University of Oslo
Department of Informatics

A fully general operational semantics for UML sequence diagrams with potential and mandatory choice

Mass Soldal Lund
Ketil Stølen

Research Report 330

ISBN 82-7368-285-4
ISSN 0806-3036

2007

Mass Soldal Lund
Ketil Stølen

Research Report 330
ISBN 82-7368-285-4
ISSN 0806-3036

Department of Informatics, University of Oslo, 2007.

A fully general operational semantics for UML sequence diagrams with potential and mandatory choice

Mass Soldal Lund and Ketil Stølen
University of Oslo, Norway
SINTEF ICT, Norway
{msl,kst}@sintef.no

Abstract

UML sequence diagrams is a specification language that has proved itself to be of great value in system development. When put to applications such as simulation, testing and other kinds of automated analysis there is a need for formal semantics. Such methods of automated analysis are by nature operational, and this motivates formalizing an operational semantics. In this report we present an operational semantics for UML 2.0 sequence diagrams that we believe gives a solid starting point for developing methods for automated analysis.

The operational semantics has been proved to be sound and complete with respect to a denotational semantics for the same language. It handles negative behavior as well as potential and mandatory choice. We are not aware of any other operational semantics for sequence diagrams of this strength.

Contents

1 Introduction

Unified Modeling Language (UML) sequence diagrams [44] and their predecessor Message Sequence Charts (MSC) [27] are specification languages that have proved themselves to be of great practical value in system development. When sequence diagrams are used to get a better understanding of the system through modeling, as system documentation or as means of communication between stakeholders of the system, it is important that the precise meaning of the diagrams is understood; in other words, there is need for a well-defined semantics. Sequence diagrams may also be put to further applications, such as simulation, testing and other kinds of automated analysis. This further increase the need for a formalized semantics; not only must the people who make and read diagrams have a common understanding of their meaning, but also the makers of methods and tools for analyzing the diagrams must share this understanding.

Methods of analysis like simulation and testing are in their nature operational; they are used for investigating what will happen when a system is executing. When developing techniques for such analysis, not only do we need to understand the precise meaning of a specification, we also need to understand precisely the executions that are specified. This motivates formalization of semantics in an operational style. In this report we present an operational semantics for UML sequence diagrams that we believe gives a solid starting point for developing such methods of analysis.

Obviously, choices must be made where the UML standard is ambiguous, but as far as possible the semantics is faithful to the standard. The semantics is easy to extend and modify. This allows us to give a “default” or “standard” interpretation, but also to experiment with the semantics and make variations on points unspecified by the standard. Specifically it has a formalized meta-level which allows definition of different execution strategies. It is not based on transformations to other formalisms, which makes it easy to work with.

In [19–21] a denotational semantics for sequence diagrams with potential and mandatory choice, called STAIRS, is presented. The operational semantics presented in this report has been proved to be sound and complete with respect to this denotational semantics.

The structure of this report is the following: In section 2 we present the background of the operational semantics, and in section 3 the syntax over which the semantics is defined. Section 4 presents the operational semantics itself, and in section 5 we present our soundness and completeness results. In section 6 we present related work and, finally, in section 7 conclusions are provided. A short presentation of the denotational semantics of STAIRS is provided in appendix A. Appendix B provides detailed proofs.

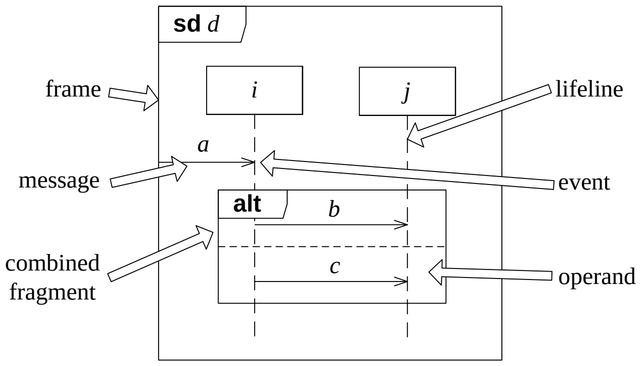


Figure 1: Sequence diagram

2 Background

2.1 Sequence diagrams

Sequence diagrams is a graphical specification language defined in the UML 2.0 standard [44].[1] The standard defines the graphical notation, but also an abstract syntax for the diagrams. Hence the language has a well-defined syntax.

Figure 1 shows a sequence diagram d in the graphical notation. A sequence diagram consists of a *frame*, representing the environment of the specified system, and one or more *lifelines*[2], representing components of the system. Arrows represent *messages* sent between lifelines or between a lifeline and the environment, and if the beginning or end of an arrow is at a lifeline this represents an *event*. *Combined fragments* are operators, like the choice operator alt, and each combined fragment has one or more *operands*.

2.2 Basic semantic model

The UML standard provides semantics of sequence diagrams. This semantics, however, is informal and defined by the means of natural language. Most notably, this is a trace based semantics:

> The semantics of an Interaction is given as a pair of sets of traces. The two trace sets represent valid traces and invalid traces. The union of these two sets need not necessarily cover the whole universe of traces. The traces that are not included are not described by this Interaction at all, and we cannot know whether they are valid or invalid. [. . .]
>
> **Basic trace model:** The semantics of an Interaction is given by a pair $[P, I]$ where P is the set of valid traces and I is the set of invalid traces. $P \cup I$ need not be the whole universe of traces.
>
> A trace is a sequence of event occurrences denoted $\langle e1, e2, ..., en\rangle$. [. . .]
>
> Two Interactions are equivalent if their pair of trace-sets are equal. [44, p. 468]

[1]In the UML standard, *Interaction* is used as the common name for diagrams specifying interaction by sending and receiving of messages. Sequence diagrams are then one kind of Interaction.

[2]In "MSC-terminology", lifelines are called *instances* or *instance lines*. When these expressions occur in this report they should be understood as synonyms to "lifeline".

In [19–21] a denotational semantics for sequence diagrams is formalized. We will refer to this as the STAIRS semantics. The STAIRS semantics is trace based and uses an extended version of the basic semantic model from the UML standard. Instead of a single pair (p, n) of positive (valid) and negative (invalid) traces, the semantic model of STAIRS is a set of pairs $\{(p_1, n_1), (p_2, n_2), \ldots, (p_m, n_m)\}$. A pair (p_i, n_i) is referred to as an *interaction obligation*. The word "obligation" is used in order to emphasize that an implementation of a specification is required to fulfill every interaction obligation of the specification. This semantic model makes it possible to distinguish between potential and mandatory choice (see section 2.3.5).

A trace is a (finite or infinite) sequence of events $\langle e_1, e_2, \ldots, e_i, \ldots \rangle$. We let $t_1 ⁀ t_2$ denote concatenation of the traces t_1 and t_2 and $\langle\rangle$ denote the empty trace. Let $\mathcal{H}$ be the trace universe. For each interaction obligation (p_i, n_i) we have that $p_i \cup n_i \subseteq \mathcal{H}$. All interaction obligations are independent of each other, and an interaction obligation is allowed to be inconsistent (i.e., we allow $p_i \cap n_i \neq \emptyset$).

In the following we assume the semantic model of STAIRS, and regard the STAIRS semantics as a correct formalization of sequence diagrams. This means that the correctness of our operational semantics is evaluated with respect to the denotational semantics of STAIRS. A presentation of the denotational semantics is found in appendix A.

2.3 Challenges

There is a number of challenges connected with making semantics for sequence diagrams. The reminder of this section is dedicated to looking into these challenges in more detail.

2.3.1 Partial vs. complete specification

An old discussion related to sequence diagrams and MSCs is whether a specification consisting of a set of diagrams represents the full behavior of the specified system or just examples of the behavior. In the former case we say that the set of sequence diagrams is a complete specification, and in the latter case a partial specification.

The UML standard states clearly that sequence diagrams are partial specifications since "[the union of valid and invalid traces] need not be the whole universe of traces." [44, p. 468]

This makes defining an operational semantics somewhat more complicated than in the case of complete specifications. It rules out solutions such as just viewing sequence diagrams as, or translating them to, other formalisms for making complete specifications, such as transition systems. In effect, the trace universe becomes a quantity that must be considered in the definition of the semantics. This excludes a solution where the invalid traces are considered just the complement of the valid traces.

2.3.2 Global vs. local view

In a certain sense, a sequence diagram has a global and a local view at the same time. The lifelines of a sequence diagram do not synchronize and represent processes that execute independently. This is the local view of the sequence diagram. At the same time, the syntax allows operators that cover several lifelines, and hence provide a global view. The best example is the alternative operator alt without guards, i.e. a non-deterministic choice. Since the lifelines are independent and do not synchronize, one of the lifelines may start executing the arguments of the operator before the others. The choice, however, is global in the sense that all lifelines must choose the same argument when resolving the choice. The semantics must reflect this duality by providing both the local and the global view of the specifications. In [29, p. 369], Jonsson and Padilla make the same observation:

> The transition system will [...] maintain a local state for each instance [...] The execution [...] may follow either of two paths, depending on which alternative is chosen in the [alt]. The crucial point here is that all three instances must choose the same alternative even if they do not arrive simultaneously to the [alt] in their execution. [...] Thus, [one instance's] entry into the first alternative has global consequences in that it "forces" the other instances to also choose the first alternative.

The result of this is that the semantics must reflect this duality; the semantics must provide both the local and the global view of the specifications. A discussion of this problem is also found in [31].

2.3.3 Weak sequencing

The weak sequencing operator seq is the implicit operator for composing sequence diagrams. The operator defines a partial order of the events in a diagram, such that the order along each lifeline and the causal relationship between the transmission and the reception of messages are preserved while any other ordering of events is arbitrary. Further there is no implicit synchronization between the lifelines in a sequence diagram.

An operational semantics must characterize the step by step execution specified by the diagram. For each step, all enabled events must be selectable. This poses a challenge, since due to the weak sequencing, there may be enabled events at arbitrary depth in the syntactical term representing the diagram.

2.3.4 Negative behavior

UML 2.0 allows negative[3] behavior to be specified by the neg operator in sequence diagrams. In the semantic model, these behaviors end up in the set of invalid traces (see section 2.2). In denotational semantics like STAIRS or the trace semantics of Harald Störrle [49, 50] this is easily handled by manipulation of the traces and trace sets (even though these two formalizations do not agree on the interpretation of neg).

It is however not clear what negative behavior means in an operational semantics. If an operational semantics should describe a step by step the execution of a sequence diagram it is not clear how we should distinguish a valid execution from an invalid execution. Immediate abortion, which is the naive solution, is not satisfactory for two reasons: 1) We may be interested in complete invalid

[3]We will use the terms *negative* and *invalid*. Further, *positive* and *valid* should be understood to be synonyms when we talk about behaviors.

executions, and 2) we need to know that the execution was stopped because it reached a state in which it became invalid. The alternative is some way of waving a flag. With this solution a meta-level at which the flag is seen and the behavior interpreted as invalid is needed in order to assign meaning to negative behavior.

We cannot know at the execution level that we are in an invalid execution, and it seems obvious that the only way of assigning meaning to negative behavior is by adding a meta-level on which we may interpret behavior as valid or invalid.

Even though he does not need it for his denotational semantics, Störrle mentions the option:

> One might also consider [the] negate-operators as being meta-logical in the sense that they express properties of traces rather that defining or modifying traces. [50]

The solution for Live Sequence Charts (LSC) (for more on LSC see section 6) is described in [17, p. 88]:

> A hot condition [...] must always be true. If an execution reaches a hot condition that evaluates to false this is a violation of the requirements, and the system should abort. For example, if we form an LSC from a prechart *Ch* and a main chart consisting of a single *false* hot condition, the semantics is that *Ch* can never occur. In other words, it is forbidden, an *anti-scenario*.

Even though they choose to abort an invalid execution, this is guided by a meta variable *Violating* in their operational semantics [16].

Cengarle and Knapp offers two ways of interpreting negative behavior. The first [7], is based on *satisfies* relations. Two such relations are provided: *positively satisfies*, $t \models_p S$, meaning that t is a valid trace of the diagram S, and *negatively satisfies*, $t \models_n S$, meaning that t is an invalid trace of S. The second [8] is an operational semantics. Two transition relations $S \xrightarrow{e}_p S'$ and $S \xrightarrow{e}_n S'$ are provided, where the former means that specification S may produce event e in a positive execution and the latter that S may produce e in a negative execution. Their exact interpretation of negative behavior is not relevant in this discussion, but we notice that also their solutions resort to use of a meta-level to distinguish between valid and invalid traces.

Whether it is obtained by waving a flag, like in the LSC semantics, or by defining an extra structure, like in the approach of Cengarle and Knapp, an extra meta-level is needed for giving an interpretation to neg.

2.3.5 Potential vs. mandatory choice

A final question is the ambiguity of the alternative operator alt, whether the non-determinism introduced by the alternative operator represents underspecification or real choices in the specified system.

It can be argued that the alternative operator is useful for underspecification, i.e. that it represents a design decision to be made later, but practitioners often interpret it as a choice between two alternatives that both should be present in the implementation. Both interpretations have some problems attached to them.

The first interpretation fits well with the observation that sequence diagrams are used for high level specification and that several steps of refinement are needed on the way to implementation. From this viewpoint it makes sense to interpret an alternative as underspecification. A problem with this is that if a sequence diagram is interpreted as a partial specification, the specification become very weak; in effect every behavior is allowed if no negative behavior is specified. Another drawback is that it is not clear how to represent non-deterministic choices that should be preserved in the implementation. Such choices are essential in, e.g., specification of security properties [28].

It may be argued that the second interpretation is more intuitive because the choice operator is used for representing choices in the system and not design choices. The drawback is of course that the possibility of underspecification is then restricted.

In [20, 21] this ambiguity is resolved by interpreting alt in the first sense, as underspecification, and by introducing xalt (explicit alternative) as a choice operator in the second sense; a choice that represents a choice in the system.

It is however not possible to distinguish these two kinds of choices at the execution level; for a single execution it is irrelevant whether a choice is specified by an alt or an xalt. But the distinction is relevant with respect to, e.g., refinement and implementation, which is to say at the meta-level. As with the operator neg, an extra meta-level is needed for giving an interpretation of xalt and to distinguish between the two kinds of choice.

3 Syntax

The graphical notation of sequence diagrams is not suited as a basis for defining semantics, and the abstract syntax of the UML standard contains more information than we need for the task. Our operational semantics is defined over a simpler abstract syntax defined in [19, 21]. This is an event-centric syntax in which the weak sequential operator seq is employed as the basic construct for combining diagram fragments.

The atom of a sequence diagram is the *event*. An event consists of a *message* and a *kind* where the kind decides whether it is the *transmit* or the *receive* event of the message. A message is a *signal*, which represents the contents of the message, together with the addresses of the transmitter and the receiver. Formally a signal is a label, and we let $\mathcal{S}$ denote the set of all signals. The transmitters and receivers are lifelines. Let $\mathcal{L}$ denote the set of all lifelines. A message m is defined as a triple

$$(s, t, r) \in \mathcal{S} \times \mathcal{L} \times \mathcal{L}$$

with signal s, transmitter t and receiver r. $\mathcal{M}$ denotes the set of all messages. On messages we define a transmitter function $tr.\_ \in \mathcal{M} \rightarrow \mathcal{L}$ and a receiver function $re.\_ \in \mathcal{M} \rightarrow \mathcal{L}$:

$$tr.(s, t, r) \stackrel{\mathsf{def}}{=} t \qquad\qquad re.(s, t, r) \stackrel{\mathsf{def}}{=} r$$

We let $\mathcal{K} = \{!, ?\}$ be the set of kinds, where ! represents transmit and ? represents receive. An event e is then a pair of a kind and a message:

$$(k, m) \in \mathcal{K} \times \mathcal{M}$$

$\mathcal{E}$ denotes the set of all events. On events we define a kind function $k.\_ \in \mathcal{E} \rightarrow \mathcal{K}$ and a message function $m.\_ \in \mathcal{E} \rightarrow \mathcal{M}$:

$$k.(k,m) \overset{\mathsf{def}}{=} k \qquad\qquad m.(k,m) \overset{\mathsf{def}}{=} m$$

We let the transmitter and receiver functions also range over events, $tr.\_, re.\_ \in \mathcal{E} \rightarrow \mathcal{L}$, and define a lifeline function $l.\_ \in \mathcal{E} \rightarrow \mathcal{L}$ that returns the lifeline of an event:

$$tr.(k,m) \overset{\mathsf{def}}{=} tr.m \qquad re.(k,m) \overset{\mathsf{def}}{=} re.m \qquad l.e \overset{\mathsf{def}}{=} \begin{cases} tr.e & \textbf{if } k.e = \,! \\ re.e & \textbf{if } k.e = \,? \end{cases}$$

A sequence diagram is built out of events, the binary operators seq, par, alt and xalt, and the unary operators neg and loop. Related to the graphical syntax, the operators represent combined fragments and their arguments the operands. In addition we let skip represent the empty sequence diagram. Let $\mathcal{D}$ be the set of all syntactically correct sequence diagrams. $\mathcal{D}$ is defined recursively as follows:[4]

$$\begin{array}{lll}
 & & \mathsf{skip} \in \mathcal{D} \\
e \in \mathcal{E} & \Rightarrow & e \in \mathcal{D} \\
d_1, d_2 \in \mathcal{D} & \Rightarrow & d_1 \ \mathsf{seq}\ d_2 \in \mathcal{D} \wedge d_1 \ \mathsf{par}\ d_2 \in \mathcal{D} \wedge \\
 & & d_1 \ \mathsf{alt}\ d_2 \in \mathcal{D} \wedge d_1 \ \mathsf{xalt}\ d_2 \in \mathcal{D} \\
d \in \mathcal{D} & \Rightarrow & \mathsf{neg}\ d \in \mathcal{D} \\
d \in \mathcal{D} \wedge I \subseteq (\mathbb{N} \cup \{0, \infty\}) & \Rightarrow & \mathsf{loop}\ I\ d \in \mathcal{D} \\
d \in \mathcal{D} \wedge n \in (\mathbb{N} \cup \{0, \infty\}) & \Rightarrow & \mathsf{loop}\langle n \rangle\ d \in \mathcal{D}
\end{array}$$

In the definitions of the two loops we have that $\mathbb{N}$ is the set of non-zero natural numbers and ∞ is a number greater than all other numbers and has the property $\infty - 1 = \infty$. The intention behind $\mathsf{loop}\ I\ d$ is that d should be looped any number $n \in I$ times. The UML standard describes two loops $\mathsf{loop}(n)$ and $\mathsf{loop}(n,m)$, where n is the minimum number and m the maximum number of iterations. We may define these as:

$$\begin{array}{rcl}
\mathsf{loop}(n)\ d & \overset{\mathsf{def}}{=} & \mathsf{loop}\ [n..\infty]\ d \\
\mathsf{loop}(n,m)\ d & \overset{\mathsf{def}}{=} & \mathsf{loop}\ [n..m]\ d
\end{array}$$

As can be expected, we have associativity of seq, par, alt and xalt. We also have commutativity of par, alt and xalt. Proofs with respect to the denotational semantics can be found in [19]. Furthermore the empty sequence diagram skip is the identity element of seq and par. The combination of skip and loop is discussed in section 4.2.1.

In this abstract syntax the diagram of figure 1 is expressed as:[5]

$$d = (?, (a, env, i))\ \mathsf{seq}\ ((!, (b, i, j))\ \mathsf{seq}\ (?, (b, i, j))\ \mathsf{alt}\ (!, (c, i, j))\ \mathsf{seq}\ (?, (c, i, j)))$$

We define a function $ll.\_ \in \mathcal{D} \rightarrow \mathbb{P}(\mathcal{L})$ that returns the set of lifelines present

[4]The set $\mathcal{D}$ is somewhat restricted by some additional syntactically constraints, see section A.1 in appendix A for details.

[5]Here we let env denote the environment of the diagram. Formally this is a gate, but gates are outside the scope of this report. Also note that seq binds stronger than alt.

in a diagram. For $e \in \mathcal{E}$ and $d, d_1, d_2 \in \mathcal{D}$ the function is defined recursively:

$$\begin{array}{lll@{\qquad}lll}
ll.\mathsf{skip} & \overset{\mathsf{def}}{=} & \emptyset & ll.(d_1 \;\mathsf{seq}\; d_2) & \overset{\mathsf{def}}{=} & ll.d_1 \cup ll.d_2 \\
ll.e & \overset{\mathsf{def}}{=} & \{l.e\} & ll.(d_1 \;\mathsf{par}\; d_2) & \overset{\mathsf{def}}{=} & ll.d_1 \cup ll.d_2 \\
ll.(\mathsf{neg}\; d) & \overset{\mathsf{def}}{=} & ll.d & ll.(d_1 \;\mathsf{alt}\; d_2) & \overset{\mathsf{def}}{=} & ll.d_1 \cup ll.d_2 \\
ll.(\mathsf{loop}\; I\; d) & \overset{\mathsf{def}}{=} & ll.d & ll.(d_1 \;\mathsf{xalt}\; d_2) & \overset{\mathsf{def}}{=} & ll.d_1 \cup ll.d_2 \\
ll.(\mathsf{loop}\langle n \rangle\; d) & \overset{\mathsf{def}}{=} & ll.d & & &
\end{array}$$

4 Operational semantics

We argue that there is a need for an operational semantics of sequence diagrams in addition to denotational semantics like the STAIRS semantics. Before proceeding with the task of defining an operational semantics, let us shed some light on the distinction between operational and denotational semantics. David A. Schmidt [47, p. 3] suggests the following:

> The *operational semantics* method uses an interpreter to define a language. The meaning of a program in the language is the evaluation history that the interpreter produces when it interprets the program. The evaluation history is a sequence of internal configurations [...]
>
> The *denotational semantics* method maps a program directly to its meaning, called its *denotation*. The denotation is usually a mathematical value, such as a number or function. No interpreters are used; a *valuation function* maps a program directly to its meaning.

As a methodology for language development he suggests that "a denotational semantics is defined to give the meaning of the language" and that "the denotational definition is implemented using an operational definition" [47, p. 4]. Hoare and He [23, p. 258] describe more explicitly the notion of an operational semantics:

> An *operational* semantics of a programming language is one that defines not the observable overall effect of a program but rather suggests a complete set of possible individual steps which may be taken in its execution. The observable effect can then be obtained by embedding the steps into an iterative loop [...]

Taken together these two descriptions suggest that formalizing an operational semantics of a language is to define an interpreter for the language. The formal definition of the interpreter describes every step that can be made in the execution of the language in such a way that the executions are in conformance with the meaning of the language as defined by a denotational semantics. In our case the input to the interpreter is a sequence diagram represented in the abstract syntax defined above. The output of the interpreter is a trace of events representing an execution.

Our solution to the challenges identified in section 2.3 is the combination of two transition systems, which we refer to as the *execution system* and the *projection system*. The execution system is a transition system over

$$[\_,\_] \in \mathcal{B} \times \mathcal{D} \tag{1}$$

where $\mathcal{B}$ represents the set of all states of the communication medium and $\mathcal{D}$ the set of all syntactically correct sequence diagrams. We let $\mathcal{EX} \overset{\mathsf{def}}{=} \mathcal{B} \times \mathcal{D}$, and refer to the elements of $\mathcal{EX}$ as execution states.

The projection system is a transition system over

$$\Pi(\_,\_,\_) \in \mathbb{P}(\mathcal{L}) \times \mathcal{B} \times \mathcal{D} \tag{2}$$

where $\mathbb{P}(\mathcal{L})$ is the powerset of the set of all lifelines. The projection system is used for finding enabled events at each stage of the execution and is defined recursively. This system handles the challenges related to weak sequencing and related to the global vs. the local view in sequence diagrams.

These two systems work together in such a way that for each step in the execution, the execution system updates the projection system by passing on the current state of the communication medium, and the projection system updates the execution system by selecting the event to execute and returning the state of the diagram after the execution of the event.

We also formalize a meta-level that encloses the execution system. At this meta-level we may define several meta-strategies that guide the execution and are used for handling the challenges related to negative behavior, and potential and mandatory choice.

