.
2: $\mathrm{T}(\mathrm{M})$ does a transition $a \notin A_1$ and H does a transition $b \in \overline{N}_1$.
3: $\mathrm{T}(\mathrm{M})$ does nothing and H does a transition $b \in \overline{N}_1$.
4: $\mathrm{T}(\mathrm{M})$ does a transition a where $\underline{a} = \mathbf{starcast}$ and H does nothing.
5: $\mathrm{T}(\mathrm{M})$ does a transition a where $\underline{a} \in \{\mathbf{t}, \mathbf{newpkt}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}$ and H does nothing.
6: $\mathrm{T}(\mathrm{M})$ does a transition a where $\underline{a} \notin \{\mathbf{t}, \mathbf{starcast}, \mathbf{newpkt}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}$ and H does nothing.

Note that these cases cover all combinations of behavior of $\mathrm{T}(\mathrm{M})$ and H (H can only do $\overline{\mathbf{newpkt}}$ actions; see the first part of the derivation in the Lemma 4.7-proof for Newpkt (T4), which is a representative derivation without alternatives).
The proof is provided below for each of the cases:

1: $\mathrm{T}(\mathrm{M})$ does a transition $a \in A_1$ and H does a transition $b \in \overline{N}_1$. Following the induction hypothesis and eliminating pairs from the action relation $\mathscr{A}$ in Table 2.10 that do not match the transition labels from A_1, it can first be concluded that

$$\mathrm{T}(\mathrm{M}) \xrightarrow{L_1} \equiv \mathrm{T}(\mathrm{M}') \wedge \mathrm{M} \xrightarrow{H\neg K:\mathbf{arrive}(m)} \mathrm{M}'$$

The mCRL2 derivation in the Lemma 4.7-proof for Newpkt (T4) shows how the first conjunct is sufficient to prove that $\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{L_1} \equiv \mathrm{T}([\,\mathrm{M}'\,])$. Under the constraints of this case, the derivation is a representative derivation without alternatives (see Definition 4.3) and so all related behavior of $\mathrm{T}([\,\mathrm{M}\,])$ is covered.
On the AWN side, the second conjunct can be used as premise in

$$\frac{\mathrm{M} \xrightarrow{\{ip\}\neg K:\mathbf{arrive}(\mathtt{newpkt}(d,dip))} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{ip:\mathbf{newpkt}(d,dip)} [\,\mathrm{M}'\,]}\ \textsc{Newpkt (T4)}$$

This case is proven if there exists a pair $(A_1,A_2) \in \mathscr{A}^{\sim}$ that satisfies $a \in A_1 \wedge \mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{A_1} \equiv \mathrm{T}([\,\mathrm{M}'\,]) \wedge [\,\mathrm{M}\,] \xrightarrow{A_2} [\,\mathrm{M}'\,]$, and the converse of Pair 2.16 satisfies this requirement.

2: $\mathrm{T}(\mathrm{M})$ does a transition $a \notin A_1$ and H does a transition $b \in \overline{N}_1$. This situation fails to generate behavior for $\mathrm{T}([\,\mathrm{M}\,])$ because no derivation similar to the one in the Lemma 4.7-proof for Newpkt (T4) is possible:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{T}(\mathrm{M}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{M}') \qquad \mathrm{H} \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \mathrm{H}}{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H} \xrightarrow{a|\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \mathrm{T}(\mathrm{M}') \,||\, \mathrm{H}}\ \textsc{Par 3}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\gamma_C(a|\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip))))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{a|\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \gamma_C}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\{\mathbf{starcast}\to\mathbf{t}\}\bullet a|\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Rename 2}}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{a|\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \{\mathbf{starcast}\to\mathbf{t}\}\bullet$$

The ALLOW 2 operator cannot be applied next because $\underline{a}|\overline{\mathbf{newpkt}} \notin V \cup \{\tau\}$. Since this means that $\mathrm{T}([\,\mathrm{M}\,])$ cannot do a transition in mCRL2 under the circumstances specified in this particular case, there is no behavior for AWN to mimic.