4.1 The execution system

The execution system has two rules. The first rule represents the execution of a single event and uses the projection system to find an enabled event to execute. It is defined as:

$$[\beta, d] \xrightarrow{e} [update(\beta, e), d'] \textbf{ if } \Pi(ll.d, \beta, d) \xrightarrow{e} \Pi(ll.d, \beta, d') \wedge e \in \mathcal{E} \tag{3}$$

In general we assume the structure of the communication medium, i.e. the means of communication, to be underspecified. The only requirement is that the following functions are defined:

- $add \in \mathcal{B} \times \mathcal{M} \rightarrow \mathcal{B}$: Adds a message.
- $rm \in \mathcal{B} \times \mathcal{M} \rightarrow \mathcal{B}$: Removes a message.
- $ready \in \mathcal{B} \times \mathcal{M} \rightarrow \mathbb{B}$: Returns **true** if the communication medium is in a state where it can deliver the message and **false** otherwise.

The function $update \in \mathcal{B} \times \mathcal{E} \rightarrow \mathcal{B}$ is defined as:

$$update(\beta, e) \overset{\mathsf{def}}{=} \begin{cases} add(\beta, m.e) & \textbf{if } k.e = ! \\ rm(\beta, m.e) & \textbf{if } k.e = ? \end{cases} \tag{4}$$

Since transmitter and receiver information is embedded into the messages, these functions are sufficient. In this report we assume the most general communication model, i.e. no ordering on the messages. This means that, e.g., message overtaking is possible. Formally then, $\mathcal{B}$ may be defined as the set of all multisets over $\mathcal{M}$, *add* as multiset union, *rm* as multiset minus and *ready* as multiset containment.

The second rule of the execution system executes silent events. The rules of the projection system handle the sequence diagram operators alt, xalt, neg and loop. Resolving these operators, such as choosing the branch of an alt, are considered silent events. We define the set of silent events to be

$$\mathcal{T} = \{\tau_{alt}, \tau_{xalt}, \tau_{neg}, \tau_{pos}, \tau_{loop}\} \tag{5}$$

with $\mathcal{T} \cap \mathcal{E} = \emptyset$. The reason for introducing all these different silent events is that they give high flexibility in defining execution strategies by making the silent events and their kinds available at the meta-level. The rule is simple:

$$[\beta, d] \xrightarrow{\tau} [\beta, d'] \textbf{ if } \Pi(ll.d, \beta, d) \xrightarrow{\tau} \Pi(ll.d, \beta, d') \wedge \tau \in \mathcal{T} \tag{6}$$

The empty diagram skip cannot be rewritten, but we assert that it produces the empty trace, i.e.:

$$[\beta, \mathsf{skip}] \xrightarrow{\langle\rangle} [\beta, \mathsf{skip}] \tag{7}$$

This also means that execution terminates when skip is reached.

4.2 The projection system

In the following sub-sections we present the rules of the projection system for each of the syntactic constructs.

4.2.1 The empty diagram

It is not possible to rewrite $\Pi(L, \beta, \mathsf{skip})$. skip is the identity element of seq and par, and we therefore have that $\mathsf{skip}\ \mathsf{seq}\ d$, $d\ \mathsf{seq}\ \mathsf{skip}$, $\mathsf{skip}\ \mathsf{par}\ d$ and $d\ \mathsf{par}\ \mathsf{skip}$ are treated as identical to d. We can also note that $\mathsf{skip}\ \mathsf{alt}\ \mathsf{skip} = \mathsf{skip}$ and $\mathsf{skip}\ \mathsf{xalt}\ \mathsf{skip} = \mathsf{skip}$. We do not allow the construction $\mathsf{neg}\ \mathsf{skip}$.

$\mathsf{loop}\langle\infty\rangle\ \mathsf{skip}$ is more problematic (see section 4.2.7 for the definition of $\mathsf{loop}\langle\infty\rangle$). Seen as a program, this construct is similar to the java fragment `while(true) { }`, i.e., a program that produces nothing and never terminates. When related to the denotational semantics, however, the semantics of $\mathsf{loop}\langle\infty\rangle\ \mathsf{skip}$ should be the empty trace $\langle\rangle$, since the denotational semantics characterize observation after infinite time. A simple solution would be to syntactically disallow the construct all together. Because we do not want to make too many syntactic constraints, and because we want to stay close to the denotational semantics we choose to let $\mathsf{loop}\langle\infty\rangle\ \mathsf{skip}$ reduce to skip, even though this may be seen as counter-intuitive from an operational point of view.

4.2.2 Event

The simplest case is the diagram consisting of only one event e. In this case the system delivers the event if the event is enabled given the set L of lifelines and the state of the communication medium. This means first that the event must belong to one of the lifelines in the set L, and secondly that the event either must be a transmit event or its message must be available in the communication medium. The need for L will be evident in the definition of rules for seq below.

$$\begin{array}{l}\Pi(L, \beta, e) \xrightarrow{e} \Pi(L, \beta, \mathsf{skip}) \\ \quad \textbf{if } l.e \in L \wedge (k.e = ! \vee ready(\beta, m.e))\end{array} \tag{8}$$

4.2.3 Weak sequencing

The weak sequencing operator seq defines a partial order on the events in a diagram; the ordering of events on each lifeline and between the transmit and receive of a message is preserved, but all other ordering of events is arbitrary. Because of this, there may be enabled events in both the left and the right argument of a seq if there are lifelines present in the right argument of the operator that are not present in the left argument. This leads to two rules for the seq operator.

If there is an overlap between the given set of lifelines and the lifelines of the left hand side of the seq, this means that the lifelines in this intersection may have enabled events on the left hand side only. Hence, with respect to these lifelines, the system must look for enabled events in the left operand.

$$\begin{aligned}&\Pi(L,\beta,d_1 \ \mathsf{seq}\ d_2) \xrightarrow{e} \Pi(L,\beta,d_1' \ \mathsf{seq}\ d_2)\\ &\quad \textbf{if } ll.d_1 \cap L \neq \emptyset \wedge \Pi(ll.d_1 \cap L,\beta,d_1) \xrightarrow{e} \Pi(ll.d_1 \cap L,\beta,d_1')\end{aligned} \tag{9}$$

If the lifelines of the left hand side do not exhaust the given set of lifelines, this means there are lifelines that are only present on the right hand side, and that there may be enabled events on the right hand side of the operator. This means the system may look for enabled events at the right hand side of the seq, but only with respect to the lifelines not represented on the left hand side.

$$\begin{aligned}&\Pi(L,\beta,d_1 \ \mathsf{seq}\ d_2) \xrightarrow{e} \Pi(L,\beta,d_1 \ \mathsf{seq}\ d_2')\\ &\quad \textbf{if } L \setminus ll.d_1 \neq \emptyset \wedge \Pi(L \setminus ll.d_1,\beta,d_2) \xrightarrow{e} \Pi(L \setminus ll.d_1,\beta,d_2')\end{aligned} \tag{10}$$

Note that the two conditions $ll.d_1 \cap L \neq \emptyset$ and $ll.d_1 \setminus L \neq \emptyset$ are not mutually exclusive. If both these condition are true at the same time there may be enabled events at both sides of the seq operator. In such a case the rules may be applied in arbitrary order.

The transitions of the system are used as conditions in the recursion of these rules. Therefore the rules will not be applied unless an enabled event is found deeper in the recursion. Because of this the system will always be able to return an enabled event if enabled events exist.

4.2.4 Interleaving

The parallel operator par specifies interleaving of the events from each of its arguments; in other words parallel merge of the executions of each of the arguments. The rules of par are similar to the rules of seq, but simpler since we do not have to preserve any order between the two operands. One of the operands is chosen arbitrarily. As with the seq rules, the use of transitions as the conditions of the rules ensures that an enabled event is found if enabled events exist.

$$\begin{aligned}&\Pi(L,\beta,d_1 \ \mathsf{par}\ d_2) \xrightarrow{e} \Pi(L,\beta,d_1' \ \mathsf{par}\ d_2)\\ &\quad \textbf{if } \Pi(ll.d_1 \cap L,\beta,d_1) \xrightarrow{e} \Pi(ll.d_1 \cap L,\beta,d_1')\end{aligned} \tag{11}$$

$$\begin{aligned}&\Pi(L,\beta,d_1 \ \mathsf{par}\ d_2) \xrightarrow{e} \Pi(L,\beta,d_1 \ \mathsf{par}\ d_2')\\ &\quad \textbf{if } \Pi(ll.d_2 \cap L,\beta,d_2) \xrightarrow{e} \Pi(ll.d_2 \cap L,\beta,d_2')\end{aligned} \tag{12}$$

4.2.5 Choice

The rules for choices end the recursion; the choice is resolved and a silent event is produced. By resolving the choice instead of looking for events deeper down, we ensure that the same choice is made for all the lifelines covered by a choice operator.

$$\Pi(L,\beta,d_1 \;\mathsf{alt}\; d_2) \xrightarrow{\tau_{alt}} \Pi(L,\beta,d_k) \textbf{ if } L \cap ll.(d_1 \;\mathsf{alt}\; d_2) \neq \emptyset, \quad \textbf{for } \; k \in \{1,2\} \tag{13}$$

$$\Pi(L,\beta,d_1 \;\mathsf{xalt}\; d_2) \xrightarrow{\tau_{xalt}} \Pi(L,\beta,d_k) \textbf{ if } L \cap ll.(d_1 \;\mathsf{xalt}\; d_2) \neq \emptyset, \quad \textbf{for } \; k \in \{1,2\} \tag{14}$$

The rules for alt and xalt are identical except for the kind of silent event they produce. This reflects the fact that the operators are indistinguishable at the execution level. Since they produce different events, the kind of the choice is available at the meta-level and this is used in the definition of meta-strategies. As with rule (8), there is a condition that makes sure the rules are only applied if the event produced is relevant to the set L of lifelines.

4.2.6 Negative

The operator neg is treated as a choice with one negative branch and one empty branch. Silent events are used to flag which branch is chosen, and hence the choice is made available at the meta-level.

$$\Pi(L,\beta,\mathsf{neg}\; d) \xrightarrow{\tau_{pos}} \Pi(L,\beta,\mathsf{skip}) \textbf{ if } ll.(\mathsf{neg}\; d) \cap L \neq \emptyset \tag{15}$$

$$\Pi(L,\beta,\mathsf{neg}\; d) \xrightarrow{\tau_{neg}} \Pi(L,\beta,d) \textbf{ if } ll.(\mathsf{neg}\; d) \cap L \neq \emptyset \tag{16}$$

Similar to the choice rules, we have the condition that $ll.(\mathsf{neg}\; d) \cap L \neq \emptyset$ to ensure that the produced event is relevant to the set of lifelines L.

4.2.7 Iteration

Informally, in $\mathsf{loop}\; I\; d$ there is a non-deterministic choice between the numbers of I. If $n \in I$ is picked, d should be iterated n times. This is formalized by a rule that chooses which number to use:

$$\Pi(L,\beta,\mathsf{loop}\; I\; d) \xrightarrow{\tau_{alt}} \Pi(L,\beta,\mathsf{loop}\langle n\rangle\; d) \textbf{ if } n \in I \wedge ll.(\mathsf{loop}\; I\; d) \cap L \neq \emptyset \tag{17}$$

$\mathsf{loop}\langle n\rangle\; d$ is a loop with a counter. In the rule the counter is decreased by one for each iteration. We also produce a silent event to represent the iteration of a loop. Even though iteration of a loop in itself is not the most relevant information at the meta-level, it may be useful for defining execution strategies, for example if we want to give iteration of the loop low priority.

$$\Pi(L,\beta,\mathsf{loop}\langle n\rangle\; d) \xrightarrow{\tau_{loop}} \Pi(L,\beta,d \;\mathsf{seq}\; \mathsf{loop}\langle n-1\rangle\; d) \textbf{ if } ll.(\mathsf{loop}\langle n\rangle\; d) \cap L \neq \emptyset \tag{18}$$

Also here we have the condition that $ll.(\mathsf{loop}\langle n\rangle\; d) \cap L = ll.d \cap L \neq \emptyset$. Since we have that $\infty - 1 = \infty$, $\mathsf{loop}\langle\infty\rangle\; d$ specifies an infinite loop. Further we assert that $\mathsf{loop}\langle 0\rangle\; d$ is equal to skip, i.e., $\mathsf{loop}\langle 0\rangle\; d \overset{\mathsf{def}}{=} \mathsf{skip}$, so we do not need a special rule for this situation.

4.3 Fairness

With respect to diagrams that contain infinite loops, we must assume weak fairness between diagram parts for the operational semantics to be complete. This means that an arbitrary diagram part may not be enabled infinitely many consecutive execution steps without being executed. With this assumption we avoid situations where some part of a diagram is enabled and starved infinitely long at the same time. Below we formalize this notion of weak fairness.

4.3.1 Diagram projection part

We define diagram projection parts to be the parts (or fragments) of a diagram that may be reached by the recursion of the projection system. This means that any argument of seq or par are projection parts, while arguments of the high-level operators are not. In addition skip is a projection part of any diagram. For example in the diagram

$$d = d_1 \;\mathsf{seq}\; ((d_2 \;\mathsf{alt}\; d_3) \;\mathsf{par}\; d_4)$$

we have that skip, d_1, d_2 alt d_3 and d_4 are projection parts, while d_2 and d_3 are not.

The reason why the arguments of high-level operators are not projection parts is that events inside an argument of a high-level operator in a given state will not be reached by the projection system and therefore will not be executed. Hence, for characterizing what is executable in the state the events inside the arguments of high-level operators are irrelevant since it is the operator itself that will be executed. Because skip is the identity element of seq and par, skip will by definition be a projection part of any diagram, but we nevertheless formalize this explicitly in the definition.

We define a relation $\_ \lhd \_ \in \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{B}$ such that $d \lhd d'$ is read as "d is a projection part of d'." We let a relation $\_ \lhd_1 \_ \in \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{B}$ be defined as

$$\begin{aligned} d \lhd_1 d' \overset{\mathsf{def}}{=} \; & d = \mathsf{skip} \,\vee \\ & \exists d'' : d' = d \;\mathsf{seq}\; d'' \,\vee \\ & \qquad d' = d'' \;\mathsf{seq}\; d \,\vee \\ & \qquad d' = d \;\mathsf{par}\; d'' \,\vee \\ & \qquad d' = d'' \;\mathsf{par}\; d \end{aligned} \tag{19}$$

and let the relation $\lhd$ be the reflexive, transitive closure of $\lhd_1$.

4.3.2 Enabled and executed projection parts

A diagram projection part d is executed in a execution step $[\beta, d'] \xrightarrow{x} [\beta', d'']$, $x \in \mathcal{E} \cup \mathcal{T}$, if d is changed in the execution of the step (i.e. what is executed is d itself or a part of d). We formalize this by means of a relation *executed* $\in \mathcal{D} \times \mathcal{EX} \times (\mathcal{E} \cup \mathcal{T}) \times \mathcal{EX} \rightarrow \mathbb{B}$:

$$\begin{aligned} & \mathit{executed}(d, [\beta, d'], x, [\beta', d'']) \overset{\mathsf{def}}{=} \\ & \quad d \lhd d' \,\wedge \\ & \quad [\beta, d'] \xrightarrow{x} [\beta', d''] \,\wedge \\ & \quad \exists d''' : (\Pi(ll.d, \beta, d) \xrightarrow{x} \Pi(ll.d, \beta, d''') \wedge d''' \lhd d'') \end{aligned} \tag{20}$$

Obviously a diagram part d is enabled in $[\beta, d']$ if there exists an execution step starting in $[\beta, d']$ in which d is executable. This is formalized by the relation $\mathit{enabled} \in \mathcal{D} \times \mathcal{EX} \to \mathbb{B}$:

$$\mathit{enabled}(d, [\beta, d']) \stackrel{\mathsf{def}}{=} \exists x, \beta', d'' : \mathit{executed}(d, [\beta, d'], x, [\beta', d'']) \tag{21}$$

4.3.3 Weak fairness

We are now able to formalize the notion of weak fairness described above. An *execution* is a sequence of execution steps:

$$[\beta_1, d_1] \xrightarrow{x_1} [\beta_2, d_2] \xrightarrow{x_2} [\beta_3, d_3] \xrightarrow{x_3} \cdots$$

We define

$$\begin{aligned}\Xi \stackrel{\mathsf{def}}{=} \{[\beta_1, d_1] \xrightarrow{x_1} [\beta_2, d_2] \xrightarrow{x_2} [\beta_3, d_3] \xrightarrow{x_3} \cdots \in (\mathcal{EX} \times (\mathcal{E} \cup \mathcal{T}))^\infty \mid \\ \forall i \in \mathbb{N} : \mathit{executed}(d_i, [\beta_i, d_i], x_i, [\beta_{i+1}, d_{i+1}])\}\end{aligned}$$

to be the set of all infinite executions. We say that an execution is weakly fair if no diagram projection part that eventually becomes enabled, stays enabled forever, without being executed infinitely often.[6] This is formalized in the relation $\mathit{wfe} \in \Xi \to \mathbb{B}$:

$$\begin{aligned}&\mathit{wfe}([\beta_1, d_1] \xrightarrow{x_1} [\beta_2, d_2] \xrightarrow{x_2} [\beta_3, d_3] \xrightarrow{x_3} \cdots) \stackrel{\mathsf{def}}{=} \\ &\quad \forall d \in \mathcal{D}, i \in \mathbb{N} : (\forall j \in \mathbb{N} \cup \{0\} : \mathit{enabled}(d, [\beta_{i+j}, d_{i+j}]) \Rightarrow \\ &\qquad \exists k \in \mathbb{N} \cup \{0\} : \mathit{executed}(d, [\beta_{i+k}, d_{i+k}], x_{i+k}, [\beta_{i+k+1}, d_{i+k+1}]))\end{aligned} \tag{22}$$

An equivalent expression for *wfe*, that perhaps is closer to the above formulation of a weakly fair execution, is the following:

$$\begin{aligned}&\mathit{wfe}([\beta_1, d_1] \xrightarrow{x_1} [\beta_2, d_2] \xrightarrow{x_2} [\beta_3, d_3] \xrightarrow{x_3} \cdots) = \\ &\quad \neg\exists d \in \mathcal{D}, i \in \mathbb{N} : (\forall j \in \mathbb{N} \cup \{0\} : \mathit{enabled}(d, [\beta_{i+j}, d_{i+j}]) \wedge \\ &\qquad \neg\exists k \in \mathbb{N} \cup \{0\} : \mathit{executed}(d, [\beta_{i+k}, d_{i+k}], x_{i+k}, [\beta_{i+k+1}, d_{i+k+1}]))\end{aligned} \tag{23}$$

To ensure weak fairness of the operational semantics we place the following condition on executions:

$$\forall \sigma \in \Xi : \mathit{wfe}(\sigma) \tag{24}$$

By using the definition of weak fair executions we may express that a trace is weakly fair. Let $\mathit{tr} \in \Xi \to (\mathcal{E} \cup \mathcal{T})^\infty$ be a function that picks out all the events of an execution and returns a trace:

$$\mathit{tr}([\beta_1, d_1] \xrightarrow{x_1} [\beta_2, d_2] \xrightarrow{x_2} [\beta_3, d_3] \xrightarrow{x_3} \cdots) \stackrel{\mathsf{def}}{=} \langle x_1, x_2, x_3, \ldots\rangle$$

The relation $\mathit{wft} \in (\mathcal{E} \cup \mathcal{T}) \times \mathcal{D} \to \mathbb{B}$ formalize that a trace is weakly fair with respect to a diagram:

$$\mathit{wft}(t, d) \stackrel{\mathsf{def}}{=} \exists \sigma \in \Xi : \pi_2(\mathit{head}(\sigma)) = d \wedge \mathit{tr}(\sigma) = t \wedge \mathit{wfe}(\sigma) \tag{25}$$

The function *head* returns the first element of σ, which is an execution state, and π_2 returns the second element of an execution state, which is a diagram. The composition $\pi_2(\mathit{head}(\sigma))$ then gives the first diagram in execution σ. Hence, $\mathit{wft}(t, d)$ expresses that the trace t represents a weakly fair execution starting with the diagram d.

[6]An intuitive discussion of weak fairness can be found in [35].

4.4 Meta-strategies

There are several strategies we may choose when executing a sequence diagram and generating the histories of its possible executions. Examples of this may be generating one or a specific number of random traces, all traces, all prefixes of a certain length, etc. We wish to have the possibility of varying the execution strategy without changing the operational semantics of the sequence diagrams. The way to do this is to define different meta-strategies for executing the diagrams with the operational semantics. An example is given below where we show how we can produce a trace while capturing the high-level properties of the trace, e.g. the trace representing negative behavior.

We define a meta-system over

$$\{\!|\_,\_|\!\} \in \mathcal{H} \times \mathcal{EX} \times \mathcal{MO} \tag{26}$$

where $\mathcal{H}$ is the set of all traces, $\mathcal{EX}$ denotes the set of states of the execution system and $\mathcal{MO}$ is a set of modes. The first place of this tuple is a "container" for a trace, the second place holds the current state of the execution system and the third place represents the mode of the execution. Let the set of modes be defined as:

$$\mathcal{MO} \stackrel{\mathsf{def}}{=} \{\mathsf{postive}, \mathsf{negative}\}$$

The strategy may be defined by the means of two rules, one rule for normal events and one rule for silent events:

$$\{\!|t, V, mo|\!\} \longrightarrow \{\!|t^\frown\langle e\rangle, V', mo|\!\} \textbf{ if } V \xrightarrow{e} V' \wedge e \in \mathcal{E} \tag{27}$$

$$\{\!|t, V, mo|\!\} \longrightarrow \{\!|t, V', mode(\tau, mo)|\!\} \textbf{ if } V \xrightarrow{\tau} V' \wedge \tau \in \mathcal{T} \tag{28}$$

The function *mode* in the silent event rule is defined as:

$$mode(\tau, mo) \stackrel{\mathsf{def}}{=} \begin{cases} \mathsf{postitive} & \textbf{if } mo = \mathsf{positive} \wedge \tau \neq \tau_{refuse} \\ \mathsf{negative} & \textbf{if } mo = \mathsf{negative} \vee \tau = \tau_{refuse} \end{cases}$$

If we use these definitions, and we have $\mathsf{positive}$ as the initial mode an execution, the mode will represent the property of the produced trace of being positive or negative behavior. This means the initial state for execution of a sequence diagram d is:

$$\{\!|\langle\rangle, [\emptyset, d], \mathsf{positive}|\!\} \tag{29}$$

5 Soundness and completeness

The operational semantics is sound and complete with respect to the denotational semantics presented in [19–21]. Informally, the *soundness* property means that if the operational semantics produces a trace from a given diagram, this trace should be included in the denotational semantics of that diagram. By *completeness* we mean that all traces in the denotational semantics of a diagram should be producible applying the operational semantics on that diagram.

Let $\mathcal{O}$ be the set of all interaction obligations. $[\![\, d \,]\!] \in \mathbb{P}(\mathcal{O})$ is the denotation of d (the formal definition is found in appendix A). We write $t \in [\![\, d \,]\!]$ for $t \in \bigcup_{(p,n)\in[\![\, d \,]\!]}(p \cup n)$. $E \circledS t$ denotes the trace t with all events not in E filtered away. $env^{!}_{\mathcal{M}}.d$ is the multiset of messages m such that the receive event but not the transmit event of m is present in d.

Theorem (Termination) *Given a diagram $d \in \mathcal{D}$ without infinite loops. Then execution of $[env^!_{\mathcal{M}}.d, d]$ will terminate.*

The proof is found as proof of theorem 3 on page 64 in appendix B.3.

Theorem (Soundness) *Given a diagram $d \in \mathcal{D}$. For all $t \in (\mathcal{E} \cup \mathcal{T})^*$, if there exists $\beta \in \mathcal{B}$ such that $[env^!_{\mathcal{M}}.d, d] \xrightarrow{t} [\beta, \mathsf{skip}]$ then $\mathcal{E} \circledS t \in [\![d]\!]$.*

The soundness theorem is a combination of theorem 1 on page 50 in appendix B.2, that proves soundness of diagram with simple operators, theorem 4 on page 50 in appendix B.3, that proves soundness of diagrams with high-level operators, and theorem 6 on page 80 in appendix B.4, which proves soundness of diagrams with infinite loop.