3: $\mathrm{T}(\mathrm{M})$ does nothing and H does a transition $\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip))) \in \overline{N}_1$. This situation fails to generate behavior for $\mathrm{T}([\,\mathrm{M}\,])$ because no derivation similar to the one in the Lemma 4.7-proof for Newpkt (T4) is possible:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{H} \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \mathrm{H}}{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H} \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}}\ \textsc{Par 5}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\gamma_C(\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip))))} \Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H})}\ \textsc{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H})}\ \text{Apply } \gamma_C}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\{\mathbf{starcast}\to\mathbf{t}\}\bullet\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H})}\ \textsc{Rename 2}}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\overline{\mathbf{newpkt}}(\{\mathrm{U}(ip)\},\{\mathrm{U}(ip)\},\mathtt{newpkt}(\mathrm{U}(d),\mathrm{U}(dip)))} \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H})}\ \text{Apply } \{\mathbf{starcast}\to\mathbf{t}\}\bullet$$

The ALLOW 2 operator cannot be applied next because $\overline{\mathbf{newpkt}} \notin V \cup \{\tau\}$. Since this means that $\mathrm{T}([\,\mathrm{M}\,])$ cannot do a transition in mCRL2 under the circumstances specified in this particular case, there is no behavior for AWN to mimic.

4: $\mathrm{T}(\mathrm{M})$ does a transition a where $\underline{a} = \mathbf{starcast}$ and H does nothing.

Following the induction hypothesis and eliminating pairs from the action relation $\mathscr{A}$ in Table 2.10 that do not match the transition label **starcast**, it can first be concluded that

$$\mathrm{T}(\mathrm{M}) \xrightarrow{\mathbf{starcast}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}(\mathrm{M}') \wedge \mathrm{M} \xrightarrow{R:\mathbf{*cast}(m)} \mathrm{M}'$$

for all $D,R \in \mathrm{Set(IP)}$ and $m \in \mathrm{MSG}$ where $R \subseteq D$.
The mCRL2 derivation in the Lemma 4.7-proof for Cast (T4-4) shows how the first conjunct is sufficient to prove that $\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{\mathbf{t}(\mathrm{U}(D),\mathrm{U}(R),\mathrm{U}(m))} \equiv \mathrm{T}([\,\mathrm{M}'\,])$. Under the constraints of this case, the derivation is a representative derivation without alternatives (see Definition 4.3) and so all related behavior of $\mathrm{T}([\,\mathrm{M}\,])$ is covered.
On the AWN side, the second conjunct can be used as premise in

$$\frac{\mathrm{M} \xrightarrow{R:\mathbf{*cast}(m)} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{\tau} [\,\mathrm{M}'\,]}\ \textsc{Cast (T4-4)}$$

This case is proven if there exists a pair $(A_1,A_2) \in \mathscr{A}^{\sim}$ that satisfies $a \in A_1 \wedge \mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{A_1} \equiv \mathrm{T}([\,\mathrm{M}'\,]) \wedge [\,\mathrm{M}\,] \xrightarrow{A_2} [\,\mathrm{M}'\,]$, and the converse of Pair 2.11 satisfies this requirement.

5: $\mathrm{T}(\mathrm{M})$ does a transition a where $\underline{a} \in \{\mathbf{t}, \mathbf{newpkt}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}$ and H does nothing. The induction hypothesis states that

$$\exists (A_1,A_2) \in \mathscr{A}^{\sim}\,.\, a \in A_1 \wedge \mathrm{T}(\mathrm{P}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}') \wedge \mathrm{P} \xrightarrow{A_2} \mathrm{P}'$$

Define $\mathscr{A}'$ as a subset of $\mathscr{A}^{\sim}$ that contains the possible values of $(A_1,A_2) \in \mathscr{A}^{\sim}$ where A_1 are actions that $\mathrm{T}(\mathrm{P} \mathbin{\langle\langle} \mathrm{Q})$ can perform in mCRL2 and where A_2 are the actions that AWN should use to mimic the actions in A_1 in order for this case to be proven:

$$\mathscr{A}' \stackrel{\text{def}}{=} \left\{ (A_1,A_2) \;\middle|\; (A_1,A_2) \in \mathscr{A}^{\sim}, \quad \mathrm{T}(\mathrm{P} \mathbin{\langle\langle} \mathrm{Q}) \xrightarrow{A_1} \equiv \mathrm{T}(\mathrm{P}' \mathbin{\langle\langle} \mathrm{Q}) \right\}$$

The following derivation is possible in mCRL2 for all $a \in A_1$ such that $(A_1,A_2) \in \mathscr{A}'$:

$$a \in V \cup \{\tau\}\ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{T}(\mathrm{M}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{M}')}\ \text{Induction hypothesis}}{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H} \xrightarrow{a} \equiv \mathrm{T}(\mathrm{M}') \,||\, \mathrm{H}}\ \textsc{Par 2}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\gamma_C(a)} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Comm 2}}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{a} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \gamma_C}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\{\mathbf{starcast}\to\mathbf{t}\}\bullet a} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Rename 2}}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{a} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \{\mathbf{starcast} \to \mathbf{t}\}\bullet}{\nabla_V \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{a} \equiv \nabla_V \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Allow 2}}{\mathrm{T}([\,\mathrm{M}\,]) \xrightarrow{a} \equiv \mathrm{T}([\,\mathrm{M}'\,])}\ \textsc{T16}$$

This derivation is valid only for $a \in A_1$ such that $\underline{a} \in V \cup \{\tau\}$. Under the constraints of this case, the derivation is also a representative derivation without alternatives (see

Definition 4.3). Finally it must be observed that it is impossible for $\mathrm{T}(\mathrm{M})$ to produce a **newpkt** action independent of H, meaning that $\underline{a} \neq \mathbf{newpkt}$. Consequently, $\mathscr{A}'$ is as follows:
{

$$
\begin{aligned}
&(\{\, \mathbf{t}(\mathrm{U}(D), \mathrm{U}(R), \mathrm{U}(m)) \,\}, \{\, \tau \,\}),\\
&(\{\, \mathbf{deliver}(\mathrm{U}(ip), \mathrm{U}(d)) \,\}, \{\, ip : \mathbf{deliver}(d) \,\}),\\
&(\{\, \mathbf{connect}(\mathrm{U}(ip'), \mathrm{U}(ip'')) \,\}, \{\, \mathbf{connect}(ip', ip'') \,\}),\\
&(\{\, \mathbf{disconnect}(\mathrm{U}(ip'), \mathrm{U}(ip'')) \,\}, \{\, \mathbf{disconnect}(ip', ip'') \,\})
\end{aligned}
$$

$\mid \quad m \in \mathrm{MSG}; \quad ip, ip', ip'' \in \mathrm{IP}; \quad d \in \mathrm{DATA}; \quad dests, R, D \in \mathrm{Set(IP)}; \quad R \subseteq D \quad \}$

For all $(A_1, A_2) \in \mathscr{A}'$ the actions A_1 in mCRL2 can be mimicked by the actions A_2 in AWN by means of the inference rules

$$\frac{\mathrm{M} \xrightarrow{\tau} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{\tau} [\,\mathrm{M}'\,]}\ \textsc{Internal}\ (\text{T4-3})$$

$$\frac{\mathrm{M} \xrightarrow{ip:\mathbf{deliver}(d)} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{ip:\mathbf{deliver}(d)} [\,\mathrm{M}'\,]}\ \textsc{Deliver}\ (\text{T4-3})$$

$$\frac{\mathrm{M} \xrightarrow{\mathbf{connect}(ip,ip')} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{\mathbf{connect}(ip,ip')} [\,\mathrm{M}'\,]}\ \textsc{Connect}\ (\text{T4-2})$$

$$\frac{\mathrm{M} \xrightarrow{\mathbf{disconnect}(ip,ip')} \mathrm{M}'}{[\,\mathrm{M}\,] \xrightarrow{\mathbf{disconnect}(ip,ip')} [\,\mathrm{M}'\,]}\ \textsc{Disconnect}\ (\text{T4-2})$$

respectively. Therefore the induction hypothesis holds for this case.

6: $\mathrm{T}(\mathrm{M})$ does a transition a where $\underline{a} \notin \{\mathbf{t}, \mathbf{starcast}, \mathbf{newpkt}, \mathbf{deliver}, \mathbf{connect}, \mathbf{disconnect}\}$ and H does nothing. This situation fails to generate behavior for $\mathrm{T}([\,\mathrm{M}\,])$ because no derivation similar to the one in the Lemma 4.7-proof for Newpkt (T4) is possible:

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{T}(\mathrm{M}) \xrightarrow{a} \equiv \mathrm{T}(\mathrm{M}')}{\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H} \xrightarrow{a} \equiv \mathrm{T}(\mathrm{M}') \,||\, \mathrm{H}}\ \textsc{Par}\ 2}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\gamma_C(a)} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Comm}\ 2}{\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{a} \equiv \Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \text{Apply } \gamma_C}{\rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}) \,||\, \mathrm{H}) \xrightarrow{\{\mathbf{starcast}\to\mathbf{t}\}\bullet a} \equiv \rho_{\{\mathbf{starcast}\to\mathbf{t}\}}\Gamma_C(\mathrm{T}(\mathrm{M}') \,||\, \mathrm{H})}\ \textsc{Rename}\ 2$$

The Allow 2 operator cannot be applied next because $\underline{a} \notin V \cup \{\tau\}$. Since this means that $\mathrm{T}([\,\mathrm{M}\,])$ cannot do a transition in mCRL2 under the circumstances specified in this particular case, there is no behavior for AWN to mimic.