Theorem (Completeness) *Given a diagram $d \in \mathcal{D}$. For all $t \in \mathcal{E}^*$, if $t \in [\![d]\!]$ then there exist trace $t' \in (\mathcal{E} \cup \mathcal{T})^*$ and $\beta \in \mathcal{B}$ such that $[env^!_{\mathcal{M}}.d, d] \xrightarrow{t'} [\beta, \mathsf{skip}]$* and *$\mathcal{E} \circledS t' = t$.*

This theorem a combination of theorem 2 on page 61 in appendix B.2, which proves completeness of diagrams with simple operators, theorem 5 on page 75 in appendix B.3, that proves completeness of diagrams with high-level operators and theorem 7 on page 86 in appendix B.4, where completeness of diagrams with infinite loop is proved.

6 Related work

In this section we present related work, which means other approaches to defining operational semantics to sequence diagrams. This presentation cannot, however, be seen independently of the history of sequence diagrams. Sequence diagrams as defined in the UML 2.0 standard is the last of a sequence of languages that have evolved over the last 15 to 20 years. The various approaches of defining semantics to sequence diagrams have emerged at different points in this history, and will clearly be influenced by the state of the language(s) at the time of their emergence.

An early version called Time Sequence Diagrams was standardized in the 1980s, see [11,24]. Better known are Message Sequence Charts (MSCs) that were first standardized by ITU in 1993 (see e.g. [4]). This standard is usually referred to as MSC-92, and describes what is now called basic MSCs. This means that MSC-92 did not have high-level constructs such as choice but merely consisted of lifelines and messages. MSC-92 had a lifeline-centric textual syntax,[7] and was given a semantics formalized in process algebra.

In 1996, a new MSC standard was defined, called MSC-96 [25]. In this standard, high-level constructs and high level MSCs were introduced, a kind of

[7]A lifeline-centric syntax means that each lifeline is characterized by itself and a diagram as collection of lifelines.

diagrams that show how control flows between basic MSCs. Further a event-centric textual syntax[8] and a new semantics were defined [26]. This semantics is also a kind of process algebra but, as will be explained, holds substantial differences from the MSC-92 semantics. Finally, the MSC-96 standard was revised in 1999 and became MSC-2000 [27], but kept the MSC-96 semantics.

The first versions of the Unified Modeling Language (UML 1.x) [43] included a version of sequence diagrams similar to MSC-92, i.e., consisting of only lifelines and messages and no higher level constructs. An important difference, however, was that the sequence diagrams of UML 1.x did not have the frame around the diagram, which in MSC-92 allowed messages to and from the environment of the specified system.

Sequence diagrams in UML 2.0 [44] may be seen as a successor of MSC-2000, since a lot of MSC language constructs have been incorporated in the UML 2.0 variant of sequence diagrams. This means that even though this report is concerned about UML sequence diagrams, much of the related work concerns MSCs. UML 2.0 sequence diagrams are, however, neither a subset nor a superset of MSC-2000; there are both similarities and differences between the languages [18]. Most notably (in our context) do MSCs not have any notion of negative behavior.

In addition to the STAIRS semantics there exist some other approaches to defining denotational semantics of UML 2.0 sequence diagrams [7, 33, 49, 50]. As explained in the introduction of section 4, denotational and operational semantics are complementary and serve different purposes. For this reason we do not go into detail on these approaches in this discussion, nor on existing approaches to denotational semantics of MSC like [30].

Several approaches to defining operational semantics to UML 2.0 sequence diagrams and MSCs exist. We do, however, find that none of these semantic definitions are satisfactory for our purposes.

The approach of Cengarle and Knapp [8] is similar to ours in that the operational semantics is defined as rules that produce events as a syntactic representation of the diagram is reduced. Contrary to ours, however, their semantics treats sequence diagrams as complete specifications (with no inconclusive behavior), something that does not conform to the UML standard (see section 2.2). The rules are defined such that a given diagram produces a set of valid and invalid traces that together exhaust the trace universe. The neg operator is replaced by an operator not. This operator is defined such that the sets of valid and invalid are swapped. This is unfortunate since specifying some behavior as invalid means also specifying the complement of this behavior as valid. We claim that this is not what you intuitively expect when specifying invalid behavior.

There are also some problems on the technical level. The semantics is based on a notion of composition of basic interactions where basic interactions are defined as partially ordered multisets (pomsets), but it is not clear how the pomsets are obtained; in other words it is not clear what the atoms of the compositions are. If the atoms are taken to be single events the reduction rules defining the seq operator do not preserve the message invariant (causality

[8]In an event-centric syntax events, as opposed to lifelines, are the basic building blocks of a diagram. The event-centric syntax of MSCs is more general than the lifeline centric-syntax in that all diagrams expressed in the lifeline-centric syntax can be expressed in the event-centric syntax, but not the other way around.

between transmission and reception of a message).

Cavarra and Klüster-Filipe [6] present an operational semantics of UML 2.0 sequence diagrams inspired by Live Sequence Charts (LSC) (see below). The semantics is formalized in pseudo-code that work on diagrams represented as locations in the diagram, but no translation from diagrams to this representation is provided. A more serious problem is the treatment of the choice operator alt. The operands of alts have guards and there is nothing to prevent the guards of more operands in an alt to evaluate to **true**. In this case the uppermost operand will be chosen, which means that the alts essentially are treated as nested **if-then-else** statements and may not be used for underspecification. Further, each lifeline is executed separately which means that synchronization at the entry of alt-fragments is necessary to ensure that all lifelines choose the same operand. They also make the same assumption about negative behavior as in LSCs, that if negative (neg) fragment is executed, then execution aborts (see section 2.3.4).

Harel and Maoz [15] use LSC semantics (see below) to define neg and assert. The operators are defined using already existing constructs of LSCs, and hence no changes or additions to the LSC semantics are needed in their approach. Because of this they also inherit the problems connected to LSCs.

In [13], Grosu and Smolka provide a semantics for UML 2.0 sequence diagrams based on translating the diagrams to Büchi automata. The approach is based on composing simple sequence diagrams (no high-level operators) in high-level sequence diagrams (interaction overview diagrams), where a simple diagram may be a positive or negative fragment of the high-level diagram it belongs to. Positive behavior is interpreted as liveness properties and negative behavior as safety properties. Hence, for a high-level diagram two Büchi automata is derived; a liveness automaton characterizing the positive behavior of the diagram and a safety automaton characterizing the negative behaviors. Compared to our operational semantics the approach is based on a large amount of transformation. Further the diagrams are composed by strict sequencing rather than weak sequencing, and hence has implicit synchronization of lifelines when entering or leaving a simple diagram.

In 1995 a formal algebraic semantics for MSC-92 was standardized by ITU [38,39]. MSC-92 has a lifeline-centric syntax and its semantics is based on characterizing each lifeline as a sequence (total order) of events. These sequences are composed in parallel and a set of algebraic rules transform the parallel composition into a structure of (strict) sequential composition and choice. The message invariant is obtained by a special function that removes from the structure all paths that violate the invariant. The main drawbacks of this semantics is that it is not a proper operational semantics since a diagram first has to be transformed into the event structure before runs can be obtained, and that this transformation replaces parallel composition with choice and hence creates an explosion in the size of the representation of the diagram. Further, the lifeline-centric syntax is not suitable for defining nested high-level constructs. In [45] a similar semantics for UML 1.x sequence diagram is given.

MSC-96 got a standardized process algebra semantics in 1998 [26, 40, 41]. This semantics is event-centric and has semantic operators for all the syntactic operators in MSC-96. Further these operators are "generalized" to preserve the message invariant by coding information about messages into the operators in the translation from syntactical diagrams to semantic expressions. Runs are characterized by inference rules over the semantic operators. Compared to our work, the most notable thing about this semantic is that is has no notion of negative behavior, and therefore also makes no distinction between negative behavior and inconclusive behavior (behavior that is neither positive nor negative). This is no surprise since MSC does not have the neg operator, but it is still a shortcoming with respect to UML sequence diagrams. The only available meta-level is a flat transition graph, and this does not give sufficient strength to extend the semantics with negative behavior. Neither is it possible to distinguish between potential and mandatory choice. Another shortcoming is the lack of an explicit communication medium; the communication model is "hard-coded" in the semantics by the "generalized operators" and does not allow for variation.

Another process algebra semantics of MSC is presented in [36]. This semantics may in some respects be seen as more general than the above semantics. A simple "core semantics" for MSCs is defined and this semantics is then inserted into an environment definition. Varying the definition of the environment allows for some of the same semantic variability and extendibility that we aim for in our semantics, e.g., with respect to the communication model. However, the semantics is heavily based on synchronization of lifelines on the entry of referenced diagrams and combined fragments, and diverges in this respect from the intended semantics of MSCs and UML sequence diagrams. Further, the same strategy as for the MSC-92 semantics is applied; interleaving is defined by means of choice, and the message invariant is obtained by removing deadlocks. In our opinion, this results in an unnecessary amount of computation, especially in the cases where we do not want to produce all traces but rather a selection of the traces that a diagram defines.

In [34] a semantics for MSC-92 is presented which is based on translating MSCs to finite automata with global states. This approach has no high-level operators, but conditions may be used for specifying choices between or iterations of diagrams. However, the insistence on translating the diagrams to *finite* automata makes it impossible to represent all reachable states because the combination of weak sequencing and loops may produce infinitely many states.

Realizability of MSCs is the focus of both [1,2] and [51]. They define synthesis of MSC to concurrent automata and parallel composition of labeled transition systems (LTS), respectively. (Each lifeline is represented as an automaton or LTS, which are then composed in parallel.) Further they define high-level MSCs as graphs where the nodes are basic MSCs. In addition, [51] defines both syntax and semantics for negative behavior. In both approaches the translation of high-level MSCs to concurrent automata/LTSs removes the semi-global nature of choices in a specification, and the high-level MSC-graphs are non-hierarchical, disallowing nesting of high level operators. Further, in [51] communication is synchronous.

Various attempts at defining Petri net semantics for MSCs have been made [12, 14, 22, 48]. In [12, 48] only basic MSCs are considered, and hence are of limited interest. In [14], high-level MSCs are defined as graphs where each node is a basic MSC. As with the above mentioned semantics it is then possible to express choices and loops, but the approach does not allow for nesting of high-level operators. In [22] a Petri net translation of the alternative operator alt is sketched, but no loop defined. In [3] a Petri-net semantics for UML 1.x sequence diagrams is presented, but as with the Petri-net semantics of basic MSCs it has big limitations and are of little interest.

Kosiuczenko and Wirsing [32] make a formalization of MSC-96 in Timed Maude, a variant of the term rewriting language Maude. Their semantics is problematic for two reasons: Every lifeline in a diagram is translated into an object specified in Maude, and the behavior of these objects are specified by the means of states and transition rules. We find this problematic because in general the states of a lifeline are implicit while this formalism assumes explicit states. Further, this way of reducing diagrams to sets of communicating objects has the effect that all choices are made locally in the objects and the choice operator alt looses its semi-global nature. Hence, this formalization does not capture the intended understanding of the alt operator.

An interesting approach is that of Jonsson and Padilla [29]. They present a semantics of MSC which is based on syntactical expansion and projection of diagram fragments during execution. Each lifeline is represented by a thread of labels where the labels refer to events or diagram fragments. The threads are executed in parallel and when a label referring to a fragment is reached the fragment is projected and expanded into the threads. Among the approaches referred to in this section, this is probably the approach that most resembles our operational semantics since their execution/expansion scheme resembles our execution/projection scheme. Because expansion produces silent events, the semantics can with small changes be extended with neg and xalt. Expansions may happen at arbitrary points since there are no rules in the semantics itself for when to expand. This creates a need for execution strategies, and the approach may be seen as having an informal meta-level where ad hoc strategies are described. However, if completeness is to be ensured, or if the semantics is to be extended with negative and potential/mandatory behavior this meta-level must be formalized. A problem with this semantics is that it requires explicit naming of all diagram fragments and this yields an unnecessary complicated syntax. Another shortcoming is the lack of a explicit communication medium; the communication model is "hard-coded" into the semantics and does not allow for variation.

Live Sequence Charts (LSC) [10,16,17] is a variant of MSC that does define a meta-level. Diagrams may be tagged as universal or existential and parts of diagrams as hot or cold, and this is evaluated at the meta-level. This yields something similar to the potential/mandatory distinction, and allows for specifying negative behavior. There is however a difference; LSCs specify what must or may happen given that some conditions are fulfilled, while our semantics distinguish between choices that must or may be present in an implementation. Further the semantics complies with neither the MSC nor the UML standard. Most importantly it requires synchronization between lifelines at every entry point of diagram fragments, e.g. when resolving a choice.

As we see, none of these approaches to defining operational semantics for UML sequence diagrams and MSCs fulfill the intentions and criteria we have set for our semantics. The shortcomings that dominate are:

- Non-conformance with the intended semantics of UML.
- No notion of explicit negative behavior and no distinction between negative behavior and inconclusive behavior (behavior that is neither positive nor negative).
- No distinction between potential and mandatory choice. (This is no sur-

prise as the alt/xalt distinction is an invention of STAIRS.)

- Lack of a proper meta-level that may be used for assigning meaning to negative and potential/mandatory behavior.
- Lack of possibility and freedom in defining and formalizing a meta-level.
- Lack of modifiability and extensibility, e.g., with respect to the communication model.
- Requiring transformations from the textual syntax into the formalism of the approach.

Our aim has been to stay close to the UML standard in both syntax and semantics. Further we have aimed to facilitate ease of extension and modification when adapting the semantics to different interpretations and applications of sequence diagrams.

7 Conclusions

In this report we have presented an operational semantics for UML 2.0 sequence diagrams. We are not aware of any other operational semantics for UML 2.0 sequence diagrams or MSCs with the same strength and generality as ours. Several other approaches have been made, but all with significant shortcomings.

Our operational semantics for UML 2.0 sequence diagrams is simple and is defined with extensibility and variation in mind. It does not involve any translation or transformation of the diagrams into other formalisms, which makes it easy to use and understand. It is sound and complete with respect to a reasonable denotational formalization of the UML standard.

We have shown how the operational semantics can be given a formalized meta-level for defining execution strategies. The meta-level given in this report is used for distinguishing valid from invalid traces, but also other meta-levels with other purposes can be defined. Most notably we can define a meta-level for distinguishing between traces of different interaction obligations. It is also possible to define meta-levels that take a black box view of diagrams. Further a meta-level may be used for defining different meta-strategies that guide the execution, such as generating a specific number of traces or prefixes of a specific length.

The semantics is implemented in the term rewriting language Maude [9], and forms the basis of a tool for analysis of sequence diagrams. The operational semantics has also been used to define test generation from sequence diagrams; see [37] for more details.

Acknowledgments

The work of this report was conducted within and funded by the Basic ICT Research project SARDAS (15295/431) under the Research Council of Norway. We would like to thank Rolv Bræk, Manfred Broy, Kathrin Greiner, Øystein Haugen, Birger Møller-Pedersen, Atle Refsdal, Ragnhild Kobro Runde, Ina Schieferdecker and Thomas Weigert for useful comments on this work.

References

[1] R. Alur, K. Etessami, and M. Yannakakis. Inference of Message Sequence Charts. *IEEE Transactions on Software Engineering*, 29(7):623–633, 2003.

[2] R. Alur, K. Etessami, and M. Yannakakis. Realizability and verification of MSC graphs. *Theoretical Computer Science*, 331(1):97–114, 2005.

[3] S. Bernardi, S. Donatelli, and J. Merseguer. From UML sequence diagrams and statecharts to analysable Petri net models. In *3rd International Workshop on Software and Performance (WOSP'02)*, pages 35–45. ACM Press, 2002.

[4] R. Bræk, J. Gorman, Ø. Haugen, B. Møller-Pedersen, G. Melby, R. Sanders, and T. Stålhane. *TIMe: The Integrated Method. Electronic Textbook v4.0.* SINTEF, 1999.

[5] R. Bruni and J. Meseguer. Generalized rewrite theories. In *30th International Colloquium on Automata, Languages and Programming (ICALP'03)*, number 2719 in Lecture Notes in Computer Science, pages 252–266. Springer, 2003.

[6] A. Cavarra and J. Küster-Filipe. Formalizing liveness-enriched sequence diagrams using ASMs. In *11th International Workshop on Abstract State Machines: Advances in Theory and Practice (ASM'04)*, number 3052 in Lecture Notes in Computer Science, pages 67–77. Springer, 2004.

[7] M. V. Cengarle and A. Knapp. UML 2.0 interactions: Semantics and refinement. In *3rd International Workshop on Critical Systems Development with UML (CSD-UML'04)*, pages 85–99. Technische Universität München, 2004.

[8] M. V. Cengarle and A. Knapp. Operational semantics of UML 2.0 interactions. Technical report TUM-I0505, Technische Universität München, 2005.

[9] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *Maude Manual (Version 2.2).* SRI International, Menlo Park, 2005.

[10] W. Damm and D. Harel. LSCs: Breathing life into Message Sequence Charts. *Formal Methods in System Design*, 19:45–80, 2001.

[11] C. Facchi. Formal semantics of Time Sequence Diagrams. Technical report TUM-I9540, Technische Universität München, 1995.

[12] P. Graubmann, E. Rudolph, and J. Grabowski. Towards a petri net based semantics for Message Sequence Charts. In *6th International SDL Forum: Using objects (SDL'93)*, pages 179–190. Elsevier, 1993.

[13] R. Grosu and S. A. Smolka. Safety-liveness semantics for UML 2.0 sequence diagrams. In *5th International Conference on Application of Concurrency to System Design (ACSD'05)*, pages 6–14. IEEE Computer Society, 2005.

[14] E. L. Gunter, A. Muscholl, and D. Peled. Compositional Message Sequence Charts. *International Journal on Software Tools for Technology Transfer*, 5(1):78–89, 2003.

[15] D. Harel and S. Maoz. Assert and negate revisited: Modal semantics for UML sequence diagrams. In *5th International Workshop on Scenarios and State Machines: Models, Algorithms, and Tools (SCESM'06)*, pages 13–19. ACM Press, 2006.

[16] D. Harel and R. Marelly. *Come, let's play: Scenario-based programming using LSCs and the Play-Engine*. Springer, 2003.

[17] D. Harel and P. S. Thiagarajan. Message Sequence Charts. In *UML for Real. Design of Embedded Real-Time Systems*. Kluwer, 2003.

[18] Ø. Haugen. Comparing UML 2.0 Interactions and MSC-2000. In *4th International SDL and MSC Workshop: System Analysis and Modeling (SAM '04)*, number 3319 in Lecture Notes in Computer Science, pages 65–79. Springer, 2004.

[19] Ø. Haugen, K. E. Husa, R. K. Runde, and K. Stølen. Why timed sequence diagrams require three-event semantics. Research report 309, Department of Informatics, University of Oslo, 2004. Revised June 2005.

[20] Ø. Haugen, K. E. Husa, R. K. Runde, and K. Stølen. STAIRS towards formal design with sequence diagrams. *Software and Systems Modeling*, 4(4):355–367, 2005.

[21] Ø. Haugen, K. E. Husa, R. K. Runde, and K. Stølen. Why timed sequence diagrams require three-event semantics. In *Scenarios: Models, transformations and tools. International Workshop, Dagstuhl Castle, Germany, September 2003. Revised selected papers*, number 3466 in Lecture Notes in Computer Science, pages 1–25. Springer, 2005.

[22] S. Heymer. A semantics for MSC based on petri net components. In *4th International SDL and MSC Workshop (SAM'00)*, pages 262–275, 2000.

[23] C. A. R. Hoare and He Jifeng. *Unifying theories of programming*. Prentice Hall, 1998.

[24] International Telecommunication Union. *Information technology – Open Systems Interconnection – Basic reference model: Conventions for the definition of OSI services, ITU-T Recommendation X.210*, 1993.

[25] International Telecommunication Union. *Message Sequence Chart (MSC), ITU-T Recommendation Z.120*, 1996.

[26] International Telecommunication Union. *Message Sequence Chart (MSC), ITU-T Recommendation Z.120, Annex B: Formal semantics of Message Sequence Charts*, 1998.

[27] International Telecommunication Union. *Message Sequence Chart (MSC), ITU-T Recommendation Z.120*, 1999.

[28] J. Jacob. On the derivation of secure components. In *IEEE Symposium on Security and Privacy*, pages 242–247, 1989.

[29] B. Jonsson and G. Padilla. An execution semantics for MSC-2000. In *10th International SDL Forum: Meeting UML (SDL'01)*, number 2078 in Lecture Notes in Computer Science, pages 365–378. Springer, 2001.

[30] J.-P. Katoen and L. Lambert. Pomsets for Message Sequence Charts. In *Formale Beschreibungstechniken für Verteilte Systeme*, pages 197–208. Shaker, 1998.

[31] E. Kindler and A. Martens. Cross-talk revisited: What's the problem? *Petri Net Newsletter*, 58:4–10, 2000.

[32] P. Kosiuczenko and M. Wirsing. Towards an integration of Message Sequence Charts and Timed Maude. *Journal of Integrated Design & Process Science*, 5(1):23–44, 2001.

[33] J. Küster-Filipe. Modelling concurrent interactions. In *10th International Conference on Algebraic Methodology and Software Technology (AMAST'04)*, number 3116 in Lecture Notes in Computer Science, pages 304–318. Springer, 2004.

[34] P. B. Ladkin and S. Leue. What do Message Sequence Charts mean? In *6th International Conference on Formal Description Techniques (FORTE'93), Formal Description Techniques VI*, pages 301–316. Elsevier, 1994.

[35] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, 1994.

[36] A.A. Letichevsky, J.V. Kapitonova, V.P. Kotlyarov, V.A. Volkov, A. A. Letichevsky, and T. Weigert. Semantics of Message Sequence Charts. In *12th International SDL Forum: Model Driven Systems Design (SDL'05)*, number 3530 in Lecture Notes in Computer Science, pages 117–132. Springer, 2005.

[37] M. S. Lund and K. Stølen. Deriving tests from UML 2.0 sequence diagrams with neg and assert. In *1st International Workshop on Automation of Software Test (AST'06)*, pages 22–28. ACM Press, 2006.

[38] S. Mauw. The formalization of Message Sequence Charts. *Computer Networks and ISDN Systems*, 28(1):1643–1657, 1996.

[39] S. Mauw and M. A. Reniers. An algebraic semantics of Basic Message Sequence Charts. *The Computer Journal*, 37(4):269–278, 1994.

[40] S. Mauw and M. A. Reniers. High-level Message Sequence Charts. In *8th International SDL Forum: Time for Testing, SDL, MSC and Trends (SDL'97)*, pages 291–306. Elsevier, 1997.

[41] S. Mauw and M. A. Reniers. Operational semantics for MSC'96. *Computer Networks*, 31(17):1785–1799, 1999.

[42] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96:73–155, 1992.

[43] Object Management Group. *Unified Modeling Language Specification, version 1.4*, 2001.

[44] Object Management Group. *Unified Modeling Language: Superstructure, version 2.0*, 2005. OMG Document: formal/2005-07-04.

[45] M. Okazaki, T. Aoki, and T. Katayama. Formalizing sequence diagrams and state machines using Concurrent Regular Expression. In *2nd International Workshop on Scenarios and State Machines: Models, Algorithms, and Tools (SCESM'03)*, 2003.

[46] R. K. Runde, Ø. Haugen, and K. Stølen. How to transform UML neg into a useful construct. In *Norsk Informatikkonferanse (NIK'05)*, pages 55–66. Tapir, 2005.

[47] D. A. Schmidt. *Denotational Semantics. A Methodology for Language Development.* wcb, 1988.

[48] M. Sgroi, A. Kondratyev, Y. Watanabe, L. Lavagno, and A. Sangiovanni-Vincentelli. Synthesis of petri nets from Message Sequence Charts specifications for protocol design. In *Design, Analysis and Simulation of Distributed Systems Symposium (DASD'04)*, pages 193–199, 2004.

[49] H. Störrle. Assert, negate and refinement in UML 2 interactions. In *2nd International Workshop on Critical Systems Development with UML (CSD-UML'03)*, pages 79–93. Technische Universität München, 2003.

[50] H. Störrle. Semantics of interaction in UML 2.0. In *IEEE Symposium on Human Centric Computing Languages and Environments (HCC'03)*, pages 129–136. IEEE Computer Society, 2003.

[51] S. Uchitel, J. Kramer, and J. Magee. Incremental elaboration of scenario-based specification and behavior models using implied scenarios. *ACM Transactions on Software Engineering and Methodology*, 13(1):37–85, 2004.

A Denotational semantics

In this section we give the denotational semantics of STAIRS as presented in [19–21], but reduced to an untimed two event version.

A.1 Assumptions

On diagrams we have the constraints that a given message should syntactically occur only once, and if both the transmitter and the receiver lifelines of the message are present in the diagram, then both the transmit event and receive event of that message must be in the diagram. Further if both the transmit event and receive event of a message are present in a diagram, they have to be inside the same argument of the same high-level operator (alt, xalt, neg or loop). A way of phrasing the latter is that in the graphical notation, messages are not allowed to cross the frame of a high-level operator or the dividing line between the arguments of a high-level operator.

STAIRS is a trace-based semantics. A trace is a sequence of events, used to represent a system run. In each trace, an transmit event should always be ordered before the corresponding receive event. We let $\mathcal{H}$ denote the set of all traces that complies with this requirement. Formally this is handled by a constraint on traces.

We use | and # to denote, respectively, truncation and length of traces. For concatenation of traces, filtering of traces, and filtering of pairs of traces, we have the functions $\frown$, Ⓢ, and Ⓣ, respectively. Concatenating two traces implies gluing them together. Hence, $t_1 \frown t_2$ denotes a trace that equals t_1 if t_1 is infinite. Otherwise, $t_1 \frown t_2$ denotes the trace directly followed by t_1 and suffixed by t_2. The filtering function Ⓢ is used to filter away elements. By $A Ⓢ t$ we denote the trace obtained from the trace t by removing all elements in t that are not in the set of elements A. For example, we have that

$$\{1, 3\} Ⓢ \langle 1, 1, 2, 1, 3, 2\rangle = \langle 1, 1, 1, 3\rangle$$

The function Ⓣ filters pairs of traces with respect to pairs of elements in the same way as Ⓢ filters traces with respect to elements. For any set of pairs of elements P and pair of traces u, by $P Ⓣ u$ we denote the pair of traces obtained from u by 1) truncating the longest trace in u at the length of the shortest traces in u if the two sequences are of unequal length; 2) for each $j \in [1 \ldots k]$, where k is the length of the shortest trace in u, selecting or deleting the two elements at index j in the two trace, depending on whether the pair of these elements is in the set P. For example, we have that

$$\{(1, f), (1, g)\} Ⓣ (\langle 1, 1, 2, 1, 2\rangle, \langle f, f, f, g, g\rangle) = (\langle 1, 1, 1\rangle, \langle f, f, g\rangle)$$

The semantic constraint on traces may now be formalized. It asserts that for all traces $h \in \mathcal{H}$, if at a point in a trace we have an receive event of a message, then up to that point we must have has at least as many transmits of that message as receives.

$$\forall i \in [1..\#h] : k.h[i] = ! \Rightarrow \#(\{(!, m.h[i])\} Ⓢ h|_i) > \#(\{(?, m.h[i])\} Ⓢ h|_i) \quad (30)$$

A.2 Semantics

As explained in section 2.2, the semantic model of STAIRS is a set of interaction obligations, where an interaction obligation is a pair (p, n) of sets of traces where the first set is interpreted as the set of positive traces and the second set is the set of negative traces. In the following we let $\mathcal{O}$ be the set of all interaction obligations. The semantics of sequence diagrams is defined by a function

$$[\![\,\_\,]\!] \in \mathcal{D} \rightarrow \mathbb{P}(\mathcal{O}) \tag{31}$$

that for any sequence diagram d yields a set $[\![\, d \,]\!]$ of interaction obligations.

The semantics of the empty sequence diagram skip is defined as only the empty positive trace

$$[\![\, \mathsf{skip} \,]\!] \overset{\mathsf{def}}{=} \{(\{\langle\rangle\}, \emptyset)\} \tag{32}$$

For a sequence diagram consisting of a single event e, its semantics is given by:

$$[\![\, e \,]\!] \overset{\mathsf{def}}{=} \{(\{\langle e \rangle\}, \emptyset)\} \tag{33}$$

A.2.1 Weak sequencing

Weak sequencing is the implicit composition mechanism combining constructs of a sequence diagram. First, we define weak sequencing of trace sets

$$s_1 \precsim s_2 \overset{\mathsf{def}}{=} \{h \in \mathcal{H} \mid \exists h_1 \in s_1, h_2 \in s_2 : \forall l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2\} \tag{34}$$

where $e.l$ denotes the set of events that may take place on the lifeline l:

$$e.l \overset{\mathsf{def}}{=} \{e \in \mathcal{E} \mid l.e = l\}$$

The seq construct itself is defined as:

$$[\![\, d_1 \,\mathsf{seq}\, d_2 \,]\!] \overset{\mathsf{def}}{=} \{o_1 \precsim o_2 \mid o_1 \in [\![\, d_1 \,]\!] \wedge o_2 \in [\![\, d_2 \,]\!]\} \tag{35}$$

where weak sequencing of interaction obligations is defined as:

$$(p_1, n_1) \precsim (p_2, n_2) \overset{\mathsf{def}}{=} (p_1 \precsim p_2, (n_1 \precsim p_2) \cup (n_1 \precsim n_2) \cup (p_1 \precsim n_2)) \tag{36}$$

A.2.2 Interleaving

The par construct represents a parallel merge or a permutation of traces such that the operands remain subsequences of the resulting traces. In order to define par, we first define parallel execution on trace sets:

$$\begin{aligned} s_1 \parallel s_2 \overset{\mathsf{def}}{=} \quad & \{h \in \mathcal{H} \mid \exists p \in \{1,2\}^\infty : \\ & \quad \pi_2((\{1\} \times \mathcal{E}) \circledT (p, h)) \in s_1 \wedge \\ & \quad \pi_2((\{2\} \times \mathcal{E}) \circledT (p, h)) \in s_2\} \end{aligned} \tag{37}$$

In this definition, we make use of an oracle, the infinite sequence p, to resolve the non-determinism in the interleaving. It determines the order in which events from traces in s_1 and s_2 are sequenced. π_2 is a projection operator returning the second element of a pair. Formally:

$$\pi_i(s_1, s_2) \overset{\mathsf{def}}{=} s_i, \;\; i \in \{1, 2\}$$

The par construct itself is defined as

$$[\![\, d_1 \,\mathsf{par}\, d_2 \,]\!] \overset{\mathsf{def}}{=} \{o_1 \parallel o_2 \mid o_1 \in [\![\, d_1 \,]\!] \wedge o_2 \in [\![\, d_2 \,]\!]\} \tag{38}$$

where parallel execution of interaction obligations is defined as:

$$(p_1, n_1) \parallel (p_2, n_2) \overset{\mathsf{def}}{=} (p_1 \parallel p_2, (n_1 \parallel p_2) \cup (n_1 \parallel n_2) \cup (p_1 \parallel n_2)) \tag{39}$$

A.2.3 Choice

The alt construct defines potential traces. The semantics is the inner union of the interaction obligations:

$$[\![\, d_1 \,\mathsf{alt}\, d_2 \,]\!] \overset{\mathsf{def}}{=} \{(p_1 \cup p_2, n_1 \cup n_2) \mid (p_1, n_1) \in [\![\, d_1 \,]\!] \wedge (p_2, n_2) \in [\![\, d_2 \,]\!]\} \tag{40}$$

The xalt construct defines mandatory choices. All implementations must be able to handle every interaction obligation.

$$[\![\, d_1 \,\mathsf{xalt}\, d_2 \,]\!] \overset{\mathsf{def}}{=} [\![\, d_1 \,]\!] \cup [\![\, d_2 \,]\!] \tag{41}$$

A.2.4 Negative

The neg construct defines negative traces:

$$[\![\, \mathsf{neg}\, d \,]\!] \overset{\mathsf{def}}{=} \{(\{\langle\rangle\}, p \cup n) \mid (p, n) \in [\![\, d \,]\!]\} \tag{42}$$

A.2.5 Iteration

The semantics of loop is defined by a semantic loop construct μ_n, where n is the number of times the loop should be iterated. Let $\biguplus$ be a generalization of potential choice (inner union of interaction obligations) such that

$$\biguplus_{i \in I} O_i \overset{\mathsf{def}}{=} \{\Big(\bigcup_{i \in I} p_i, \bigcup_{i \in I} n_i\Big) \mid \forall i \in I : (p_i, n_i) \in O_i\} \tag{43}$$

where O_i are sets of interaction obligations. loop is then defined as:

$$[\![\, \mathsf{loop}\ I\ d \,]\!] \overset{\mathsf{def}}{=} \biguplus_{i \in I} \mu_i\ [\![\, d \,]\!] \tag{44}$$

For $n \in \mathbb{N}$ (finite loop), μ_n is defined as:

$$\begin{aligned} \mu_0\ O &\overset{\mathsf{def}}{=} \{(\{\langle\rangle\}, \emptyset)\} \\ \mu_1\ O &\overset{\mathsf{def}}{=} O \\ \mu_n\ O &\overset{\mathsf{def}}{=} O \succsim \mu_{n-1}\ O \quad \textbf{if } n > 1 \end{aligned} \tag{45}$$

In order to define infinite loop we need some auxiliary definitions. We define the chains of interaction obligation as:

$$\begin{aligned} \mathit{chains}(O) \overset{\mathsf{def}}{=} \{\bar{o} \in \mathcal{O}^{\infty} \mid \bar{o}[1] \in O \wedge \\ \forall j \in \mathbb{N} : \exists o \in O : \bar{o}[j+1] = \bar{o}[j] \succsim o\} \end{aligned} \tag{46}$$

From a chain of interaction obligations we define its chains of positive and negative traces:

$$pos(\bar{o}) \overset{\mathsf{def}}{=} \{\bar{t} \in \mathcal{H}^\infty \mid \forall j \in \mathbb{N} : \bar{t}[j] \in \pi_1(\bar{o}[j]) \wedge \\ \exists t \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \succsim \{t\}\} \tag{47}$$

$$negs(\bar{o}) \overset{\mathsf{def}}{=} \{\bar{t} \in \mathcal{H}^\infty \mid \exists i \in \mathbb{N} : \forall j \in \mathbb{N} : \bar{t}[j] \in \pi_2(\bar{o}[j+i-1]) \wedge \\ \exists t \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \succsim \{t\}\} \tag{48}$$

For every chain of traces $\bar{t}$ we have that for all $l \in \mathcal{L}$, the sequence

$$e.l \circledS \bar{t}[1], e.l \circledS \bar{t}[2], e.l \circledS \bar{t}[3], \ldots$$

is a chain ordered by $\sqsubseteq$. We let $\sqcup_l \bar{t}$ denote the least upper bound of this chain and define the approximations of $\bar{t}$ as:

$$\sqcup \bar{t} \overset{\mathsf{def}}{=} \{h \in \mathcal{H} \mid \forall l \in \mathcal{L} : e.l \circledS h = \sqcup_l \bar{t}\} \tag{49}$$

For a chain of interaction obligations we define:

$$\sqcup \bar{o} \overset{\mathsf{def}}{=} \left(\bigcup_{\bar{t} \in pos(\bar{o})} \sqcup \bar{t}, \bigcup_{\bar{t} \in negs(\bar{o})} \sqcup \bar{t} \right) \tag{50}$$

and let infinite loop be defined as:

$$\mu_\infty\, O \quad \overset{\mathsf{def}}{=} \quad \{\sqcup \bar{o} \mid \bar{o} \in chains(O)\} \tag{51}$$

B Proofs of soundness, completeness and termination of the operational semantics

In this appendix we provide proofs of the soundness and completeness of the operational semantics with respect to the denotational semantics presented in [20, 21]. We also prove termination of execution of diagrams without infinite loops.

Informally, the *soundness* property means that if the operational semantics produces a trace from a given diagram, this trace should be included in the denotational semantics of that diagram. By *completeness* we mean that all traces in the denotational semantics of a diagram should be producible applying the operational semantics on that diagram.

The proofs are based on interpreting the rules of the operational semantics as a rewrite theory. For a formal treatment of rewrite theories, see [5, 42].

In section B.1 we provide some basic definitions that will be applied in the following proofs. In section B.2 we prove soundness and completeness of simple diagrams (theorem 1 on page 50 and theorem 2 on page 61). In section B.3 we prove termination (theorem 3 on page 64), soundness (theorem 4 on page 73) and completeness (theorem 5 on page 75) of diagrams with high-level operators but without infinite loops. In section B.4 we prove soundness (theorem 6 on page 80) and completeness (theorem 7 on 86) of diagrams with infinite loops.

B.1 Definitions

Definition 1 *We define a function $msg.\_ \in \mathcal{D} \rightarrow \mathbb{P}(\mathcal{M})$ that returns the messages of a diagram:*

$$\begin{array}{llllll}
msg.\mathsf{skip} & \overset{\mathsf{def}}{=} & \emptyset & msg.(d_1 \;\mathsf{seq}\; d_2) & \overset{\mathsf{def}}{=} & msg.d_1 \cup msg.d_2 \\
msg.e & \overset{\mathsf{def}}{=} & \{m.e\} & msg.(d_1 \;\mathsf{par}\; d_2) & \overset{\mathsf{def}}{=} & msg.d_1 \cup msg.d_2 \\
msg.(\mathsf{neg}\; d) & \overset{\mathsf{def}}{=} & msg.d & msg.(d_1 \;\mathsf{alt}\; d_2) & \overset{\mathsf{def}}{=} & msg.d_1 \cup msg.d_2 \\
msg.(\mathsf{loop}\; I\; d) & \overset{\mathsf{def}}{=} & msg.d & msg.(d_1 \;\mathsf{xalt}\; d_2) & \overset{\mathsf{def}}{=} & msg.d_1 \cup msg.d_2
\end{array}$$

Definition 2 *We define a function $ev.\_ \in \mathcal{D} \rightarrow \mathbb{P}(\mathcal{E})$ that returns the events of a diagram:*

$$\begin{array}{llllll}
ev.\mathsf{skip} & \overset{\mathsf{def}}{=} & \emptyset & ev.(d_1 \;\mathsf{seq}\; d_2) & \overset{\mathsf{def}}{=} & ev.d_1 \cup ev.d_2 \\
ev.e & \overset{\mathsf{def}}{=} & \{e\} & ev.(d_1 \;\mathsf{par}\; d_2) & \overset{\mathsf{def}}{=} & ev.d_1 \cup ev.d_2 \\
ev.(\mathsf{neg}\; d) & \overset{\mathsf{def}}{=} & ev.d & ev.(d_1 \;\mathsf{alt}\; d_2) & \overset{\mathsf{def}}{=} & ev.d_1 \cup ev.d_2 \\
ev.(\mathsf{loop}\; I\; d) & \overset{\mathsf{def}}{=} & ev.d & ev.(d_1 \;\mathsf{xalt}\; d_2) & \overset{\mathsf{def}}{=} & ev.d_1 \cup ev.d_2
\end{array}$$

Definition 3 (Communication medium) *In the following we use the powerset over the set of messages as the set of all states of the communication medium:*

$$\mathcal{B} \overset{\mathsf{def}}{=} \mathbb{P}(\mathcal{M})$$

Further we define the operators on the medium states:

- $add(\beta, m) \overset{\mathsf{def}}{=} \beta \cup \{m\}$
- $rm(\beta, m) \overset{\mathsf{def}}{=} \beta \setminus \{m\}$

- $ready(\beta, m) \overset{\mathsf{def}}{=} m \in \beta$

Definition 4 *We use the following shorthand notation:*

- *If there exists a sequence of rewrites*

$$[\beta, d] \xrightarrow{e_1} [\beta_1, d_1] \xrightarrow{e_2} \cdots \xrightarrow{e_n} [\beta_n, d_n]$$

we will write

$$[\beta, d] \xrightarrow{\langle e_1, e_2, \ldots, e_n \rangle} [\beta_n, d_n]$$

- *We say that an event e is* enabled *in execution state $[\beta, d]$ iff there exists a rewrite*

$$[\beta, d] \xrightarrow{e} [\beta', d']$$

for some β', d'.

- *We write*

$$[\beta, d] \xrightarrow{e}$$

for

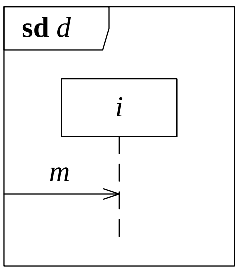
$$\exists \beta', d' : [\beta, d] \xrightarrow{e} [\beta', d']$$

- *If an event e is not enabled in $[\beta, d]$ we write*

$$[\beta, d] \overset{e}{\not\rightarrow}$$

- *If for all events e we have that e is not enabled in $[\beta, d]$ (i.e. no events are enabled in $[\beta, d]$) we write*

$$[\beta, d] \not\rightarrow$$



$$d = (?, (m, j, i))$$
$$[\![d]\!] = \{\langle (?, (m, j, i)) \rangle\}$$

Figure 2: Diagram with single input

The denotational semantics has an implicit notion of environment, which allows for, e.g., the diagram in figure 2 which consist of only one input event as long as the lifeline of the corresponding output event does not occur in the diagram. In the operational semantics, the state of the communication medium β in an execution state $[\beta, d]$ may be seen as an explicit notion of environment. Given the diagram d in figure 2, the operational semantics produces the trace of $[\![d]\!]$ only if the message m is available in the state of the communication medium, i.e.:

$$(m, j, i) \in \beta \Rightarrow [\beta, d] \xrightarrow{(?, (m, j, i))} [\beta \setminus \{(m, j, i)\}, \mathsf{skip}]$$

$$(m, j, i) \notin \beta \Rightarrow [\beta, d] \not\twoheadrightarrow$$

In order to align the two semantics we must make explicit the environment of the denotational semantics.

Definition 5 (Environment) *The function* $env.\_ \in \mathcal{D} \rightarrow \mathbb{P}(\mathcal{E})$ *returns the environment of a diagram* d

$$env.d \overset{\mathsf{def}}{=} \{(k, m) | k \in \{!, ?\} \wedge m \in msg.d \wedge (k, m) \notin ev.d\}$$

Further we define

$$\begin{aligned} env_{\mathcal{M}}.d &\overset{\mathsf{def}}{=} \{m.e | e \in env.d\} \\ env^{k}.d &\overset{\mathsf{def}}{=} \{(k, m) | (k, m) \in env.d\} \\ env^{k}_{\mathcal{M}}.d &\overset{\mathsf{def}}{=} \{m | (k, m) \in env.d\} \end{aligned}$$

B.2 Simple sequence diagrams

In this section we prove soundness and completeness of what we refer to as *simple diagrams*; diagram without high-level constructs such as loop and choice. The general case is addressed in section B.3. In section B.3 we also prove termination.

Definition 6 (Simple diagram) *A diagram is called simple iff it only contains*

- *events*
- skip
- *the operators* seq *and* par

In this section we are only concerned with simple diagrams. We also restrict ourselves to diagrams where no event occurs more than once:

$$\begin{aligned} d = d_1 \ \mathsf{seq}\ d_2 &\quad\Rightarrow\quad ev.d_1 \cap ev.d_2 = \emptyset \\ d = d_1 \ \mathsf{par}\ d_2 &\quad\Rightarrow\quad ev.d_1 \cap ev.d_2 = \emptyset \end{aligned}$$

It follows from this condition that no message occur in more than one transmit and one receive event.

Lemma 1 *If* d *is a simple diagram, then*

$$[\![d]\!] = \{(T, \emptyset)\} \quad \text{with} \quad T \subseteq \mathcal{H}$$

Proof of lemma 1
ASSUME: d simple
PROVE: $[\![d]\!] = \{(T, \emptyset)\} \wedge T \subseteq \mathcal{H}$
PROOF: by induction on the structure of d
⟨1⟩1. CASE: $d = \mathsf{skip}$ (induction step)
  ⟨2⟩1. $[\![d]\!] = \{(\{\langle\rangle\}, \emptyset)\}$
    PROOF: Def. 32.
  ⟨2⟩2. $\{\langle\rangle\} \subseteq \mathcal{H}$
    PROOF: Def. of $\mathcal{H}$ and (30)

$\langle 2 \rangle 3.$ Q.E.D.

PROOF: $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$

$\langle 1 \rangle 2.$ CASE: $d = e$ (induction step)

$\langle 2 \rangle 1.$ $\llbracket e \rrbracket = \{(\{\langle e \rangle\}, \emptyset)\}$

PROOF: Def. 33.

$\langle 2 \rangle 2.$ $\{\langle e \rangle\} \subseteq \mathcal{H}$

PROOF: Def. of $\mathcal{H}$ and (30).

$\langle 2 \rangle 3.$ Q.E.D.

PROOF: $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$.

$\langle 1 \rangle 3.$ $d = d_1 \; \mathsf{seq} \; d_2$ (induction step)

ASSUME: 1. $\llbracket d_1 \rrbracket = \{(T_1, \emptyset)\}$ (induction hypothesis)
2. $\llbracket d_2 \rrbracket = \{(T_2, \emptyset)\}$ (induction hypothesis)
3. $T_1, T_2 \subseteq \mathcal{H}$ (induction hypothesis)

PROVE: $\llbracket d_1 \; \mathsf{seq} \; d_2 \rrbracket = \{(T, \emptyset)\} \wedge T \subseteq \mathcal{H}$

$\langle 2 \rangle 1.$ $\llbracket d_1 \; \mathsf{seq} \; d_2 \rrbracket = \{o_1 \succsim o_2 | o_1 \in \llbracket d_1 \rrbracket \wedge o_2 \in \llbracket d_2 \rrbracket\}$

PROOF: Def. (35).

$\langle 2 \rangle 2.$ $\llbracket d_1 \; \mathsf{seq} \; d_2 \rrbracket = \{(T_1, \emptyset) \succsim (T_2, \emptyset)\}$

PROOF: $\langle 2 \rangle 1$ and induction hypothesis.

$\langle 2 \rangle 3.$ $\llbracket d_1 \; \mathsf{seq} \; d_2 \rrbracket = \{(T_1 \succsim T_2, (\emptyset \succsim T_2) \cup (\emptyset \succsim \emptyset) \cup (T_1 \succsim \emptyset))\}$

PROOF: $\langle 2 \rangle 2$ and def. (36).

$\langle 2 \rangle 4.$ $\llbracket d_1 \; \mathsf{seq} \; d_2 \rrbracket = \{(T, \emptyset)\}$, with $T = T_1 \succsim T_2 \subseteq \mathcal{H}$

$\langle 3 \rangle 1.$ $T_1 \succsim T_2$ is a set, and $T_1 \succsim T_2 \subseteq \mathcal{H}$

PROOF: Def. (34).

$\langle 3 \rangle 2.$ $(\emptyset \succsim T_2) \cup (\emptyset \succsim \emptyset) \cup (T_1 \succsim \emptyset) = \emptyset$

PROOF: Def. (34).

$\langle 3 \rangle 3.$ Q.E.D.

PROOF: $\langle 2 \rangle 3$, $\langle 3 \rangle 1$ and $\langle 3 \rangle 2$.

$\langle 2 \rangle 5.$ Q.E.D.

PROOF: $\langle 2 \rangle 4$

$\langle 1 \rangle 4.$ $d = d_1 \; \mathsf{par} \; d_2$ (induction step)

ASSUME: 1. $\llbracket d_1 \rrbracket = \{(T_1, \emptyset)\}$ (induction hypothesis)
2. $\llbracket d_2 \rrbracket = \{(T_2, \emptyset)\}$ (induction hypothesis)
3. $T_1, T_2 \subseteq \mathcal{H}$ (induction hypothesis)

PROVE: $\llbracket d_1 \; \mathsf{par} \; d_2 \rrbracket = \{(T, \emptyset)\} \wedge T \subseteq \mathcal{H}$

$\langle 2 \rangle 1.$ $\llbracket d_1 \; \mathsf{par} \; d_2 \rrbracket = \{o_1 \parallel o_2 | o_1 \in \llbracket d_1 \rrbracket \wedge o_2 \in \llbracket d_2 \rrbracket\}$

PROOF: Def. (38).

$\langle 2 \rangle 2.$ $\llbracket d_1 \; \mathsf{par} \; d_2 \rrbracket = \{(T_1, \emptyset) \parallel (T_2, \emptyset)\}$

PROOF: $\langle 2 \rangle 1$ and induction hypothesis.