Appendix F

Operators of the AWN input language

The AWN input language has the following unary and binary operators:

Signature	**Operator description**
`+` *Number*	Returns the operand.
`!` *Boolean*	Returns the logical negation of the operand.
`!` *Set*	Returns the complement of the operand.
`-` *Number*	Returns the arithmetic negation of the operand.
Number `+` *Number*	Returns the sum of two numbers.
List `+` *List*	Returns the concatenation of two lists.
Set `+` *Set*	Returns the union of two sets.
Number `-` *Number*	Returns the second operand subtracted from the first.
Set `-` *Set*	Returns the first operand without the members of the first.
Number `*` *Number*	Returns the product of two numbers.
Number `/` *Number*	Returns the quotient of two numbers.
Number `div` *Number*	Returns the quotient of two numbers, rounded down.
Number `mod` *Number*	Returns the remainder after calculating the quotient.
Number `^` *Number*	Calculates the exponentiation of two numbers.
Boolean `&&` *Boolean*	Returns *true* if both of the operands are *true*.
Boolean `||` *Boolean*	Returns *true* if one of the operands is *true*.
Boolean `^^` *Boolean*	Returns *true* if exactly one of the operands is *true*.
Type `in` *Set*	Returns *true* if both the first operand is a member of the second.
Type `in` *List*	Returns *true* if both the first operand is a member of the second.
Set `cap` *Set*	Returns the intersection of two sets.
Set `cup` *Set*	Returns the union of two sets.
Set `oplus` *Set*	Returns the symmetric difference of two sets.
Type `==` *Type*	Returns *true* if the operands are equal (requires the same type).
Type `<>` *Type*	Returns *true* if the operands are not equal (requires the same type).
Number `>` *Number*	Returns *true* if the the first operand is greater than the second one.
Number `<` *Number*	Returns *true* if the the second operand is greater than the first one.
Number `>=` *Number*	Returns *true* if the the first operand is greater than or equal to the second one.
Number `<=` *Number*	Returns *true* if the the second operand is greater than or equal to the first one.
Set `subset` *Set*	Returns *true* if the the first operand is a proper subset of the second operand.
Set `supset` *Set*	Returns *true* if the the second operand is a proper subset of the first operand.
Set `subseteq` *Set*	Returns *true* if the the first operand is a subset of the second operand.
Set `supseteq` *Set*	Returns *true* if the the second operand is a subset of the first operand.

Appendix G

Source files for leader election protocol

1 leader.awn

```
protocol LeaderProtocol;

type IP = struct(ip : Integer) extends $IP;
type B = struct(sip: IP, sn: Integer) extends $MSG;
type Trace = struct(ip: IP, lip: IP, lno: Integer) extends $TRACE;

//Main process
process Voting(lip: IP, lno: Integer, voted: Boolean, ip: IP, no: Integer)
uses m: $MSG, sip: IP, sn: Integer
  = receive(m) . [B(m) == new B(sip, sn)]
      Eval(sip, sn, lip, lno, voted, ip, no) /* receive ballot */
  + [!voted] (
        broadcast(new B(ip, no)) .
          Eval(ip, no, lip, lno, true, ip, no) /* cast a ballot */
      + receive(m) . [B(m) == new B(sip, sn)]
          Eval(sip, sn, lip, lno, voted, ip, no) /* receive ballot */
  )
  + trace(new Trace(ip, lip, lno)) . Voting(lip, lno, voted, ip, no)
;

//Helper process
process Eval(sip: IP, sn: Integer, lip: IP,
             lno: Integer, voted: Boolean, ip: IP, no: Integer)
  = [sn >= lno] Voting(sip, sn, voted, ip, no) /* vote better */
  + [sn < lno] Voting(lip, lno, voted, ip, no) /* vote worse */
;

const ALL_NODES: set of IP = {
    new IP(1), new IP(2), new IP(3), new IP(4), new IP(5)
  };

network
  NETWORK =
       new IP(1) : Voting(new IP(1), 8, false, new IP(1), 8) : ALL_NODES
    || new IP(2) : Voting(new IP(2), 2, false, new IP(2), 2) : ALL_NODES
    || new IP(3) : Voting(new IP(3), 5, false, new IP(3), 5) : ALL_NODES
    || new IP(4) : Voting(new IP(4), 4, false, new IP(4), 4) : ALL_NODES
    || new IP(5) : Voting(new IP(5), 8, false, new IP(5), 8) : ALL_NODES
;
```

Appendix H

Source files for AODV protocol

Here are some (partial) files that give an impression of the implementation of the AODV protocol:

1 main.awn

```
protocol MAIN;

import data;
import aodvProcess;
import qmsg;

network NETWORK =
     new IP(1):AODV(new IP(1), 1, [] of Route, [] of RouteRequest, [] of StoreEntry)
       << QMSG([new NewPkt(new Data(10), new IP(3))] of $MSG):{new IP(2)}
  || new IP(2):AODV(new IP(2), 1, [] of Route, [] of RouteRequest, [] of StoreEntry)
       << QMSG([new NewPkt(new Data(20), new IP(3))] of $MSG):{new IP(1), new IP(3)}
  || new IP(3):AODV(new IP(3), 1, [] of Route, [] of RouteRequest, [] of StoreEntry)
       << QMSG([] of $MSG):{new IP(2)}
;
```

2 aodv.awn

```
library aodvProcess;

import data;
import newpktProcess;
import pktProcess;
import rreqProcess;
import rrepProcess;
import rrerProcess;

sequential process AODV(ip: IP, sn: SQN, routeTable: RouteTable,
                        rreqs: RouteRequests, store: Store)
uses msg: $MSG, dip: IP, data: Data, dests: Dests,
     pre, pre2: set of IP, rreqid: RouteRequestID
  = receive(msg) . (
      [ msg is NewPkt ]
        trace(new Trace(ip, msg)) .
        NEWPKT(NewPkt(msg), ip, sn, routeTable, rreqs, store)
    + [ msg is Pkt ] PKT(Pkt(msg), ip, sn, routeTable, rreqs, store)
    + [ msg is RouteRequestMsg ]!
      [[routeTable := update(routeTable,
        new Route(RouteRequestMsg(msg).sip, 0, K::unknown,
```

```
22                    F::valid, 1, RouteRequestMsg(msg).sip, {} of IP))]]
23          RREQ(RouteRequestMsg(msg), ip, sn, routeTable, rreqs, store)
24      + [ msg is RouteReplyMsg ]
25        [[routeTable := update(routeTable,
26          new Route(RouteReplyMsg(msg).sip, 0, K::unknown,
27                    F::valid, 1, RouteReplyMsg(msg).sip, {} of IP))]]
28          RREP(RouteReplyMsg(msg), ip, sn, routeTable, rreqs, store)
29      + [ msg is RouteErrorMsg ]
30        [[routeTable := update(routeTable,
31          new Route(RouteErrorMsg(msg).sip, 0, K::unknown,
32                    F::valid, 1, RouteErrorMsg(msg).sip, {} of IP))]]
33          RERR(RouteErrorMsg(msg), ip, sn, routeTable, rreqs, store)
34    ) + [dip in (qD(store) cap vD(routeTable))] (
35      [[data := head(getQueue(store, dip))]]
36      unicast(nextHop(routeTable, dip), new Pkt(data, dip, ip)) . (
37        [[store := drop(dip, store)]]
38          AODV(ip, sn, routeTable, rreqs, store)
39      ) > (
40        [[dests := [new Dest(rip, inc(getSN(routeTable, rip)))
41           | rip in vD(routeTable)
42           @ nextHop(routeTable, rip) == nextHop(routeTable, dip)] ]]
43        [[routeTable := invalidate(routeTable, dests)]]
44        [[store := setRRF(store, dests)]]
45        [[ pre := with init x := {} of IP, dest in dests do
46             x + precs(routeTable, dest.ip)
47           end ]]
48        [[dests := [d | d in dests @ precs(routeTable, d.ip) != {} of IP] ]]
49          groupcast(pre, new RouteErrorMsg(dests, ip)) .
50          AODV(ip, sn, routeTable, rreqs, store)
51      )
52    ) + [dip in (qD(store) - vD(routeTable)) && getP(store, dip) == P::req] (
53        [[store := unsetRRF(store, dip)]]
54        [[sn := inc(sn)]]
55        [[rreqid := nrreqid(rreqs, ip)]]
56        [[rreqs := rreqs + [new RouteRequest(ip, rreqid)] ]]
57          broadcast(new RouteRequestMsg(0, rreqid, dip, getSN(routeTable, dip),
58                                        getK(routeTable, dip), ip, sn, ip)) .
59          AODV(ip, sn, routeTable, rreqs, store)
60    );
```

3 qmsg.awn

```
1  sequential process QMSG(msgs: list of $MSG) uses msg: $MSG
2    = receive(msg) . QMSG(msgs + [msg])
3    + [ msgs != [] of $MSG] (
4        send(head(msgs)) . QMSG(tail(msgs))
5      + receive(msg) . QMSG(msgs + [msg])
6    )
7  ;
```