$\langle 2 \rangle 3.$ $\llbracket d_1 \; \mathsf{par} \; d_2 \rrbracket = \{(T_1 \parallel T_2, (\emptyset \parallel T_2) \cup (\emptyset \parallel \emptyset) \cup (T_1 \parallel \emptyset))\}$

PROOF: $\langle 2 \rangle 2$ and def. (39).

$\langle 2 \rangle 4.$ $\llbracket d_1 \; \mathsf{par} \; d_2 \rrbracket = \{(T, \emptyset)\}$, with $T = T_1 \parallel T_2 \subseteq \mathcal{H}$

$\langle 3 \rangle 1.$ $T_1 \parallel T_2$ is a set, and $T_1 \parallel T_2 \subseteq \mathcal{H}$

PROOF: Def. (37).

$\langle 3 \rangle 2.$ $(\emptyset \parallel T_2) \cup (\emptyset \parallel \emptyset) \cup (T_1 \parallel \emptyset) = \emptyset$

PROOF: Def. (37).

$\langle 3 \rangle 3.$ Q.E.D.

PROOF: $\langle 2 \rangle 3$, $\langle 3 \rangle 1$ and $\langle 3 \rangle 2$.

$\langle 2 \rangle 5.$ Q.E.D.

PROOF: $\langle 2 \rangle 4$

$\langle 1 \rangle 5.$ Q.E.D.

PROOF: ⟨1⟩1, ⟨1⟩2, ⟨1⟩3 and ⟨1⟩4.

□

Given lemma 1 we will in the following associate $[\![d]\!]$ with T, and write $t \in [\![d]\!]$ when $t \in T$ as shorthand notation.

Lemma 2 *Given two simple diagrams $d_1, d_2 \in \mathcal{D}$, then*

$$env.(d_1 \ \mathsf{seq}\ d_2) = env.(d_1 \ \mathsf{par}\ d_2) = (env.d_1 \setminus ev.d_2) \cup (env.d_2 \setminus ev.d_1)$$

Proof of lemma 2

⟨1⟩1. $env.(d_1 \ \mathsf{seq}\ d_2)$

$$\begin{array}{lll}
= & \{(k,m) \,|\, k \in \{!,?\} \wedge m \in msg.(d_1 \ \mathsf{seq}\ d_2) & \\
 & \quad \wedge\ (k,m) \notin ev.(d_1 \ \mathsf{seq}\ d_2)\} & \text{(Def. 5 of } env.\_) \\
= & \{(k,m) \,|\, k \in \{!,?\} \wedge m \in (msg.d_1 \cup msg.d_2) & \text{(Defs. of } ev.\_ \\
 & \quad \wedge\ (k,m) \notin (ev.d_1 \cup ev.d_2)\} & \text{and } msg.\_)
\end{array}$$

⟨1⟩2. $env.(d_1 \ \mathsf{par}\ d_2)$

$$\begin{array}{lll}
= & \{(k,m) \,|\, k \in \{!,?\} \wedge m \in msg.(d_1 \ \mathsf{par}\ d_2) & \\
 & \quad \wedge\ (k,m) \notin ev.(d_1 \ \mathsf{par}\ d_2)\} & \text{(Def. 5 of } env.\_) \\
= & \{(k,m) \,|\, k \in \{!,?\} \wedge m \in (msg.d_1 \cup msg.d_2) & \text{(Defs. of } ev.\_ \\
 & \quad \wedge\ (k,m) \notin (ev.d_1 \cup ev.d_2)\} & \text{and } msg.\_)
\end{array}$$

⟨1⟩3. $\{(k,m) \,|\, k \in \{!,?\} \wedge m \in (msg.d_1 \cup msg.d_2) \ \wedge (k,m) \notin (ev.d_1 \cup ev.d_2)\}$

$$\begin{array}{ll}
= & \{(k,m) \,|\, k \in \{!,?\} \wedge (m \in msg.d_1 \vee m \in msg.d_2) \\
 & \quad \wedge\ (k,m) \notin ev.d_1 \wedge (k,m) \notin ev.d_2\} \\
= & \{(k,m) \,|\, (k \in \{!,?\} \wedge m \in msg.d_1 \wedge (k,m) \notin ev.d_1 \wedge (k,m) \notin ev.d_2) \\
 & \quad \vee\ (k \in \{!,?\} \wedge m \in msg.d_2 \wedge (k,m) \notin ev.d_1 \wedge (k,m) \notin ev.d_2) \\
= & \{(k,m) \,|\, k \in \{!,?\} \wedge m \in msg.d_1 \wedge (k,m) \notin ev.d_1 \wedge (k,m) \notin ev.d_2\} \\
 & \cup \{(k,m) \,|\, k \in \{!,?\} \wedge m \in msg.d_2 \wedge (k,m) \notin ev.d_1 \wedge (k,m) \notin ev.d_2\} \\
= & (\{(k,m) \,|\, k \in \{!,?\} \wedge m \in msg.d_1 \wedge (k,m) \notin ev.d_1\} \ \cap \\
 & \quad \{(k,m) \,|\, (k,m) \notin ev.d_2\}) \ \cup \\
 & (\{(k,m) \,|\, k \in \{!,?\} \wedge m \in msg.d_2 \wedge (k,m) \notin ev.d_2\} \ \cap \\
 & \quad \{(k,m) \,|\, (k,m) \notin ev.d_1\}) \\
= & (\{(k,m) \,|\, k \in \{!,?\} \wedge m \in msg.d_1 \wedge (k,m) \notin ev.d_1\} \ \setminus \\
 & \quad \overline{\{(k,m) \,|\, (k,m) \notin ev.d_2\}}) \ \cup \\
 & (\{(k,m) \,|\, k \in \{!,?\} \wedge m \in msg.d_2 \wedge (k,m) \notin ev.d_2\} \ \setminus \\
 & \quad \overline{\{(k,m) \,|\, (k,m) \notin ev.d_1\}}) \\
= & (env.d_1 \setminus \{(k,m) \,|\, (k,m) \in ev.d_2\}) \\
 & \cup\ (env.d_2 \setminus \{(k,m) \,|\, (k,m) \in ev.d_1\}) \\
= & (env.d_1 \setminus ev.d_2) \cup (env.d_2 \setminus ev.d_1)
\end{array}$$

PROOF: Def. of $env.\_$, and set theory.

⟨1⟩4. Q.E.D.

PROOF: ⟨1⟩1, ⟨1⟩2 and ⟨1⟩3.

□

Definition 7 *The function $ev.\_ \in \mathcal{E}^* \to \mathbb{P}(\mathcal{E})$ returns the events of a trace t. The function is defined recursively:*

$$\begin{array}{rcl}
ev.\langle\rangle & \overset{\mathsf{def}}{=} & \emptyset \\
ev.(\langle e\rangle ^\frown t) & \overset{\mathsf{def}}{=} & \{e\} \cup ev.t
\end{array}$$

Lemma 3 *Given $t_1, t_2 \in \mathcal{E}^*$, then $ev.(t_1 ^\frown t_2) = ev.t_1 \cup ev.t_2$*

Proof of lemma 3
ASSUME: $t_1, t_2 \in \mathcal{E}^*$
PROVE: $ev.(t_1 \frown t_2) = ev.t_1 \cup ev.t_2$
PROOF: by induction on t_1.
⟨1⟩1. Induction start. $t_1 = \langle\rangle$
  ⟨2⟩1. $ev.(t_1 \frown t_2) = ev.(\langle\rangle \frown t_2) = ev.t_2$
    PROOF: ⟨1⟩1 and identity of $\_\frown\_$.
  ⟨2⟩2. $ev.t_1 \cup ev.t_2 = ev.\langle\rangle \cup ev.t_2 = \emptyset \cup ev.t_2 = ev.t_2$
    PROOF: ⟨1⟩1 and def. of $ev.\_$.
  ⟨2⟩3. Q.E.D.
    PROOF: ⟨2⟩1 and ⟨2⟩2.
⟨1⟩2. Induction step
    ASSUME: $ev.(t_1 \frown t_2) = ev.t_1 \cup ev.t_2$ (induction hypothesis)
    PROVE: $ev.((\langle e\rangle \frown t_1) \frown t_2) = ev.(\langle e\rangle \frown t_1) \cup ev.t_2$
  ⟨2⟩1. $ev.((\langle e\rangle \frown t_1) \frown t_2) = ev.(\langle e\rangle \frown (t_1 \frown t_2)) = \{e\} \cup ev.(t_1 \frown t_2)$
      $= \{e\} \cup ev.t_1 \cup ev.t_2$
    PROOF: Associativity of $\_\frown\_$, def. of $ev.\_$ and induction hypothesis.
  ⟨2⟩2. $ev.(\langle e\rangle \frown t_1) \cup ev.t_2 = \{e\} \cup ev.t_1 \cup ev.t_2$
    PROOF: Def. of $ev.\_$.
  ⟨2⟩3. Q.E.D.
    PROOF: ⟨2⟩1 and ⟨2⟩2.
⟨1⟩3. Q.E.D.
  ⟨1⟩1 and ⟨1⟩2.

□

Lemma 4 *Given simple diagrams $d, d' \in \mathcal{D}$ without repetition of events. Then, for all traces t, if there exist $\beta, \beta' \in \mathcal{B}$ such that*

$$[\beta, d] \xrightarrow{t} [\beta', d']$$

then

$$ev.t = ev.d \setminus ev.d'$$

Proof of lemma 4
ASSUME: $[\beta, d] \xrightarrow{t} [\beta', d']$
PROVE: $e \in ev.t \iff e \in (ev.d \setminus ev.d')$
⟨1⟩1. $e \in (ev.d \setminus ev.d') \iff (e \in ev.d \wedge e \notin ev.d')$
  PROOF: Set theory.
⟨1⟩2. $e \in ev.t \Rightarrow e \in ev.d \wedge e \notin ev.d'$
  PROOF: Assume $e \in ev.t$. Then there must exist t_1, t_2 such that $t = t_1 \frown \langle e\rangle \frown t_2$ and $\beta_1, \beta_2, d_1, d_2$ such that

$$[\beta, d] \xrightarrow{t_1} [\beta_1, d_1] \xrightarrow{e} [\beta_2, d_2] \xrightarrow{t_2} [\beta', d']$$

  By the assumption that d, and therefore also d_1 and d_2, are simple and have no repetition of events, and by rules (3)-(12) we must have that $e \in ev.d_1$ and $e \notin ev.d_2$. Because events cannot reappear during rewrite of a diagram, this also means that $e \in ev.d$ and $e \notin ev.d'$
⟨1⟩3. $e \in ev.d \wedge e \notin ev.d' \Rightarrow e \in ev.t$

PROOF: Assume $e \in ev.d$ and $e \notin ev.d'$. Because application of rules (3)-(12) is the only way of removing an event there must exist a sequence of rewrites such that

$$[\beta, d] \xrightarrow{t_1 \frown \langle e \rangle \frown t_2} [\beta', d']$$

which means there must exist t_1 and t_2 such that $t = t_1 \frown \langle e \rangle \frown t_2$. By the definition of $ev.t$ we then have that $e \in ev.t$.

$\langle 1 \rangle 4$. Q.E.D.

PROOF: $\langle 1 \rangle 1$, $\langle 1 \rangle 2$ and $\langle 1 \rangle 3$.

$\square$

Lemma 5 *Given simple diagram $d \in \mathcal{D}$. For all traces t, if there exist $\beta, \beta' \in \mathcal{B}$ such that*

$$[\beta, d] \xrightarrow{t} [\beta', \mathsf{skip}]$$

then

$$ev.t = ev.d$$

Proof of lemma 5 Assume $[\beta, d] \xrightarrow{t} [\beta', \mathsf{skip}]$. By lemma 4 and the definition of $ev.\mathsf{skip}$, we have

$$ev.t = ev.d \setminus ev.\mathsf{skip} = ev.d \setminus \emptyset = ev.d$$

$\square$

Definition 8 *We formally define the filtering operator _Ⓢ_ for finite sequences over A:*

$$\begin{array}{rcl} V Ⓢ \langle\rangle & \overset{\mathsf{def}}{=} & \langle\rangle \\ v \in V \Rightarrow V Ⓢ (\langle v \rangle \frown t) & \overset{\mathsf{def}}{=} & \langle v \rangle \frown (V Ⓢ t) \\ v \notin V \Rightarrow V Ⓢ (\langle v \rangle \frown t) & \overset{\mathsf{def}}{=} & V Ⓢ t \end{array}$$

(with $V \subseteq A$ and $t \in A^*$).

Lemma 6 *Given simple diagram $d \in \mathcal{D}$. For all $t \in \mathcal{E}^*$, if there exists β such that*

$$[env^!_{\mathcal{M}}.d, d] \xrightarrow{t} [\beta, \mathsf{skip}]$$

then

$$t \in \mathcal{H}$$

Proof of lemma 6

ASSUME: $[env^!_{\mathcal{M}}.d, d] \xrightarrow{t} [\beta, \mathsf{skip}]$

PROVE: $t \in \mathcal{H}$

$\langle 1 \rangle 1$. $t \in \mathcal{H} \iff$

$\forall i \in [1..\#t] : k.t[i] = ! \Rightarrow \#(\{(!, m.t[i])\} Ⓢ t|_i) > \#(\{(?, m.t[i])\} Ⓢ t|_i)$

PROOF: Semantic constraint on traces (30).

$\langle 1 \rangle 2$. $\forall i \in [1..\#t] : k.t[i] = ! \Rightarrow \#(\{(!, m.t[i])\} Ⓢ t|_i) > \#(\{(?, m.t[i])\} Ⓢ t|_i)$

LET: j be arbitrary $j \in [1..\#t]$

PROVE: $k.t[j] = ! \Rightarrow \#(\{(!, m.t[j])\} Ⓢ t|_j) > \#(\{(?, m.t[j])\} Ⓢ t|_j)$

$\langle 2 \rangle 1$. CASE: $t[j]$ transmit event

LET: $t[j] = (!, m)$. Then $k.t[j] = !$ and $m.t[j] = m$
PROVE: $\#(\{(!, m)\} \circledS t|_j) > \#(\{(?, m)\} \circledS t|_j)$
$\langle 3\rangle 1.$ $\#(\{(!, m)\} \circledS t|_j) = 1$
PROOF: The assumptions that d is simple and has no repetition of events, and lemma 5.
$\langle 3\rangle 2.$ $\#(\{(?, m)\} \circledS t|_j) = 0$
PROOF: by contradiction
ASSUME: $\#(\{(?, m)\} \circledS t|_j) > 0$
$\langle 4\rangle 1.$ There exist s, s' such that

$$t|_j = s \frown \langle (?, m)\rangle \frown s'$$

PROOF: Defs. of $\#\_$ and $\_\circledS\_$.
$\langle 4\rangle 2.$ There exist $\beta', \beta'', \beta'' \in \mathcal{B}, d', d'', d'' \in \mathcal{D}$ such that

$$[env^!_{\mathcal{M}}.d, d] \xrightarrow{s} [\beta', d'] \xrightarrow{(?,m)} [\beta'', d''] \xrightarrow{s'} [\beta''', d''']$$

PROOF: Assumption and $\langle 4\rangle 1$.
$\langle 4\rangle 3.$ $m \in \beta'$
PROOF: $\langle 4\rangle 2$ and rules (3)-(12).
$\langle 4\rangle 4.$ $m \notin env^!_{\mathcal{M}}.d$
PROOF: $(!, m) \in ev.t = ev.d$ (by assumption, and lemma 5) and def. 5 of $env^!_{\mathcal{M}}.\_$.
$\langle 4\rangle 5.$ $(!, m) \in ev.s$
PROOF: $\langle 4\rangle 3$, $\langle 4\rangle 4$ and rules (3)-(12).
$\langle 4\rangle 6.$ $\#(\{(!, m)\} \circledS t|_{j-1}) = 0$
PROOF: $\langle 3\rangle 1$ and defs. of $\#\_$ and $\_\circledS\_$ and assumption that $t[j] = (!, m)$.
$\langle 4\rangle 7.$ $(!, m) \notin ev.t|_{j-1}$
PROOF: $\langle 4\rangle 6$ and def. of $\_\circledS\_$.
$\langle 4\rangle 8.$ $(!, m) \notin ev.s$
PROOF: $\langle 4\rangle 7$, $\langle 4\rangle 1$ and def. of $ev.\_$.
$\langle 4\rangle 9.$ Q.E.D.
PROOF: $\langle 4\rangle 5$ and $\langle 4\rangle 8$ yield a contradiction, so we must have that $\#(\{(?, m)\} \circledS t|_j) = 0$
$\langle 3\rangle 3.$ Q.E.D.
PROOF: $\langle 3\rangle 1$ and $\langle 3\rangle 2$
$\langle 2\rangle 2.$ CASE: $t[j]$ receive event
LET: $t[j] = (?, m)$
$\langle 3\rangle 1.$ Q.E.D.
PROOF: Trivial since $k.t[j] = ?$ and therefore $(k.t[j] = !) =$ **false**.
$\langle 2\rangle 3.$ Q.E.D.
PROOF: $\langle 2\rangle 1$ and $\langle 2\rangle 2$.
$\langle 1\rangle 3.$ Q.E.D.
PROOF: $\langle 1\rangle 1$ and $\langle 1\rangle 2$.

□

Lemma 7 *Given simple diagrams $d_1, d_2 \in \mathcal{D}$. For all traces t, if there exists $\beta \in \mathcal{B}$ such that*

$$[env^!_{\mathcal{M}}.(d_1 \ \mathsf{seq}\ d_2), d_1 \ \mathsf{seq}\ d_2] \xrightarrow{t} [\beta, \mathsf{skip}]$$

then there exist traces t_1, t_2 *and* $\beta_1, \beta_2 \in \mathcal{B}$ *such that*

$$\begin{aligned}
&[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \mathsf{skip}] \wedge \\
&[end^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \mathsf{skip}] \wedge \\
&\forall l \in \mathcal{L} : e.l \circledS t = e.l \circledS t_1 {}^\frown e.l \circledS t_2
\end{aligned}$$

Proof of lemma 7

ASSUME: There exists $\beta \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1 \textsf{ seq } d_2] \xrightarrow{t} [\beta, \mathsf{skip}]$

PROVE: There exist traces t_1, t_2, and $\beta_1, \beta_2 \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \mathsf{skip}] \wedge$
$[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \mathsf{skip}] \wedge$
$\forall l \in \mathcal{L} : e.l \circledS t = e.l \circledS t_1 {}^\frown e.l \circledS t_2$

$\langle 1 \rangle 1.$ There exist trace t_1, t_2, and $\beta_1, \beta_2 \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \mathsf{skip}] \wedge$
$[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \mathsf{skip}]$

$\langle 2 \rangle 1.$ LET: $t = \langle e \rangle {}^\frown t'$

$\langle 2 \rangle 2.$ There exist $\beta_1', \beta_2' \in \mathcal{B}, d_1', d_2' \in \mathcal{D}$ such that
$[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{e} [\beta_1', d_1'] \veebar$
$[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{e} [\beta_2', d_2']$
(where $\veebar$ denotes exclusive or)

$\langle 3 \rangle 1.$ There exist $\beta' \in \mathcal{B}, d' \in \mathcal{D}$ such that
$[env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1 \textsf{ seq } d_2] \xrightarrow{e} [\beta', d'] \xrightarrow{t'} [\beta, \mathsf{skip}]$
PROOF: Assumption and $\langle 2 \rangle 1$

$\langle 3 \rangle 2.$ $[env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1 \textsf{ seq } d_2] \xrightarrow{e} [\beta', d_1' \textsf{ seq } d_2] \veebar$
$[env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1 \textsf{ seq } d_2] \xrightarrow{e} [\beta', d_1 \textsf{ seq } d_2']$
PROOF: $\langle 3 \rangle 1$, rules (3), (9) and (10), the assumption $ev.d_1 \cap ev.d_2 = \emptyset$ and lemma 4.

$\langle 3 \rangle 3.$ CASE: $[env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1 \textsf{ seq } d_2] \xrightarrow{e} [\beta', d_1' \textsf{ seq } d_2]$
PROVE: $[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{e} [\beta_1', d_1'] \wedge [env^!_{\mathcal{M}}.d_2, d_2] \overset{e}{\not\rightarrow}$

$\langle 4 \rangle 1.$ $[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{e} [\beta_1', d_1']$

$\langle 5 \rangle 1.$ $\Pi(ll.(d_1 \textsf{ seq } d_2), env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1 \textsf{ seq } d_2) \xrightarrow{e}$
$\Pi(ll.(d_1 \textsf{ seq } d_2), env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1' \textsf{ seq } d_2)$
PROOF: By rule (3) this is a necessary condition for $\langle 3 \rangle 3$ to be satisfied.

$\langle 5 \rangle 2.$ $\Pi(ll.d_1 \cap ll.(d_1 \textsf{ seq } d_2), env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1) \xrightarrow{e}$
$\Pi(ll.d_1 \cap ll.(d_1 \textsf{ seq } d_2), env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1')$
PROOF: By rule (9) this is a necessary condition for $\langle 5 \rangle 1$ to be satisfied.

$\langle 5 \rangle 3.$ $\Pi(ll.d_1, env^!_{\mathcal{M}}.d_1, d_1) \xrightarrow{e} \Pi(ll.d_1, env^!_{\mathcal{M}}.d_1, d_1')$

$\langle 6 \rangle 1.$ $ll.d_1 \cap ll.(d_1 \textsf{ seq } d_2) = ll.d_1 \cap (ll.d_1 \cup ll.d_2) = ll.d_1$
PROOF: Def. of $ll.\_$

$\langle 6 \rangle 2.$ CASE: $e = (!, m)$
It follows from the rules that the state of the communication medium is irrelevant.

$\langle 6 \rangle 3.$ CASE: $e = (?, m)$
In this case it must be shown that $m \in env^!_{\mathcal{M}}.d_1$.

⟨7⟩1. $m \in msg.d_1$

PROOF: Defs. of $ev.\_$ and $msg.\_$, rules (8)-(12), ⟨6⟩3 and ⟨5⟩2.

⟨7⟩2. $m \in env^{!}_{\mathcal{M}}.(d_1 \text{ seq } d_2)$

PROOF: It follows from the recursive use of rules (3)-(12), and def. of *ready* that this is a necessary condition for ⟨3⟩3 to be satisfied.

⟨7⟩3. $m \in env^{!}_{\mathcal{M}}.(d_1 \text{ seq } d_2) \iff m \in env^{!}_{\mathcal{M}}.d_1 \wedge (!, m) \notin ev.d_2$

PROOF: $m \in env^{!}_{\mathcal{M}}.(d_1 \text{ seq } d_2)$

$\Leftrightarrow$	$(!, m) \in env.(d_1 \text{ seq } d_2)$	(Def. 5)
$\Leftrightarrow$	$(!, m) \in (env.d_1 \setminus ev.d_2) \cup (env.d_2 \setminus ev.d_1)$	(Lemma 2)
$\Leftrightarrow$	$((!, m) \in env.d_1 \wedge (!, m) \notin ev.d_2) \vee$ $((!, m) \in env.d_2 \wedge (!, m) \notin ev.d_1)$	
$\Leftrightarrow$	$((!, m) \in env.d_1 \wedge (!, m) \notin ev.d_2) \vee$ $((!, m) \in env.d_2 \wedge (!, m) \in env.d_1)$	(⟨7⟩1)
$\Leftrightarrow$	$(!, m) \in env.d_1 \wedge$ $((!, m) \notin ev.d_2 \vee (!, m) \in env.d_2)$	
$\Leftrightarrow$	$(!, m) \in env.d_1 \wedge ((!, m) \notin ev.d_2 \vee$ $((!, m) \notin ev.d_2 \wedge m \in msg.d_2))$	(Def. 5)
$\Leftrightarrow$	$(!, m) \in env.d_1 \wedge (!, m) \notin ev.d_2$	
$\Leftrightarrow$	$m \in env^{!}_{\mathcal{M}}.d_1 \ \wedge (!, m) \notin ev.d_2$	(Def. 5)

⟨7⟩4. Q.E.D.

PROOF: ⟨7⟩2 and ⟨7⟩3.

⟨6⟩4. Q.E.D.

PROOF: ⟨5⟩2, ⟨6⟩1, ⟨6⟩2 and ⟨6⟩3.