4 data.awn

```
1  library data;
2
3  //Basic data types:
4  type SQN = Integer;
5  type Nat = Integer;
6  type RouteRequestID = Integer;
7  type K = enum(known, unknown);
8  type F = enum(valid, invalid);
9  type P = enum(req, no_req, undef);
10 type Data = struct(d: Integer) extends $DATA;
11
12 //More complicated data types:
13 type IP = struct(address: Integer) extends $IP;
14 type Route = struct(ip: IP, sqn: SQN, k: K, f: F,
15                     hopCount: Nat, nip: IP, precursors: set of IP);
16 type RouteTable = list of Route;
17 type Queue = list of Data;
18 type StoreEntry = struct(ip: IP, p: P, queue: Queue);
19 type Store = list of StoreEntry;
20 type RouteRequest = struct(ip: IP, id: RouteRequestID);
21 type RouteRequests = list of RouteRequest;
22 type Dest = struct(ip: IP, sqn: SQN);
23 type Dests = list of Dest;
24
25 //Traces:
26 type Trace = struct(ip: IP, msg: $MSG) extends $TRACE;
27
28 //Messages:
29 type NewPkt = struct(data: Data, dip: IP) extends $MSG;
30 type Pkt = struct(data: Data, dip, sip: IP) extends $MSG;
31 type RouteRequestMsg = struct(hopCount: Integer, id: RouteRequestID, dip: IP,
32        dsn: SQN, dsk: K, oip: IP, osn: SQN, sip: IP) extends $MSG;
33 type RouteReplyMsg = struct(hopCount: Integer, dip: IP,
34        dsn: SQN, oip: IP, sip: IP) extends $MSG;
35 type RouteErrorMsg = struct(dests: Dests, sip: IP) extends $MSG;
36 type Msg = struct() extends $MSG;
37
38 function drop(dst: IP, store: Store): Store
39   = with q := getQueue(store, dst) do
40     if |q| <= 1 then [ s | s in store @ s.ip != dst ] else
41       [ if s.ip == dst then s else
42           new StoreEntry(s.ip, s.p, tail(s.queue))
43         end | s in store ]
44     end
45   end;
46
47 //adds a packet to the queued data packets
48 function add(data: Data, dst: IP, store: Store): Store
49   = ifexists s in store @ s.ip == dst then
50       store - [s] + [new StoreEntry(dst, P::no_req, s.queue)]
```

```
51      else
52        store + [new StoreEntry(dst, P::req, [data])]
53      end;
54
55  //set the request-required flag to no-req
56  function unsetRRF(store: Store, dst: IP): Store
57    = ifexists s in store @ s.ip == dst then
58        store - [s] + [new StoreEntry(dst, P::no_req, s.queue)]
59      else
60        store
61      end;
62
63  //set the request-required flag to req
64  function setRRF(store: Store, dsts: Dests): Store
65    = [if exists(dst in dsts @ dst.ip == s.ip) then
66         new StoreEntry(s.ip, P::req, s.queue)
67       else s end | s in store];
68
69  //selects the data queue for a particular destination
70  function getQueue(store: Store, dst: IP): Queue
71    = ifexists s in store @ s.ip == dst then s.queue else [] of Data end;
72
73  //selects the flag for a destination from the store
74  function getP(store: Store, dst: IP): P
75    = ifexists s in store @ s.ip == dst then s.p else P::undef end;
76
77  //selects the route for a particular destination
78  partial function getRoute(routeTable: RouteTable, dst: IP): Route
79    = ifexists r in routeTable @ r.ip == dst then r else undefined Route end;
80
81  //increments the sequence number
82  function inc(sqn: SQN): SQN
83    = if sqn != 0 then sqn + 1 else sqn end;
84
85  //returns the larger sequence number
86  function max(sqn1, sqn2: SQN): SQN
87    = if sqn1 > sqn2 then sqn1 else sqn2 end;
88
89  //returns the sequence number of a particular route
90  function getSN(routeTable: RouteTable, dst: IP): SQN
91    = ifexists r in routeTable @ r.ip == dst then r.sqn else 0 end;
92
93  //determines whether the sequence number is known
94  function getK(routeTable: RouteTable, dst: IP): K
95    = ifexists r in routeTable @ r.ip == dst then r.k else K::unknown end;
96
97  ...
```

Appendix I

`TxtGen` language

1 Files

In the translation framework, models are translated to text by means of a custom-built model-to-text converter. This converter is driven by the `.txtgen` files, plain-text files written in the `TxtGen` language.

Model-to-text conversions must be specified in files that have the `.txtgen` extension. Files must start with a header that specifies their identifier, as in

```
1  library mCRL22text;
```

Files can import other files with the **import** syntax:

```
1  import "mCRL22text.txtgen";
```

2 Metamodels

`TxtGen` files work with models and must be given information about those models, which is contained in their metamodel. The code snippet below gives an example of how a metamodel in a given file is imported and linked to an identifier:

```
1  metamodel "mCRL2.ecore" as mCRL2;
```

Classes and other types contained within the metamodel are referenced by writing the identifier of the metamodel followed by `::` followed by the name of the type.

3 Rules

`TxtGen` files contains a number of *rules*, at least one of which must be a *main rule* – the `TxtGen` converter will look for one of these rules as a starting point for exporting a given model to text. `TxtGen` rules follow this grammar:

$$
\begin{aligned}
\textit{rule} &::= (\varepsilon \mid \textbf{main})\ \textit{ruleName}\texttt{(}\textit{ruleParam}\texttt{,}\ \cdots\ \texttt{,}\ \textit{ruleParam}\texttt{)}\ \texttt{=}\ \textit{ruleExpr}\texttt{;} \\
\textit{ruleParam} &::= \textit{paramName}\texttt{:}(\textit{primitiveType} \mid \textit{importedType}) \\
\textit{primitiveType} &::= \texttt{Boolean} \mid \texttt{Integer} \mid \texttt{String} \\
\textit{importedType} &::= \textit{metamodelName}\ \texttt{::}\ \textit{typeName}
\end{aligned}
$$

Each rule has a name (which does not have to be unique) and a number of typed parameters. Main rules must have exactly one parameter, namely one with the model type that it converts to text. How rules are referenced, is discussed in Section 4.2.

The body of a rule consists of an expression from which a text can be constructed. The following section describes the different expression types.

4 Rule expressions

The body of rules consists of a *rule expression*. Such expressions adhere to the following grammar:

$$
\begin{aligned}
\mathit{ruleExpr} &::= \mathit{seqExpr} \mid \cdots \mid \mathit{seqExpr} \\
\mathit{seqExpr} &::= \mathit{multExpr}\ \mathit{seqExpr} \mid \varepsilon \\
\mathit{multExpr} &::= \mathit{operand} \mid \mathit{operand}\ \texttt{?} \mid \mathit{operand}\ \texttt{*} \mid \mathit{operand}\ \texttt{+} \\
\mathit{operand} &::= \mathit{literal} \\
&\quad \mid \textbf{out}(\mathit{dataExpr}) \\
&\quad \mid \mathit{ruleName}(\mathit{dataExpr}, \cdots, \mathit{dataExpr}) \\
&\quad \mid \textbf{newline} \mid \textbf{indent} \mid \textbf{unindent} \\
&\quad \mid \textbf{newdir}(\mathit{string}) \mid \textbf{newfile}(\mathit{string}) \\
&\quad \mid \textbf{equal}(\mathit{dataExpr}, \mathit{dataExpr}) \mid \textbf{!equal}(\mathit{dataExpr}, \mathit{dataExpr}) \\
&\quad \mid \textbf{enum}(\mathit{dataExpr}, \mathit{dataExpr})) \\
&\quad \mid \textbf{error}(\mathit{string}, \mathit{dataExpr}, \cdots, \mathit{dataExpr})) \\
&\quad \mid (\mathit{ruleExpr}))
\end{aligned}
$$

4.1 Literals

The simplest scenario is where the expression generates literal text. For example, the rule

```
1  Sort(sort: mCRL2::IntSort) = "Int";
```

does not do anything with the contents of its parameter `sort`, but simply writes the text `"Int"` to the current file (without the quotes).

4.2 Primitive types

The contents of fields that have a primitive type can be written to the current file with the **out** function (the default Java `toString()` method is used for this purpose; more sophisticated value-to-text conversion is not supported). For example, let `map` be an object with the `String` field `name` and let that field contain the value `Hello world!`. The expression `out(map.name)` then writes `"Hello world!"` to the current file (without the quotes).

If the field provided as parameter of the **out** function has been assigned the value `null`, the expression fails and the `TxtGen` converter moves on to the next *alternative*. The concept of alternatives is discussed in Section 4.6.

4.3 Non-primitive types

If the field has a class – or any other non-primitive type – as type, it is typically exported to text by referencing another rule. Suppose, for example, that one uses the following rule to write mCRL2 mappings to the current file:

```
1  Map(m: mCRL2::Mapping) = "map " out(m.name) ": " Sort(m.sort) ";";
```

This expression depends on the existence of some rule named `Sort` that takes an mCRL2 sort as input.

If there are multiple rules that match the reference to a rule, `TxtGen` selects the one with parameter types that are as close to the types of the input values as possible. This means that it is possible to overload methods:

```
1  Sort(sort : M::Sort) = error("Missing support for sort ", type(sort));
2  Sort(sort : M::BoolSort) = "Bool";
3  Sort(sort : M::IntSort) = "Int";
```

In the code snippet above, `M::BoolSort` and `M::IntSort` are subclasses of `M::Sort`. If a reference to the `Sort` rule is made, the `TxtGen` converter selects the closest match: if the parameter is of type `M::BoolSort`, the second definition of the rule is chosen; if the parameter is of type `M::IntSort`, the third definition is chosen; and if the parameter is not a subtype of `M::BoolSort` or `M::IntSort` and there are no other matching definitions, the first definition is chosen, which generates an error (see Section 4.6).