⟨5⟩4. Q.E.D.

PROOF: ⟨5⟩3 and rule (3)

⟨4⟩2. $[env^{!}_{\mathcal{M}}.d_2, d_2] \overset{e}{\not\to}$

⟨5⟩1. $e \notin ev.d_2$

PROOF: ⟨3⟩3 and application of rules imply $e \in ev.d_1$, and by assumption, $ev.d_1 \cap ev.d_2 = \emptyset$.

⟨5⟩2. $e \notin ev.d_2 \setminus ev.d_2'$

PROOF: ⟨5⟩1.

⟨5⟩3. $\neg([env^{!}_{\mathcal{M}}.d_2, d_2] \xrightarrow{e} [\beta_2', d_2'])$

PROOF: ⟨5⟩2 and lemma 4.

⟨5⟩4. Q.E.D.

PROOF: ⟨5⟩3

⟨4⟩3. Q.E.D.

PROOF: ⟨4⟩1 and ⟨4⟩2

⟨3⟩4. CASE: $[env^{!}_{\mathcal{M}}.(d_1 \text{ seq } d_2), d_1 \text{ seq } d_2] \xrightarrow{e} [\beta', d_1 \text{ seq } d_2']$

PROVE: $[env^{!}_{\mathcal{M}}.d_2, d_2] \xrightarrow{e} [\beta_2', d_2'] \wedge [env^{!}_{\mathcal{M}}.d_1, d_1] \overset{e}{\not\to}$

⟨4⟩1. $[env^{!}_{\mathcal{M}}.d_2, d_2] \xrightarrow{e} [\beta_2', d_2']$

⟨5⟩1. $\Pi(ll.(d_1 \text{ seq } d_2), env^{!}_{\mathcal{M}}.(d_1 \text{ seq } d_2), d_1 \text{ seq } d_2) \xrightarrow{e}$
$\Pi(ll.(d_1 \text{ seq } d_2), env^{!}_{\mathcal{M}}.(d_1 \text{ seq } d_2), d_1 \text{ seq } d_2')$

PROOF: By rule (3) this is a necessary condition for ⟨3⟩4 to be satisfied.

⟨5⟩2. $\Pi(ll.(d_1 \text{ seq } d_2) \setminus ll.d_1, env^{!}_{\mathcal{M}}.(d_1 \text{ seq } d_2), d_2) \xrightarrow{e}$
$\Pi(ll.(d_1 \text{ seq } d_2) \setminus ll.d_2, env^{!}_{\mathcal{M}}.(d_1 \text{ seq } d_2), d_2')$

PROOF: By rule (10) this is a necessary condition for ⟨5⟩1 to be satisfied.

$\langle 5\rangle 3.$ $\Pi(ll.d_2, env^!_{\mathcal{M}}.d_2, d_2) \xrightarrow{e} \Pi(ll.d_2, env^!_{\mathcal{M}}.d_2, d_2')$

$\langle 6\rangle 1.$ $l.e \in ll.d_2$

$\langle 7\rangle 1.$ $l.e \in ll.(d_1 \mathsf{~seq~} d_2) \setminus ll.d_1$

PROOF: This follows from $\langle 5\rangle 2$ and the rules.

$\langle 7\rangle 2.$ $l.e \in ll.(d_1 \mathsf{~seq~} d_2) \setminus ll.d_1 \Rightarrow l.e \in (ll.d_1 \cup ll.d_2) \wedge l.e \notin ll.d_1 \Rightarrow (l.e \in ll.d_1 \vee l.e \in ll.d_2) \wedge l.e \notin ll.d_1 \Rightarrow l.e \in ll.d_2$

PROOF: Def. of $ll.\_$ and $\langle 7\rangle 1$.

$\langle 7\rangle 3.$ Q.E.D.

PROOF: $\langle 7\rangle 2$.

$\langle 6\rangle 2.$ CASE: $e = (!, m)$

It follows from the rules that the state of the communication medium is irrelevant.

$\langle 6\rangle 3.$ CASE: $e = (?, m)$

In this case it must be shown that $m \in env^!_{\mathcal{M}}.d_2$.

$\langle 7\rangle 1.$ $m \in msg.d_2$

PROOF: Defs. of $ev.\_$ and $msg.\_$, rules (8)-(12), $\langle 6\rangle 3$ and $\langle 5\rangle 2$.

$\langle 7\rangle 2.$ $m \in env^!_{\mathcal{M}}.(d_1 \mathsf{~seq~} d_2)$

PROOF: It follows from the recursive use of rules (3)-(12), and def. of *ready* that this is a necessary condition for $\langle 3\rangle 4$ to be satisfied.

$\langle 7\rangle 3.$ $m \in env^!_{\mathcal{M}}.(d_1 \mathsf{~seq~} d_2) \iff m \in env^!_{\mathcal{M}}.d_2 \wedge (!, m) \notin ev.d_1$

PROOF: $m \in env^!_{\mathcal{M}}.(d_2 \mathsf{~seq~} d_1)$

$$\begin{array}{lll}
\Leftrightarrow & (!, m) \in env.(d_1 \mathsf{~seq~} d_2) & \text{(Def. 5)} \\
\Leftrightarrow & (!, m) \in (env.d_1 \setminus ev.d_2) \cup (env.d_2 \setminus ev.d_1) & \text{(Lemma 2)} \\
\Leftrightarrow & ((!, m) \in env.d_1 \wedge (!, m) \notin ev.d_2) \vee & \\
 & ((!, m) \in env.d_2 \wedge (!, m) \notin ev.d_1) & \\
\Leftrightarrow & ((!, m) \in env.d_2 \wedge (!, m) \notin ev.d_1) \vee & \\
 & ((!, m) \in env.d_1 \wedge (!, m) \in env.d_2) & (\langle 7\rangle 1) \\
\Leftrightarrow & (!, m) \in env.d_2 \wedge & \\
 & ((!, m) \notin ev.d_1 \vee (!, m) \in env.d_1) & \\
\Leftrightarrow & (!, m) \in env.d_2 \wedge ((!, m) \notin ev.d_1 \vee & \\
 & ((!, m) \notin ev.d_1 \wedge m \in msg.d_1)) & \text{(Def. 5)} \\
\Leftrightarrow & (!, m) \in env.d_2 \wedge (!, m) \notin ev.d_1 & \\
\Leftrightarrow & m \in env^!_{\mathcal{M}}.d_2 \ \wedge (!, m) \notin ev.d_1 & \text{(Def. 5)}
\end{array}$$

$\langle 7\rangle 4.$ Q.E.D.

PROOF: $\langle 7\rangle 2$ and $\langle 7\rangle 3$.

$\langle 6\rangle 4.$ Q.E.D.

PROOF: $\langle 5\rangle 2$, $\langle 6\rangle 1$, $\langle 6\rangle 2$ and $\langle 6\rangle 3$.

$\langle 5\rangle 4.$ Q.E.D.

PROOF: $\langle 5\rangle 3$ and rule (3)

$\langle 4\rangle 2.$ $[env^!_{\mathcal{M}}.d_1, d_1] \overset{e}{\nrightarrow}$

$\langle 5\rangle 1.$ $e \notin ev.d_1$

PROOF: $\langle 3\rangle 4$ and application of rules imply $e \in ev.d_1$, and by assumption, $ev.d_1 \cap ev.d_2 = \emptyset$.

$\langle 5\rangle 2.$ $e \notin ev.d_1 \setminus ev.d_1'$

PROOF: $\langle 5\rangle 1$.

$\langle 5\rangle 3.$ $\neg([env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{e} [\beta_1', d_1'])$

PROOF: $\langle 5\rangle 2$ and lemma 4.

$\langle 5\rangle 4.$ Q.E.D.

PROOF: $\langle 5\rangle 3$

$\langle 4\rangle 3$. Q.E.D.

PROOF: $\langle 4\rangle 1$ and $\langle 4\rangle 2$

$\langle 3\rangle 5$. Q.E.D.

PROOF: $\langle 3\rangle 2$, $\langle 3\rangle 3$ and $\langle 3\rangle 4$.

$\langle 2\rangle 3$. Q.E.D.

PROOF: We now have that $[\beta', d'] \xrightarrow{t'} [\beta, \mathsf{skip}]$ where $d' = d'_1$ seq d_2 or $d' = d_1$ seq d'_2. By substituting $[env^!_{\mathcal{M}}.d', d'] \xrightarrow{t'} [\beta, \mathsf{skip}]$ for the assumption we may repeat the argument until $t' = \langle\rangle$, and we get $\langle 1\rangle 1$. The justification of the substitution is that if $e = (?, m)$ then $\beta' = env^!_{\mathcal{M}}.d \setminus \{m\}$, but we know that $e \notin ev.d'$. If $e = (!, m)$, then $\beta' = env^!_{\mathcal{M}}.d \cup \{m\}$, and $env^!_{\mathcal{M}}.d' = env^!_{\mathcal{M}}.d \cup \{m\} \Leftrightarrow (?, m) \in ev.d'$ and $env^!_{\mathcal{M}}.d' = env^!_{\mathcal{M}}.d \Leftrightarrow (?, m) \notin ev.d$.

$\langle 1\rangle 2$. $\forall l \in \mathcal{L} : e.l Ⓢ t = e.l Ⓢ t_1 \frown e.l Ⓢ t_2$

PROOF: by contradiction

ASSUME: 1. The traces t_1 and t_2 from $\langle 1\rangle 1$

2. l arbitrary chosen lifeline in $\mathcal{L}$

3. $e.l Ⓢ t \neq e.l Ⓢ t_1 \frown e.l Ⓢ t_2$

$\langle 2\rangle 1$. $\#t = \#t_1 + \#t_2$

PROOF: This follows from $ev.t = ev.d = ev.d_1 \cup ev.d_2 = ev.t_1 \cup ev.t_2$ (lemma 5) and the assumption that there is no repetition of events in d.

$\langle 2\rangle 2$. There exist $e_1, e_2 \in e.l$ such that $\{e_1, e_2\} Ⓢ t \neq \{e_1, e_2\} Ⓢ t_1 \frown \{e_1, e_2\} Ⓢ t_2$

PROOF: By assumption 3, $\langle 2\rangle 1$ and the assumption that there is no repetition of events, such events must exist.

$\langle 2\rangle 3$. CASE: $e_1, e_2 \notin ev.d_1 \cup ev.d_2$

$\langle 3\rangle 1$. $\{e_1, e_2\} Ⓢ t = \langle\rangle$

PROOF: Lemma 5

$\langle 3\rangle 2$. $\{e_1, e_2\} Ⓢ t_1 \frown \{e_1, e_2\} Ⓢ t_2 = \langle\rangle \frown \langle\rangle = \langle\rangle$

PROOF: Lemma 5

$\langle 2\rangle 4$. CASE: $e_1 \notin ev.d_1 \cup ev.d_2 \wedge e_2 \in ev.d_1$

$\langle 3\rangle 1$. $\{e_1, e_2\} Ⓢ t = \langle e_2\rangle$

PROOF: Lemma 5

$\langle 3\rangle 2$. $\{e_1, e_2\} Ⓢ t_1 \frown \{e_1, e_2\} Ⓢ t_2 = \langle e_2\rangle \frown \langle\rangle = \langle e_2\rangle$

PROOF: Lemma 5

$\langle 2\rangle 5$. CASE: $e_1 \notin ev.d_1 \cup ev.d_2 \wedge e_2 \in ev.d_2$

$\langle 3\rangle 1$. $\{e_1, e_2\} Ⓢ t = \langle e_2\rangle$

PROOF: Lemma 5

$\langle 3\rangle 2$. $\{e_1, e_2\} Ⓢ t_1 \frown \{e_1, e_2\} Ⓢ t_2 = \langle\rangle \frown \langle e_2\rangle = \langle e_2\rangle$

PROOF: Lemma 5

$\langle 2\rangle 6$. CASE: $e_1 \in ev.d_1 \wedge e_2 \notin ev.d_1 \cup ev.d_2$

$\langle 3\rangle 1$. $\{e_1, e_2\} Ⓢ t = \langle e_1\rangle$

PROOF: Lemma 5

$\langle 3\rangle 2$. $\{e_1, e_2\} Ⓢ t_1 \frown \{e_1, e_2\} Ⓢ t_2 = \langle e_1\rangle \frown \langle\rangle = \langle e_1\rangle$

PROOF: Lemma 5

$\langle 2\rangle 7$. CASE: $e_1 \in ev.d_2 \wedge e_2 \notin ev.d_1 \cup ev.d_2$

$\langle 3\rangle 1$. $\{e_1, e_2\} Ⓢ t = \langle e_1\rangle$

PROOF: Lemma 5

$\langle 3\rangle 2$. $\{e_1, e_2\} Ⓢ t_1 \frown \{e_1, e_2\} Ⓢ t_2 = \langle\rangle \frown \langle e_1\rangle = \langle e_1\rangle$

PROOF: Lemma 5

$\langle 2 \rangle 8$. CASE: $e_1, e_2 \in ev.d_1$

$\langle 3 \rangle 1$. $e_1, e_2 \notin ev.d_2$

PROOF: Assumption $ev.d_1 \cap ev.d_2 = \emptyset$.

$\langle 3 \rangle 2$. $t_1 = ev.d_1 \circledS t \wedge t_2 = ev.d_2 \circledS t$

PROOF: Lemma 4 and assumption $ev.d_1 \cap ev.d_2 = \emptyset$

$\langle 3 \rangle 3$. $\{e_1, e_2\} \circledS t_1 \frown \{e_1, e_2\} \circledS t_2$

$= \{e_1, e_2\} \circledS (ev.d_1 \circledS t) \frown \{e_1, e_2\} \circledS (ev.d_2 \circledS t)$

$= (\{e_1, e_2\} \cap ev.d_1) \circledS t \frown (\{e_1, e_2\} \cap ev.d_2) \circledS t$

$= \{e_1, e_2\} \circledS t \frown \emptyset \circledS t$

$= \{e_1, e_2\} \circledS t \frown \langle\rangle$

$= \{e_1, e_2\} \circledS t$

PROOF: $\langle 3 \rangle 2$, $\{e_1, e_2\} \subseteq ev.d_1$ ($\langle 2 \rangle 8$) and $\{e_1, e_2\} \cap ev.d_2 = \emptyset$ ($\langle 3 \rangle 1$).

$\langle 2 \rangle 9$. CASE: $e_1, e_2 \in ev.d_2$

$\langle 3 \rangle 1$. $e_1, e_2 \notin ev.d_1$

PROOF: Assumption $ev.d_1 \cap ev.d_2 = \emptyset$.

$\langle 3 \rangle 2$. $t_1 = ev.d_1 \circledS t \wedge t_2 = ev.d_2 \circledS t$

PROOF: Lemma 4 and assumption $ev.d_1 \cap ev.d_2 = \emptyset$

$\langle 3 \rangle 3$. $\{e_1, e_2\} \circledS t_1 \frown \{e_1, e_2\} \circledS t_2$

$= \{e_1, e_2\} \circledS (ev.d_1 \circledS t) \frown \{e_1, e_2\} \circledS (ev.d_2 \circledS t)$

$= \{e_1, e_2\} \cap ev.d_1 \circledS t \frown \{e_1, e_2\} \cap ev.d_2 \circledS t$

$= \emptyset \circledS t \frown \{e_1, e_2\} \circledS t$

$= \langle\rangle \frown \{e_1, e_2\} \circledS t$

$= \{e_1, e_2\} \circledS t$

PROOF: $\langle 3 \rangle 2$, $\{e_1, e_2\} \subseteq ev.d_2$ ($\langle 2 \rangle 9$) and $\{e_1, e_2\} \cap ev.d_1 = \emptyset$ ($\langle 3 \rangle 1$).

$\langle 2 \rangle 10$. CASE: $e_1 \in ev.d_1 \wedge e_2 \in ev.d_2$

$\langle 3 \rangle 1$. $e_1 \notin ev.d_2 \wedge e_2 \notin ev.d_1$

PROOF: Assumption $ev.d_1 \cap ev.d_2 = \emptyset$.

$\langle 3 \rangle 2$. $t_1 = ev.d_1 \circledS t \wedge t_2 = ev.d_2 \circledS t$

PROOF: Lemma 4 and assumption $ev.d_1 \cap ev.d_2 = \emptyset$

$\langle 3 \rangle 3$. $\{e_1, e_2\} \circledS t_1 = \{e_1, e_2\} \circledS (ev.d_1 \circledS t) = \{e_1, e_2\} \cap ev.d_1 \circledS t = \{e_1\} \circledS t = \langle e_1 \rangle$

PROOF: Lemma 4, $\langle 2 \rangle 10$, $\langle 3 \rangle 1$ and $\langle 3 \rangle 2$.

$\langle 3 \rangle 4$. $\{e_1, e_2\} \circledS t_2 = \{e_1, e_2\} \circledS (ev.d_2 \circledS t) = \{e_1, e_2\} \cap ev.d_2 \circledS t = \{e_2\} \circledS t = \langle e_2 \rangle$

PROOF: Lemma 4, $\langle 2 \rangle 10$, $\langle 3 \rangle 1$ and $\langle 3 \rangle 2$.

$\langle 3 \rangle 5$. $\{e_1, e_2\} \circledS t = \langle e_2, e_1 \rangle$

PROOF: By assumption 3, $\langle 3 \rangle 3$ and $\langle 3 \rangle 4$ we have that $\{e_1, e_2\} \circledS t \neq \langle e_1, e_2 \rangle$. By $\langle 2 \rangle 10$ and lemma 5 we have that $\{e_, e_2\} \subseteq ev.t$.

$\langle 3 \rangle 6$. There exist traces s_1, s_2, s_3 such that $t = s_1 \frown \langle e_2 \rangle \frown s_2 \frown \langle e_1 \rangle \frown s_3$

PROOF: $\langle 3 \rangle 5$.

$\langle 3 \rangle 7$. There exist $\beta, \beta' \in \mathcal{B}, d_1', d_2', d_2'' \in \mathcal{D}$ such that

$[env^!_{\mathcal{M}}.(d_1 \ \mathsf{seq}\ d_2), d_1 \ \mathsf{seq}\ d_2] \xrightarrow{s_1} [\beta, d_1' \ \mathsf{seq}\ d_2'] \xrightarrow{e_2} [\beta', d_1' \ \mathsf{seq}\ d_2'']$

PROOF: $e_2 \in ev.d_2$ and $\langle 3 \rangle 6$.

$\langle 3 \rangle 8$. $l.e_1 = l.e_2 = l$

PROOF: $\langle 2 \rangle 2$, defs. of $e.\_$ and $l.\_$.

$\langle 3 \rangle 9$. $l \notin ll.d_1'$

$\langle 4 \rangle 1$. $l \in ll.(d_1' \ \mathsf{seq}\ d_2') \setminus ll.d_1'$

PROOF: By $\langle 3\rangle 7$ and application of the rules this is a necessary condition for $\langle 3\rangle 8$.

$\langle 4\rangle 2$. $l \in ll.(d_1' \text{ seq } d_2') \wedge l \notin ll.d_1'$

PROOF: $\langle 4\rangle 1$.

$\langle 4\rangle 3$. Q.E.D.

PROOF: $\langle 4\rangle 2$

$\langle 3\rangle 10$. $l \in ll.d_1'$

$\langle 4\rangle 1$. $e_1 \in ev.d_1'$

PROOF: $\langle 2\rangle 10$, $e_1 \notin ev.s_1$ and lemma 4.

$\langle 4\rangle 2$. Q.E.D.

PROOF: $\langle 3\rangle 8$ and def. of $ll.\_$.

$\langle 2\rangle 11$. CASE: $e_1 \in ev.d_2 \wedge e_2 \in ev.d_1$

$\langle 3\rangle 1$. $e_1 \notin ev.d_1 \wedge e_2 \notin ev.d_2$

PROOF: Assumption $ev.d_1 \cap ev.d_2 = \emptyset$.

$\langle 3\rangle 2$. $t_1 = ev.d_1 Ⓢ t \wedge t_2 = ev.d_2 Ⓢ t$

PROOF: Lemma 4 and assumption $ev.d_1 \cap ev.d_2 = \emptyset$

$\langle 3\rangle 3$. $\{e_1, e_2\} Ⓢ t_1 = \{e_1, e_2\} Ⓢ (ev.d_1 Ⓢ t) = \{e_1, e_2\} \cap ev.d_1 Ⓢ t = \{e_2\} Ⓢ t = \langle e_2 \rangle$

PROOF: Lemma 4, $\langle 2\rangle 11$, $\langle 3\rangle 1$ and $\langle 3\rangle 2$.

$\langle 3\rangle 4$. $\{e_1, e_2\} Ⓢ t_2 = \{e_1, e_2\} Ⓢ (ev.d_2 Ⓢ t) = \{e_1, e_2\} \cap ev.d_2 Ⓢ t = \{e_1\} Ⓢ t = \langle e_1 \rangle$

PROOF: Lemma 4, $\langle 2\rangle 11$, $\langle 3\rangle 1$ and $\langle 3\rangle 2$.

$\langle 3\rangle 5$. $\{e_1, e_2\} Ⓢ t = \langle e_1, e_2 \rangle$

PROOF: By assumption 3, $\langle 3\rangle 3$ and $\langle 3\rangle 4$ we have that $\{e_1, e_2\} Ⓢ t \neq \langle e_2, e_1 \rangle$. By $\langle 2\rangle 10$ and lemma 5 we have that $\{e_, e_2\} \subseteq ev.t$.

$\langle 3\rangle 6$. There exist traces s_1, s_2, s_3 such that $t = s_1 \frown \langle e_1 \rangle \frown s_2 \frown \langle e_2 \rangle \frown s_3$

PROOF: $\langle 3\rangle 5$.

$\langle 3\rangle 7$. There exist $\beta, \beta' \in \mathcal{B}, d_1', d_2', d_2'' \in \mathcal{D}$ such that
$[env^!_{\mathcal{M}}.(d_1, d_2), d_1 \text{ seq } d_2] \xrightarrow{s_1} [\beta, d_1' \text{ seq } d_2'] \xrightarrow{e_1} [\beta', d_1' \text{ seq } d_2'']$

PROOF: $e_1 \in ev.d_2$ and $\langle 3\rangle 6$.

$\langle 3\rangle 8$. $l.e_1 = l.e_2 = l$

PROOF: $\langle 2\rangle 2$, defs. of $e.\_$ and $l.\_$.

$\langle 3\rangle 9$. $l \notin ll.d_1'$

$\langle 4\rangle 1$. $l \in ll.(d_1' \text{ seq } d_2') \setminus ll.d_1'$

PROOF: By $\langle 3\rangle 7$ and application of the rules this is a necessary condition for $\langle 3\rangle 8$.

$\langle 4\rangle 2$. $l \in ll.(d_1' \text{ seq } d_2') \wedge l \notin ll.d_1'$

PROOF: $\langle 4\rangle 1$.

$\langle 4\rangle 3$. Q.E.D.

PROOF: $\langle 4\rangle 2$

$\langle 3\rangle 10$. $l \in ll.d_1'$

$\langle 4\rangle 1$. $e_2 \in ev.d_1'$

PROOF: $\langle 2\rangle 11$, $e_2 \notin ev.s_1$ and lemma 4.

$\langle 4\rangle 2$. Q.E.D.

PROOF: $\langle 3\rangle 8$ and def. of $ll.\_$.

$\langle 2\rangle 12$. Q.E.D.

PROOF: All possible cases yield a contradiction, so we must have that $e.l Ⓢ t = e.l Ⓢ t_1 \frown e.l Ⓢ t_2$. Since l was arbitrary chosen, this must be true for all lifelines in $\mathcal{L}$.

⟨1⟩3. Q.E.D.
  Proof: ⟨1⟩1 and ⟨1⟩2.

□

Definition 9 *We define a function $\Omega \in \mathbb{P}(\mathcal{E}) \times \mathbb{P}(\mathcal{E}) \times \mathcal{E}^* \rightarrow \{1,2\}^*$ in the following way*

$$\Omega(E_1, E_2, \langle\rangle) \overset{\mathsf{def}}{=} \langle\rangle$$
$$\Omega(E_1, E_2, \langle e\rangle^\frown t) \overset{\mathsf{def}}{=} \begin{cases} \langle 1\rangle^\frown \Omega(E_1, E_2, t) & \textbf{if } e \in E_1 \\ \langle 2\rangle^\frown \Omega(E_1, E_2, t) & \textbf{if } e \in E_2 \\ \Omega(E_1, E_2, t) & \textbf{otherwise} \end{cases}$$

where we assume $E_1 \cap E_2 = \emptyset$.