4.4 Whitespace

`TxtGen` provides special expressions for inserting whitespace characters: the **indent** and **unindent** expressions set the current level of indentation, and the **newline** expression writes a newline character to the current file. A new line will be indented automatically as soon as more text is generated; before that, **indent** and **unindent** can be used to change the indentation of the line.

4.5 Directories and files

There are two expressions that allow the user to create directories and files, namely **newdir** and **newfile**. They both require a path string as input. Note that files can only be created in an existing directory.

The most recently created file is the file to where text is written. This means that a file must be created before any text is generated!

4.6 Alternatives

There are several situations in which a `TxtGen` expression 'fails', which means that it did not correctly generate text. One of these situations is when the contents of a field is converted to text with the **out** function and the contents is `null` (see Section 4.2. Another example involves *guards*:

- The **equal** function takes two parameters. Both parameters must be `null` or both parameters must have the same value as determined by the default Java `equals()` method in order for the current expression to succeed. The **!equal** function behaves opposite to the **equal** function.
- The **enum** function takes two parameters, the first of which must be the field of an object with an enumerable as its type, and the second must be one of the enumerable values. The function succeeds if and only if the field has been assigned the given value. The **!enum** function behaves opposite to the **enum** function.
- The **error** function always fails. It is useful for assisting the user by providing more information on what part of the model-to-text conversion failed with a message and the values of a number of objects.

It is important to note that the algorithm that checks whether an expression fails is *not* recursive. In other words, references to rules only fail if one of their parameters cannot be evaluated; they do not

fail if the body of the rules to which they refer fail!

Expressions in `TxtGen` of the form *seqExpr* | ⋯ | *seqExpr* denote lists of expressions that are explored from left to right, and the first sub-expression that does not fail is applied. The entire expression fails if there is all sub-expressions fail. The different sub-expressions are called *alternatives*.

4.7 Kleene operators

Expressions can be used as the parameter of the Kleene star operator or a related unary operator:

- The Kleene star denotes that its operand can occur zero or more times. For `TxtGen`, this means that the operand is repeated until it fails after which the converter continues with the subsequent expression. The operand with the Kleene star can therefore never fail (see Section 4.6), because the converter will simply continue with the subsequent expression even if the operand already fails at its first evaluation.
- The + operator behaves the same as the Kleene star operator, except that it requires the operator to be applied at least once: if the operator was not applied successfully at least one time, the expression fails.
- The ? operator means that its operand is optional. Just like the Kleene star operator, it cannot fail.

5 Data expressions

Data expressions are used in the parameters of the functions of rule expressions. They follow the grammar

$$
\begin{aligned}
\mathit{dataExpr} &::= \mathit{dataOperand}\ \mathit{selectFieldExpr} \\
\mathit{selectFieldExpr} &::= \varepsilon \mid \mathit{dataExpr}\,.\,\mathit{fieldName}\ \mathit{selectFieldExpr} \\
\mathit{dataOperand} &::= \mathit{paramName} \\
&\quad \mid \mathit{boolLiteral} \mid \mathit{intLiteral} \mid \mathit{stringLiteral} \\
&\quad \mid \textbf{type}(\mathit{dataExpr}) \\
&\quad \mid \textbf{first}(\mathit{dataExpr}) \\
&\quad \mid \textbf{next}(\mathit{dataExpr})
\end{aligned}
$$

First, the parameters and their fields can be used directly as a data expression; literal values of booleans, integers, and strings are also valid data expressions; and the **type** function converts the type of its parameter to a string (this is primarily used for debugging).

Second, each list in a model is given its own iterator that can be manipulated with the **first** and **next** functions: the **first** function selects the first element of the list, and the **next** function selects the next element of the list. These functions can fail – for example when a list is empty or when a list contains no more elements – which makes the rule expression that contains them fail (see Section 4.6). This means that the functions can be used in combination with the Kleene star operator $*$ (see Section 4.7) in order to export the elements of a list:

```
1  NamedProcess(p: M::NamedProcess)
2    = "proc" out(p.name) ("(" Variable(first(p.parameters))
3                                (", " Variable(next(p.parameters)))* ")")?
4      " = " Process(p.rootProcess) ";";
5  Variable(v: M::Variable) = ...
6  Process(p: M::Process) = ...
```

The code snippet above shows how a rule traverses all parameters of a named process and exports each one. If there are no parameters, no brackets are exported either.