Definition 10 *We generalize the concatenation operator $\_^\frown\_$ to range over pairs of traces:*

$$(t_1, t_2)^\frown(s_1, s_2) \overset{\mathsf{def}}{=} (t_1{}^\frown s_1, t_2{}^\frown s_2)$$

Lemma 8 $\pi_i(t_1{}^\frown s_1, t_2{}^\frown s_2) = \pi_i(t_1, t_2)^\frown \pi_i(s_1, s_2),\ i \in \{1, 2\}$

Proof of lemma 8
⟨1⟩1. $\pi_i(t_1{}^\frown s_1, t_2{}^\frown s_2) = t_i{}^\frown s_i$
  Proof: Def. of π_i.
⟨1⟩2. $\pi_i(t_1, t_2)^\frown \pi_i(s_1, s_2) = t_i{}^\frown s_i$
  Proof: Def. of π_i.
⟨1⟩3. Q.E.D.
  Proof: ⟨1⟩1 and ⟨1⟩2.

□

Definition 11 *We formally define the pair filtering operator $\_\circledT\_$ (for finite sequences):*

$$\begin{aligned} P\circledT(t_1, t_2) &\overset{\mathsf{def}}{=} P\circledT(t_1|_{min(\#t_1,\#t_2)}, t_2|_{min(\#t_1,\#t_2)}) \\ P\circledT(\langle\rangle, \langle\rangle) &\overset{\mathsf{def}}{=} (\langle\rangle, \langle\rangle) \\ (v_1, v_2) \in P \Rightarrow P\circledT(\langle v_1\rangle^\frown t_1, \langle v_2\rangle^\frown t_2) &\overset{\mathsf{def}}{=} (\langle v_1\rangle, \langle v_2\rangle)^\frown P\circledT(t_1, t_2) \\ (v_1, v_2) \notin P \Rightarrow P\circledT(\langle v_1\rangle^\frown t_1, \langle v_2\rangle^\frown t_2) &\overset{\mathsf{def}}{=} P\circledT(t_1, t_2) \end{aligned}$$

where P is a set of pairs ($P \subseteq A \times B$, $t_1 \in A^*$, $t_2 \in B^*$).

Lemma 9 *Given simple diagrams $d_1, d_2 \in \mathcal{D}$. For all traces t, if there exists $\beta \in \mathcal{B}$ such that*

$$[env^!_{\mathcal{M}}.(d_1 \ \mathsf{par}\ d_2), d_1 \ \mathsf{par}\ d_2] \xrightarrow{t} [\beta, \mathsf{skip}]$$

then there exist traces t_1, t_2 and $\beta_1, \beta_2 \in \mathcal{B}$ such that

$$[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \mathsf{skip}] \wedge \pi_2((\{1\} \times \mathcal{E})\circledT(\Omega(ev.d_1, ev.d_2, t), t)) = t_1\ \wedge$$
$$[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \mathsf{skip}] \wedge \pi_2((\{2\} \times \mathcal{E})\circledT(\Omega(ev.d_1, ev.d_2, t), t)) = t_2$$

Proof of lemma 9

ASSUME: There exists $\beta \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.(d_1 \ \mathsf{par}\ d_2), d_1 \ \mathsf{par}\ d_2] \xrightarrow{t} [\beta, \mathsf{skip}]$

PROVE: There exist traces t_1, t_2, and $\beta_1, \beta_2 \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \mathsf{skip}] \wedge$
$[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \mathsf{skip}] \wedge$
$\pi_2((\{1\} \times \mathcal{E}) Ⓣ (\Omega(ev.d_1, ev.d_2, t), t)) = t_1 \wedge$
$\pi_2((\{2\} \times \mathcal{E}) Ⓣ (\Omega(ev.d_1, ev.d_2, t), t)) = t_2 \wedge$

⟨1⟩1. LET: $t = \langle e \rangle ^\frown t'$

⟨1⟩2. There exist $\beta_1', \beta_2' \in \mathcal{B}, d_1', d_2' \in \mathcal{D}$ such that
$([env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{e} [\beta_1', d_1'] \wedge$
$\pi_2((\{1\} \times \mathcal{E}) Ⓣ (\Omega(ev.d_1, ev.d_2, \langle e \rangle ^\frown t'), \langle e \rangle ^\frown t')) =$
$\langle e \rangle ^\frown \pi_2((\{1\} \times \mathcal{E}) Ⓣ (\Omega(ev.d_1, ev.d_2, t'), t'))) \veebar$
$([env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{e} [\beta_2', d_2'] \wedge$
$\pi_2((\{2\} \times \mathcal{E}) Ⓣ (\Omega(ev.d_1, ev.d_2, \langle e \rangle ^\frown t'), \langle e \rangle ^\frown t')) =$
$\langle e \rangle ^\frown \pi_2((\{2\} \times \mathcal{E}) Ⓣ (\Omega(ev.d_1, ev.d_2, t'), t')))$

⟨2⟩1. There exist $\beta' \in \mathcal{B}, d' \in \mathcal{D}$ such that
$[env^!_{\mathcal{M}}.(d_1 \ \mathsf{par}\ d_2), d_1 \ \mathsf{par}\ d_2] \xrightarrow{e} [\beta', d'] \xrightarrow{t'} [\beta, \mathsf{skip}]$
PROOF: Assumption and ⟨1⟩1

⟨2⟩2. $[env^!_{\mathcal{M}}.(d_1 \ \mathsf{par}\ d_2), d_1 \ \mathsf{par}\ d_2] \xrightarrow{e} [\beta', d_1' \ \mathsf{par}\ d_2] \veebar$
$[env^!_{\mathcal{M}}.(d_1 \ \mathsf{par}\ d_2), d_1 \ \mathsf{par}\ d_2] \xrightarrow{e} [\beta', d_1 \ \mathsf{par}\ d_2']$
PROOF: ⟨2⟩1, rules (3), (11), (12), assumption that $ev.d_1 \cap ev.d_2 = \emptyset$, and lemma 4.

⟨2⟩3. CASE: $[env^!_{\mathcal{M}}.(d_1 \ \mathsf{par}\ d_2), d_1 \ \mathsf{par}\ d_2] \xrightarrow{e} [\beta', d_1' \ \mathsf{par}\ d_2]$

⟨3⟩1. There exists $\beta_1' \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{e} [\beta_1', d_1']$
PROOF: Identical to proof of lemma 7.

⟨3⟩2. $[env^!_{\mathcal{M}}.d_2, d_2] \overset{e}{\not\rightarrow}$
PROOF: Identical to proof of lemma 7.

⟨3⟩3. $\pi_2((\{1\} \times \mathcal{E}) Ⓣ (\Omega(ev.d_1, ev.d_2, \langle e \rangle ^\frown t'), \langle e \rangle ^\frown t'))$
$= \langle e \rangle ^\frown \pi_2((\{1\} \times \mathcal{E}) Ⓣ (\Omega(ev.d_1, ev.d_2, t'), t'))$

⟨4⟩1. $e \in ev.d_1$
PROOF: ⟨3⟩1 and lemma 4.

⟨4⟩2. $\Omega(ev.d_1, ev.d_2, \langle e \rangle ^\frown t') = \langle 1 \rangle ^\frown \Omega(ev.d_1, ev.d_2, t')$
PROOF: ⟨4⟩1, def. 9 of Ω and the assumption that $ev.d_1 \cap ev.d_2 = \emptyset$.

⟨4⟩3. $\pi_2((\{1\} \times \mathcal{E}) Ⓣ (\Omega(ev.d_1, ev.d_2, \langle e \rangle ^\frown t'), \langle e \rangle ^\frown t'))$

$$\begin{array}{rll} = & \pi_2((\{1\} \times \mathcal{E}) & \\ & Ⓣ (\langle 1 \rangle ^\frown \Omega(ev.d_1, ev.d_2, t'), \langle e \rangle ^\frown t')) & (⟨4⟩2) \\ = & \pi_2((\langle 1 \rangle, \langle e \rangle) & \\ & ^\frown ((\{1\} \times \mathcal{E}) Ⓣ (\Omega(ev.d_1, ev.d_2, t'), t'))) & (\text{Def. 11 of } Ⓣ) \\ = & \pi_2((\langle 1 \rangle, \langle e \rangle)) & \\ & ^\frown \pi_2((\{1\} \times \mathcal{E}) Ⓣ (\Omega(ev.d_1, ev.d_2, t'), t')) & (\text{Lemma 8}) \\ = & \langle e \rangle ^\frown \pi_2((\{1\} \times \mathcal{E}) Ⓣ (\Omega(ev.d_1, ev.d_2, t'), t')) & (\text{Def. of } \pi_2) \end{array}$$

⟨4⟩4. Q.E.D.
PROOF: ⟨4⟩2 and ⟨4⟩3.

⟨3⟩4. Q.E.D.
  PROOF: ⟨3⟩1, ⟨3⟩2 and ⟨3⟩3.
⟨2⟩4. CASE: $[env^!_{\mathcal{M}}.(d_1 \textsf{ par } d_2), d_1 \textsf{ par } d_2] \xrightarrow{e} [\beta'', d_1 \textsf{ par } d_2']$
  PROOF: Identical to proof of ⟨2⟩3.
⟨2⟩5. Q.E.D.
  PROOF: ⟨2⟩2, ⟨2⟩3 and ⟨2⟩4.
⟨1⟩3. Q.E.D.
PROOF: We now have that $[\beta', d'] \xrightarrow{t'} [\beta, \textsf{skip}]$ where $d' = d_1' \textsf{ par } d_2$ or $d' = d_1 \textsf{ par } d_2'$. By substituting $[env^!_{\mathcal{M}}.d', d'] \xrightarrow{t'} [\beta, \textsf{skip}]$ for the assumption we may repeat the argument until $t' = \langle\rangle$, and we get what we wanted to prove. The justification of the substitution is that if $e = (?, m)$ then $\beta' = env^!_{\mathcal{M}}.d \setminus \{m\}$, but we know that $e \notin ev.d'$. If $e = (!, m)$, then $\beta' = env^!_{\mathcal{M}}.d \cup \{m\}$, and $env^!_{\mathcal{M}}.d' = env^!_{\mathcal{M}}.d \cup \{m\} \Leftrightarrow (?, m) \in ev.d'$ and $env^!_{\mathcal{M}}.d' = env^!_{\mathcal{M}}.d \Leftrightarrow (?, m) \notin ev.d$.

□

Theorem 1 (Soundness) *Given a simple diagram $d \in \mathcal{D}$. Then, for all traces t, if there exists $\beta \in \mathcal{B}$ such that*

$$[env^!_{\mathcal{M}}.d, d] \xrightarrow{t} [\beta, \textsf{skip}]$$

then

$$t \in [\![\, d \,]\!]$$

Proof of theorem 1
ASSUME: There exists $\beta \in \mathcal{B}$ such that
  $[env^!_{\mathcal{M}}.d, d] \xrightarrow{t} [\beta, \textsf{skip}]$
PROVE: $t \in [\![\, d \,]\!]$
PROOF: by induction on the structure of d
⟨1⟩1. CASE: $d = \textsf{skip}$ (induction start: empty diagram)
  ASSUME: There exists $\beta \in \mathcal{B}$ such that
    $[env^!_{\mathcal{M}}.\textsf{skip}, \textsf{skip}] \xrightarrow{t} [\beta, \textsf{skip}]$
  PROVE: $t \in [\![\, \textsf{skip} \,]\!]$
  ⟨2⟩1. $t = \langle\rangle$
    PROOF: Rule (7)
  ⟨2⟩2. $\langle\rangle \in [\![\, \textsf{skip} \,]\!]$
    ⟨3⟩1. $[\![\, \textsf{skip} \,]\!] = \{\langle\rangle\}$
      PROOF: Definition (32).
    ⟨3⟩2. Q.E.D.
      PROOF: $\langle\rangle \in \{\langle\rangle\}$
  ⟨2⟩3. Q.E.D.
    PROOF: ⟨2⟩1 and ⟨2⟩2.
⟨1⟩2. CASE: $d = (!, m)$ (induction start: single transmit event)
  ASSUME: There exists $\beta \in \mathcal{B}$ such that
    $[env^!_{\mathcal{M}}.(!, m), (!, m)] \xrightarrow{t} [\beta, \textsf{skip}]$
  PROVE: $t \in [\![\, (!, m) \,]\!]$
  ⟨2⟩1. $t = \langle(!, m)\rangle$
    ⟨3⟩1. $[env^!_{\mathcal{M}}.(!, m), (!, m)] \xrightarrow{(!,m)} [update(env^!_{\mathcal{M}}.(!, m), m.(!, m)), \textsf{skip}]$

$\langle 4\rangle 1.$ $\Pi(ll.(!,m), env^!_{\mathcal{M}}.(!,m), (!,m))$
$\xrightarrow{(!,m)} \Pi(ll.(!,m), env^!_{\mathcal{M}}.(!,m), \mathsf{skip})$

$\langle 5\rangle 1.$ $ll.(!,m) = \{l.(!,m)\}$
PROOF: Def. of $ll.\_$

$\langle 5\rangle 2.$ $l.(!,m) \in ll.(!,m)$
PROOF: $\langle 5\rangle 1$

$\langle 5\rangle 3.$ $k.(!,m) =!$
PROOF: Def. of $k.\_$

$\langle 5\rangle 4.$ Q.E.D.
PROOF: $\langle 5\rangle 2$, $\langle 5\rangle 3$ and rule (8)

$\langle 4\rangle 2.$ Q.E.D.
PROOF: $\langle 4\rangle 1$ and rule (3)

$\langle 3\rangle 2.$ Q.E.D.
PROOF: $\langle 3\rangle 1$ and lemma 5

$\langle 2\rangle 2.$ $\langle(!,m)\rangle \in [\![(!,m)]\!]$

$\langle 3\rangle 1.$ $[\![(!,m)]\!] = \{\langle(!,m)\rangle\}$
PROOF: Def. (33).

$\langle 3\rangle 2.$ Q.E.D.
PROOF: $\langle(!,m)\rangle \in \{\langle(!,m)\rangle\}$.

$\langle 2\rangle 3.$ Q.E.D.
PROOF: $\langle 2\rangle 1$ and $\langle 2\rangle 2$.

$\langle 1\rangle 3.$ CASE: $d = (?,m)$ (induction start: single receive event)
ASSUME: There exists $\beta \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.(?,m), (?,m)] \xrightarrow{t} [\beta, \mathsf{skip}]$
PROVE: $t \in [\![(?,m)]\!]$

$\langle 2\rangle 1.$ $t = \langle(?,m)\rangle$

$\langle 3\rangle 1.$ $env^!_{\mathcal{M}}.(?,m) = \{m\}$

$\langle 4\rangle 1.$ $env.(?,m) = \{(!,m)\}$
PROOF: Def. 5

$\langle 4\rangle 2.$ Q.E.D.
PROOF: $\langle 4\rangle 1$ and def. 5

$\langle 3\rangle 2.$ $[\{m\}, (?,m)] \xrightarrow{(?,m)} [update(\{m\}, m.(?,m)), \mathsf{skip}]$

$\langle 4\rangle 1.$ $\Pi(ll.(?,m), \{m\}, (?,m)) \xrightarrow{(?,m)} \Pi(ll.(?,m), \{m\}, \mathsf{skip})$

$\langle 5\rangle 1.$ $ll.(?,m) = \{l.(?,m)\}$
PROOF: Def. of $ll.\_$

$\langle 5\rangle 2.$ $l.(?,m) \in ll.(?,m)$
PROOF: $\langle 5\rangle 1$

$\langle 5\rangle 3.$ $ready(\{m\}, m.(?,m))$
PROOF: $ready(\{m\}, m.(?,m)) = m \in \{m\} = true$ (def. 3 and def. of $m.\_$)

$\langle 5\rangle 4.$ Q.E.D.
PROOF: $\langle 5\rangle 2$, $\langle 5\rangle 3$ and rule (8)

$\langle 4\rangle 2.$ Q.E.D.
PROOF: $\langle 4\rangle 1$ and rule (3)

$\langle 3\rangle 3.$ Q.E.D.
PROOF: $\langle 3\rangle 1$, $\langle 3\rangle 2$ and lemma 5

$\langle 2\rangle 2.$ $\langle(?,m)\rangle \in [\![(?,m)]\!]$

$\langle 3\rangle 1.$ $[\![(?,m)]\!] = \{\langle(?,m)\rangle\}$

PROOF: Def. (33).

$\langle 3\rangle 2.$ Q.E.D.

PROOF: $\langle(?, m)\rangle \in \{\langle(?, m)\rangle\}$.

$\langle 2\rangle 3.$ Q.E.D.

PROOF: $\langle 2\rangle 1$ and $\langle 2\rangle 2$.

$\langle 1\rangle 4.$ CASE: $d = d_1$ seq d_2 (induction step)

ASSUME: 1. There exists $\beta \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.(d_1 \mathsf{~seq~} d_2), d_1 \mathsf{~seq~} d_2] \xrightarrow{t} [\beta, \mathsf{skip}]$

2. For all traces t_1, if there exists $\beta_1 \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \mathsf{skip}]$,
then $t_1 \in [\![d_1]\!]$ (induction hypothesis)

3. For all traces t_2, if there exists $\beta_2 \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \mathsf{skip}]$,
then $t_2 \in [\![d_2]\!]$ (induction hypothesis)

PROVE: $t \in [\![d_1 \mathsf{~seq~} d_2]\!]$

$\langle 2\rangle 1.$ $[\![d_1 \mathsf{~seq~} d_2]\!] = \{h \in \mathcal{H} | \exists h_1 \in [\![d_1]\!], h_2 \in [\![d_2]\!] :$
$\forall l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2\}$

PROOF: Assumption that d is simple, and defs. (34), (35) and (36).

$\langle 2\rangle 2.$ $t \in \mathcal{H}$

PROOF: Assumption 1 and lemma 6.

$\langle 2\rangle 3.$ $\exists h_1 \in [\![d_1]\!], h_2 \in [\![d_2]\!] : \forall l \in \mathcal{L} : e.l \circledS t = e.l \circledS h_1 \frown e.l \circledS h_2$

$\langle 3\rangle 1.$ There exist traces s_1, s_2, and $\beta_1, \beta_2 \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{s_1} [\beta_1, \mathsf{skip}] \wedge$
$[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{s_2} [\beta_2, \mathsf{skip}] \wedge$
$\forall l \in \mathcal{L} : e.l \circledS t = e.l \circledS s_1 \frown e.l \circledS s_2$

PROOF: Assumption 1 and lemma 7.

$\langle 3\rangle 2.$ $s_1 \in [\![d_1]\!], s_2 \in [\![d_2]\!]$

PROOF: $\langle 3\rangle 1$and assumptions 2 and 3 (induction hypothesis).

$\langle 3\rangle 3.$ Q.E.D.

PROOF: $\langle 3\rangle 1$ and $\langle 3\rangle 2$

$\langle 2\rangle 4.$ Q.E.D.

PROOF: $\langle 2\rangle 1$, $\langle 2\rangle 2$ and $\langle 2\rangle 3$.

$\langle 1\rangle 5.$ CASE: $d = d_1$ par d_2 (induction step)

ASSUME: 1. There exists $\beta \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.(d_1 \mathsf{~par~} d_2), d_1 \mathsf{~par~} d_2] \xrightarrow{t} [\beta, \mathsf{skip}]$

2. For all traces t_1, if there exists $\beta_1 \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \mathsf{skip}]$,
then $t_1 \in [\![d_1]\!]$ (induction hypothesis)

3. For all traces t_2, if there exists $\beta_2 \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \mathsf{skip}]$,
then $t_2 \in [\![d_2]\!]$ (induction hypothesis)

PROVE: $t \in [\![d_1 \mathsf{~par~} d_2]\!]$

$\langle 2\rangle 1.$ $[\![d_1 \mathsf{~par~} d_2]\!] = \{h \in \mathcal{H} | \exists p \in \{1, 2\}^\infty :$
$\pi_2((\{1\} \times \mathcal{E}) \circledT (p, h)) \in [\![d_1]\!] \wedge$
$\pi_2((\{2\} \times \mathcal{E}) \circledT (p, h)) \in [\![d_2]\!] \}$

PROOF: Assumption that d is simple, and defs. (37), (38) and (39).

$\langle 2\rangle 2.$ $t \in \mathcal{H}$

PROOF: Assumption 1 and lemma 6.

$\langle 2\rangle 3.$ $\exists p \in \{1,2\}^\infty : \pi_2((\{1\} \times \mathcal{E}) ⓣ (p,t)) \in [\![\, d_1 \,]\!] \wedge$
$\pi_2((\{2\} \times \mathcal{E}) ⓣ (p,t)) \in [\![\, d_2 \,]\!]$

$\langle 3\rangle 1.$ There exist traces s_1, s_2, and $\beta_1, \beta_2 \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{s_1} [\beta_1, \mathsf{skip}] \wedge$
$[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{s_2} [\beta_2, \mathsf{skip}] \wedge$
$\pi_2((\{1\} \times \mathcal{E}) ⓣ (\Omega(ev.d_1, ev.d_2, t), t)) = s_1 \wedge$
$\pi_2((\{2\} \times \mathcal{E}) ⓣ (\Omega(ev.d_1, ev.d_2, t), t)) = s_2$

PROOF: Assumption 1 and lemma 9.

$\langle 3\rangle 2.$ $s_1 \in [\![\, d_1 \,]\!], s_2 \in [\![\, d_2 \,]\!]$

PROOF: $\langle 3\rangle 1$ and assumptions 2 and 3 (induction hypothesis).

$\langle 3\rangle 3.$ LET: $p = \Omega(ev.d_1, ev.d_2, t)^\frown \{1\}^\infty$

$\langle 3\rangle 4.$ Q.E.D.

PROOF: $\langle 3\rangle 1$, $\langle 3\rangle 2$, $\langle 3\rangle 3$ and def. of _ⓣ_.

$\langle 2\rangle 4.$ Q.E.D.

PROOF: $\langle 2\rangle 1$, $\langle 2\rangle 2$ and $\langle 2\rangle 3$.

$\langle 1\rangle 6.$ Q.E.D.

PROOF: $\langle 1\rangle 1$, $\langle 1\rangle 2$, $\langle 1\rangle 3$, $\langle 1\rangle 4$ and $\langle 1\rangle 5$

$\square$

Lemma 10 *Given $t \in \mathcal{E}^*$. Then $ev.t = \bigcup_{l \in \mathcal{L}} ev.(e.l ⓢ t)$.*

Proof of lemma 10

ASSUME: $t \in \mathcal{E}^*$

PROVE: $ev.t = \bigcup_{l \in \mathcal{L}} ev.(e.l ⓢ t)$

PROOF: by induction on t

$\langle 1\rangle 1.$ Induction start: $t = \langle\rangle$

$\langle 2\rangle 1.$ $ev.t = ev.\langle\rangle = \emptyset$

PROOF: $\langle 1\rangle 1$ and def. of $ev.\_$.

$\langle 2\rangle 2.$ $\bigcup_{l \in \mathcal{L}} ev.(e.l ⓢ t)$

$$\begin{aligned} &= \bigcup_{l \in \mathcal{L}} ev.(e.l ⓢ \langle\rangle) && (\langle 1\rangle 1) \\ &= \bigcup_{l \in \mathcal{L}} ev.\langle\rangle && (\text{def. of } \_ⓢ\_) \\ &= \bigcup_{l \in \mathcal{L}} \emptyset && (\text{def. of } ev.\_) \\ &= \emptyset \end{aligned}$$

$\langle 2\rangle 3.$ Q.E.D.

PROOF: $\langle 2\rangle 1$ and $\langle 2\rangle 2$.

$\langle 1\rangle 2.$ Induction step

ASSUME: $ev.t = \bigcup_{l \in \mathcal{L}} ev.(e.l ⓢ t)$ (induction hypothesis)

PROVE: $ev.(\langle e\rangle^\frown t) = \bigcup_{l \in \mathcal{L}} ev.(e.l ⓢ (\langle e\rangle^\frown t))$

$\langle 2\rangle 1.$ ASSUME: 1. $\mathcal{L} = \{l_1, l_2, \ldots, l_i, \ldots, l_k, \ldots\}$
2. $l.e = l_i$

$\langle 2\rangle 2.$ $e \in e.l_i$

PROOF: Assumption 2 and defs. of $l.\_$ and $e.\_$.

$\langle 2\rangle 3.$ $i \neq j \Rightarrow e \notin e.l_j$

PROOF: Assumption 2 and defs. of $l.\_$ and $e.\_$.

$\langle 2\rangle 4.$ $ev.(\langle e\rangle^\frown t) = \{e\} \cup ev.t$

PROOF: Def. of $ev.\_$.

$\langle 2\rangle 5.$ $\bigcup_{l \in \mathcal{L}} ev.(e.l ⓢ (\langle e\rangle^\frown t))$

$$
\begin{array}{rll}
= & ev.(e.l_1 \circledS (\langle e \rangle ^\frown t)) \cup ev.(e.l_2 \circledS (\langle e \rangle ^\frown t)) \cup \cdots & \\
& \cup\ ev.(e.l_i \circledS (\langle e \rangle ^\frown t)) \cup \cdots & \\
& \cup\ ev.(e.l_k \circledS (\langle e \rangle ^\frown t)) \cup \cdots & \text{(Assumption 1)} \\
= & ev.(e.l_1 \circledS t) \cup ev.(e.l_2 \circledS t) \cup \cdots & (\langle 2 \rangle 2,\ \langle 2 \rangle 3 \text{ and} \\
& \cup\ ev.(\langle e \rangle ^\frown e.l_i \circledS t) \cup \cdots \cup ev.(e.l_k \circledS t) \cup \cdots & \text{def. of } \_\circledS\_) \\
= & ev.(e.l_1 \circledS t) \cup ev.(e.l_2 \circledS t) \cup \cdots & \\
& \cup\ \{e\} \cup ev.(e.l_i \circledS t) \cup \cdots \cup ev.(e.l_k \circledS t) \cup \cdots & \text{(Def. of } ev.\_) \\
= & \{e\} \cup ev.(e.l_1 \circledS t) \cup ev.(e.l_2 \circledS t) \cup \cdots & \\
& \cup\ ev.(e.l_i \circledS t) \cup \cdots \cup ev.(e.l_k \circledS t) \cup \cdots & \text{(Prop. of } \cup) \\
= & \{e\} \cup \bigcup_{l \in \mathcal{L}} ev.(e.l \circledS t) & \text{(Assumption 1)} \\
= & \{e\} \cup ev.t & \text{(Ind. hyp.)}
\end{array}
$$

$\langle 2 \rangle 6$. Q.E.D.

Proof: $\langle 2 \rangle 4$ and $\langle 2 \rangle 5$.

$\langle 1 \rangle 3$. Q.E.D.

Proof: $\langle 1 \rangle 1$ and $\langle 1 \rangle 2$.

□

Lemma 11 *Given diagram* $d \in \mathcal{D}$. *If there exist* $\beta, \beta' \in \mathcal{B},\ d' \in \mathcal{D},\ e_1, e_2 \in \mathcal{E}$ *such that*

$$[\beta, d] \xrightarrow{\langle e_1, e_2 \rangle} [\beta', d']$$

and

$$l.e_1 \neq l.e_2, \quad m.e_1 \neq m.e_2$$

then

$$[\beta, d] \xrightarrow{\langle e_2, e_1 \rangle} [\beta', d']$$

Proof of lemma 11

Assume: There exist $d' \in \mathcal{D},\ \beta, \beta' \in \mathcal{B},\ e_1, e_2 \in \mathcal{E}$ such that

1. $[\beta, d] \xrightarrow{\langle e_1, e_2 \rangle} [\beta', d']$
2. $l.e_1 \neq l.e_2$
3. $m.e_1 \neq m.e_2$

Prove: $[\beta, d] \xrightarrow{\langle e_2, e_1 \rangle} [\beta', d']$

Proof: by contradiction

$\langle 1 \rangle 1$. Assume not $[\beta, d] \xrightarrow{\langle e_2, e_1 \rangle} [\beta', d']$. Then there must exist $\beta'' \in \mathcal{B},\ d'' \in \mathcal{D}$ such that $[\beta, d] \xrightarrow{e_1} [\beta'', d''] \xrightarrow{e_2} [\beta', d']$ and something in $[\beta, d]$, that is not present in $[\beta'', d'']$, prevents e_2 from being enabled (i.e. $[\beta, d] \overset{e_2}{\not\rightarrow}$). There are two ways in which this can be the case.

$\langle 2 \rangle 1$. Case: There exist $d_1, d_1', d_2 \in \mathcal{D}$ such that $d = d_1 \textsf{ seq } d_2$, $d'' = d_1' \textsf{ seq } d_2$, $e_2 \in ev.d_2$, $l.e_2 \in ll.d_1$ and $l.e_2 \notin ll.d_1'$.

This implies that $e_1 \in ev.d_1$ and $l.e_1 = l.e_2$ (because only e has been removed from the diagram), which is impossible because of assumption 2.

$\langle 2 \rangle 2$. Case: $k.e_2 = ?$, $ready(\beta, m.e_2) = \textbf{false}$ and $ready(\beta'', m.e_2) = \textbf{true}$.

This implies that $\beta'' = add(\beta, m.e_2)$ which again implies that $k.e_1 = !$ and $m.e_1 = m.e_2$. But this is impossible because of assumption 3

$\langle 1 \rangle 2$. Q.E.D.

Proof: Because the assumption of $\langle 1 \rangle 1$ leads to contradiction we must have that $[\beta, d] \xrightarrow{\langle e_2, e_1 \rangle} [\beta', d']$ is possible.

$\square$

Lemma 12 *For all traces t, if there exit traces t_1, t_2, and $\beta_1, \beta_2 \in \mathcal{B}, d_1, d_2 \in \mathcal{D}$ such that*

$$\begin{aligned}&[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \mathsf{skip}] \wedge\\ &[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \mathsf{skip}] \wedge\\ &\forall l \in \mathcal{L} : e.l \circledS t = e.l \circledS t_1 {}^\frown e.l \circledS t_2 \wedge\\ &t \in \mathcal{H}\end{aligned}$$

then there exists $\beta \in \mathcal{B}$ such that

$$[env^!_{\mathcal{M}}.(d_1 \ \mathsf{seq}\ d_2), d_1 \ \mathsf{seq}\ d_2] \xrightarrow{t} [\beta, \mathsf{skip}]$$

Proof of lemma 12

ASSUME: t is given and there exit traces t_1, t_2, and $\beta_1, \beta_2 \in \mathcal{B}, d_1, d_2 \in \mathcal{D}$ such that

1. $[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \mathsf{skip}]$
2. $[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \mathsf{skip}]$
3. $\forall l \in \mathcal{L} : e.l \circledS t = e.l \circledS t_1 {}^\frown e.l \circledS t_2$
4. $t \in \mathcal{H}$

PROVE: There exists $\beta \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.(d_1 \ \mathsf{seq}\ d_2), d_1 \ \mathsf{seq}\ d_2] \xrightarrow{t} [\beta, \mathsf{skip}]$

$\langle 1 \rangle 1.$ $ev.t = ev.t_1 \cup ev.t_2$

PROOF:
$$\begin{aligned} ev.t &= \textstyle\bigcup_{l \in \mathcal{L}} ev.(e.l \circledS t) && \text{(Lemma 10)}\\ &= \textstyle\bigcup_{l \in \mathcal{L}} ev.(e.l \circledS t_1 {}^\frown e.l \circledS t_2) && \text{(Assumption 3)}\\ &= \textstyle\bigcup_{l \in \mathcal{L}} (ev.(e.l \circledS t_1) \cup ev.(e.l \circledS t_2)) && \text{(Lemma 3)}\\ &= \textstyle\bigcup_{l \in \mathcal{L}} ev.(e.l \circledS t_1) \cup \bigcup_{l \in \mathcal{L}} ev.(e.l \circledS t_2) && \text{(Properties of } \cup)\\ &= ev.t_1 \cup ev.t_2 && \text{(Lemma 10)}\end{aligned}$$

$\langle 1 \rangle 2.$ $t_1 = ev.d_1 \circledS t$

$\langle 2 \rangle 1.$ $ev.t = ev.d_1 \cup ev.d_2$, $ev.d_1 = ev.t_1$

PROOF: $\langle 1 \rangle 1$, assumptions 1 and 2, and lemma 5.

$\langle 2 \rangle 2.$ LET: $ev.d_1 \circledS t = t_1'$

ASSUME: $t_1 \neq t_1'$

$\langle 2 \rangle 3.$ $ev.d_1 = ev.t_1 = ev.t_1'$

PROOF: $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$.

$\langle 2 \rangle 4.$ There exist $e_1, e_2 \in ev.d_1$, and traces $s_1, s_2, s_2', s_3, s_3'$ such that

$t_1 = s_1 {}^\frown \langle e_1 \rangle {}^\frown s_2 {}^\frown \langle e_2 \rangle {}^\frown s_3$

$t_1' = s_1 {}^\frown s_2' {}^\frown \langle e_2, e_1 \rangle {}^\frown s_3'$

PROOF: By $\langle 2 \rangle 2$, t_1 and t_1' are unequal. By $\langle 2 \rangle 3$ and the assumption that d has no repetition of events, we do not have the case that t_1 is a proper prefix of t_1' or that t_1' is a proper prefix of t_1. Let s_1 be the longest trace such that $s_1 \sqsubseteq t_1$ and $s_1 \sqsubseteq t_1'$ and let e_1 be the first element of t_1 after s_1 ($e_1 = t_1[\#s_1 + 1]$). Then we must have that $e_1 \neq t_1'[\#s_1 + 1]$, and, by $\langle 2 \rangle 3$, that $e_1 = t_1'[j]$ for some $j > \#s_1 + 1$. Let $e_2 = t_1'[j-1]$. By $\langle 2 \rangle 3$, the assumption that there are no repetition of events and the fact that t_1 and t_1' is equal up to s_1, we must have that $e_2 = t_1[k]$ for some $k > \#s_1 + 1$.

$\langle 2 \rangle 5.$ There exist $e_1, e_2 \in ev.d_1$ such that

$\{e_1, e_2\} \circledS t_1 = \langle e_1, e_2 \rangle$ and

$\{e_1, e_2\} \circledS t_1' = \langle e_2, e_1 \rangle$.

PROOF: ⟨2⟩4.

⟨2⟩6. $\{e_1, e_2\} \circledS t = \langle e_2, e_1 \rangle$

PROOF: ⟨2⟩2 and ⟨2⟩5.

⟨2⟩7. $l.e_1 \neq l.e_2$

PROOF: If there exists l such that $l.e_1 = l.e_2 = l$, then $e_1, e_2 \in e.l$ which, by ⟨2⟩5 and ⟨2⟩6, imply that $(\{e_1, e_2\} \cap e.l) \circledS t \neq (\{e_1, e_2\} \cap e.l) \circledS t_1$. But this contradicts assumption 3.

⟨2⟩8. $m.e_1 \neq m.e_2$

PROOF: If we assume that $m.e_1 = m.e_2$, we must have, by the assumption that there are no repetition of events, that there exists m such that $e_1 = (!, m)$ and $e_2 = (?, m)$ or $e_1 = (?, m)$ and $e_2 = (!, m)$. The first case contradicts, by ⟨2⟩6, the assumption that $t \in \mathcal{H}$ (assumption 4), and the second case contradicts, by ⟨2⟩5, the fact that $t_1 \in \mathcal{H}$ (assumption 1 and lemma 6).

⟨2⟩9. Q.E.D.

PROOF: If we assume that $t_1 \neq ev.d_1 \circledS t$ (⟨2⟩2) we get the result that $l.e_1 \neq l.e_2$ (⟨2⟩7) and $m.e_1 \neq m.e_2$ (⟨2⟩8) for any pair of events $e_1, e_2 \in ev.d_1 = ev.t_1$ such that $\{e_1, e_2\} \circledS t_1 \neq (\{e_1, e_2\} \cap ev.d_1) \circledS t$. By lemma 11, this means that the order of e_1 and e_2 in t_1' is arbitrary and we may swap their position without this affecting assumption 1. We may repeat this argument until $t_1'[\#s_1 + 1] = e_1$. We may then substitute $s_1 \frown \langle e_1 \rangle$ for s_1 in ⟨2⟩4 and repeat the argument until $t_1 = t_1' = s_1$. For this reason we can assume ⟨1⟩2 without this affecting the rest of the proof.

⟨1⟩3. $t_2 = ev.d_2 \circledS t$

PROOF: Identical to proof of ⟨1⟩2

⟨1⟩4. We may now describe a sequence of transitions such that

$$[env^!_{\mathcal{M}}.(d_1 \ \mathsf{seq} \ d_2), d_1 \ \mathsf{seq} \ d_2] \xrightarrow{t} [\beta, \mathsf{skip}]$$

⟨2⟩1. LET: $t = \langle e \rangle \frown t'$

⟨2⟩2. We have that either

there exists t_1' such that $t_1 = \langle e \rangle \frown t_1'$ and $e \notin ev.t_2$, or

there exists t_2' such that $t_2 = \langle e \rangle \frown t_2'$ and $e \notin ev.t_1$.

PROOF: ⟨1⟩1, ⟨1⟩2, ⟨1⟩3, the overall assumption that $ev.d_1 \cap ev.d_2 = \emptyset$, and the facts that $ev.d_1 = ev.t_1$ and $ev.d_2 = ev.t_2$ (assumptions 1 and 2, and lemma 5.

⟨2⟩3. CASE: $t_1 = \langle e \rangle \frown t_1'$ and $e \notin ev.t_2$

PROVE: There exists $d_1' \in \mathcal{D}$ and $\beta' \in \mathcal{B}$ such that $[env^!_{\mathcal{M}}.(d_1 \ \mathsf{seq} \ d_2), d_1 \ \mathsf{seq} \ d_2] \xrightarrow{e} [\beta', d_1' \ \mathsf{seq} \ d_2]$

⟨3⟩1. There exists $d_1' \in \mathcal{D}$ such that

$$\Pi(ll.d_1, env^!_{\mathcal{M}}.d_1, d_1) \xrightarrow{e} \Pi(ll.d_1, env^!_{\mathcal{M}}.d_1, d_1')$$

PROOF: Case assumption ⟨2⟩3, assumption 1 and rule (3).

⟨3⟩2. There exists $d_1' \in \mathcal{D}$ such that

$$\Pi(ll.d_1 \cap ll.(d_1 \ \mathsf{seq} \ d_2), env^!_{\mathcal{M}}.(d_1 \ \mathsf{seq} \ d_2), d_1) \xrightarrow{e}$$
$$\Pi(ll.d_1 \cap ll.(d_1 \ \mathsf{seq} \ d_2), env^!_{\mathcal{M}}.(d_1 \ \mathsf{seq} \ d_2), d_1')$$

⟨4⟩1. $ll.d_1 \cap ll.(d_1 \ \mathsf{seq} \ d_2) = ll.d_1 \cap (ll.d_1 \cup ll.d_2) = ll.d_1$

PROOF: Def. of $ll.\_$.

⟨4⟩2. $k.e = !$ implies that the state of the communication medium is irrelevant. $k.e = ?$ implies that $m.e \in env^!_{\mathcal{M}}.(d_1 \ \mathsf{seq} \ d_2)$

PROOF: Rule (8). $k.e = ?$ implies that $m.e \in env^!_{\mathcal{M}}.d_1$ and also that $(!, m.e) \notin ev.t'$ because $t \in \mathcal{H}$ (assumption 4), which means $(!, m.e) \notin ev.d'_1 \cup ev.d_2$ ($\langle 1\rangle 1$ and lemma 5). This implies $m.e \in env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2)$ because $m.e \in env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2) \Leftrightarrow m.e \in env^!_{\mathcal{M}}.d_1 \wedge (!, m.e) \notin ev.d_2$ (see proof of lemma 7).

$\langle 4\rangle 3.$ Q.E.D.

PROOF: $\langle 3\rangle 1$, $\langle 4\rangle 1$, $\langle 4\rangle 2$.

$\langle 3\rangle 3.$ There exists $d'_1 \in \mathcal{D}$ such that

$\Pi(ll.(d_1 \textsf{ seq } d_2), env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1 \textsf{ seq } d_2) \xrightarrow{e}$
$\Pi(ll.(d_1 \textsf{ seq } d_2), env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d'_1 \textsf{ seq } d_2)$

$\langle 4\rangle 1.$ $ll.d_1 \cap ll.(d_1 \textsf{ seq } d_2) \neq \emptyset$

PROOF: $e \in ev.d_1$ and def. of $ll.\_$.

$\langle 4\rangle 2.$ Q.E.D.

PROOF: $\langle 3\rangle 2$, $\langle 4\rangle 1$ and rule (9)

$\langle 3\rangle 4.$ Q.E.D.

PROOF: $\langle 3\rangle 3$ and rule (3).

$\langle 2\rangle 4.$ CASE: $t_2 = \langle e\rangle ^\frown t'_2$ and $e \notin ev.t_1$

PROVE: There exists $d'_2 \in \mathcal{D}$ and $\beta' \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1 \textsf{ seq } d_2] \xrightarrow{e} [\beta', d_1 \textsf{ seq } d'_2]$

$\langle 3\rangle 1.$ There exists $d'_2 \in \mathcal{D}$ such that

$\Pi(ll.d_2, env^!_{\mathcal{M}}.d_2, d_2) \xrightarrow{e} \Pi(ll.d_2, env^!_{\mathcal{M}}.d_2, d'_2)$

PROOF: Case assumption $\langle 2\rangle 4$, assumption 2 and rule (3).

$\langle 3\rangle 2.$ There exists $d'_2 \in \mathcal{D}$ such that

$\Pi(ll.(d_1 \textsf{ seq } d_2) \setminus ll.d_1, env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_2) \xrightarrow{e}$
$\Pi(ll.(d_1 \textsf{ seq } d_2) \setminus ll.d_1, env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d'_2)$

$\langle 4\rangle 1.$ $l.e \in ll.(d_1 \textsf{ seq } d_2) \setminus ll.d_1$

$\langle 5\rangle 1.$ $l.e \in ll.(d_1 \textsf{ seq } d_2) \setminus ll.d_1$
$\Leftrightarrow (l.e \in ll.e_1 \vee l.e \in ll.d_2) \wedge l.e \notin ll.d_1$
$\Leftrightarrow l.e \in ll.d_2 \wedge l.e \notin ll.d_1$

PROOF: Def. of $ll.\_$ and basic set theory.

$\langle 5\rangle 2.$ $l.e \in ll.d_2$

PROOF: Assumption 2, case assumption, lemma 5 and def. of $ll.\_$.

$\langle 5\rangle 3.$ $l.e \notin ll.d_1$

PROOF: From assumption 3, $\langle 2\rangle 1$ and the case assumption we know that $e.(l.e) \circledS t_1 = \langle\rangle$. This implies that $e.(l.e) \cap ev.t_1 = e.(l.e) \cap ev.d_2 = \emptyset$, which again imply that $l.e \notin ll.d_1$ (lemma 5, defs. of $e.\_$, $l.\_$, $ev.\_$ and $ll.\_$).

$\langle 5\rangle 4.$ Q.E.D.

PROOF: $\langle 5\rangle 1$, $\langle 5\rangle 2$ and $\langle 5\rangle 3$.

$\langle 4\rangle 2.$ $k.e = !$ implies that the state of the communication medium is irrelevant. $k.e = ?$ implies that $m.e \in env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2)$

PROOF: Rule (8). $k.e = ?$ implies that $m.e \in env^!_{\mathcal{M}}.d_1$ and also that $(!, m.e) \notin ev.t'$ because $t \in \mathcal{H}$ (assumption 4), which means $(!, m.e) \notin ev.d'_1 \cup ev.d_2$ ($\langle 1\rangle 1$ and lemma 5). This implies $m.e \in env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2)$ because $m.e \in env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2) \Leftrightarrow m.e \in env^!_{\mathcal{M}}.d_1 \wedge (!, m.e) \notin ev.d_2$ (see proof of lemma 7).

$\langle 4\rangle 3.$ Q.E.D.

PROOF: $\langle 3\rangle 2$, $\langle 4\rangle 1$, $\langle 4\rangle 2$.

$\langle 3\rangle 3.$ There exists $d'_2 \in \mathcal{D}$ such that

$\Pi(ll.(d_1 \textsf{ seq } d_2), env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1 \textsf{ seq } d_2) \xrightarrow{e}$
$\Pi(ll.(d_1 \textsf{ seq } d_2), env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1 \textsf{ seq } d_2')$

⟨4⟩1. $ll.(d_1 \textsf{ seq } d_2) \setminus ll.d_1 \neq \emptyset$
PROOF: $l.e \in ll.(d_1 \textsf{ seq } d_2) \setminus ll.d_1$ (see ⟨3⟩2).

⟨4⟩2. Q.E.D.
PROOF: ⟨3⟩2, ⟨4⟩1 and rule (10)

⟨3⟩4. Q.E.D.
PROOF: ⟨3⟩3 and rule (3).

⟨2⟩5. We may now substitute t for t' and repeat the argument until $t = \langle\rangle$. (For justification of this see proof of lemma 7.)

⟨2⟩6. Q.E.D.
PROOF: ⟨2⟩1-⟨2⟩4

⟨1⟩5. Q.E.D.
PROOF: ⟨1⟩1-⟨1⟩4

□

Lemma 13 *Given traces t, t_1, t_2 and oracle $p \in \{1,2\}^\infty$. If*

$$\pi_2((\{1\} \times \mathcal{E}) \circledT (p,t)) = t_1 \ \wedge$$
$$\pi_2((\{2\} \times \mathcal{E}) \circledT (p,t)) = t_2$$

then

$$ev.t = ev.t_1 \cup ev.t_2$$

Proof of lemma 13

ASSUME: t, t_1, t_2 and $p \in \{1,2\}^\infty$ given, and

1. $\pi_2((\{1\} \times \mathcal{E}) \circledT (p,t)) = t_1$
2. $\pi_2((\{2\} \times \mathcal{E}) \circledT (p,t)) = t_2$

PROVE: $ev.t = ev.t_1 \cup ev.t_2$
PROOF: by induction on t

⟨1⟩1. Induction start: $t = \langle\rangle$
PROVE: $ev.t = ev.t_1 \cup ev.t_2$

⟨2⟩1. $ev.t = ev.\langle\rangle = \emptyset$
PROOF: ⟨1⟩1 and def. of $ev.\_$.

⟨2⟩2. $ev.t_1 \cup ev.t_2$

$$\begin{aligned}
&= ev.\pi_2((\{1\} \times \mathcal{E}) \circledT (p, \langle\rangle)) \\
&\quad \cup\, ev.\pi_2((\{2\} \times \mathcal{E}) \circledT (p, \langle\rangle)) && \text{(Assumptions 1 and 2, and ⟨1⟩1)} \\
&= ev.\pi_2((\{1\} \times \mathcal{E}) \circledT (\langle\rangle, \langle\rangle)) \\
&\quad \cup\, ev.\pi_2((\{2\} \times \mathcal{E}) \circledT (\langle\rangle, \langle\rangle)) && \text{(Def. 11 of } \_\circledT\_) \\
&= ev.\pi_2(\langle\rangle, \langle\rangle) \cup ev.\pi_2(\langle\rangle, \langle\rangle) && \text{(Def. 11 of } \_\circledT\_) \\
&= ev.\langle\rangle \cup ev.\langle\rangle && \text{(Def. of } \pi_2) \\
&= \emptyset \cup \emptyset && \text{(Def. of. } ev.\_) \\
&= \emptyset
\end{aligned}$$

⟨2⟩3. Q.E.D.
PROOF: ⟨2⟩1 and ⟨2⟩2.

⟨1⟩2. Induction step
ASSUME: $ev.t = ev.t_1 \cup ev.t_2$ (induction hypothesis)

PROVE: $ev.(t^\frown\langle e\rangle) = ev.\pi_2((\{1\}\times\mathcal{E})\textcircled{\scriptsize T}(p, t^\frown\langle e\rangle))$
$\cup\, ev.\pi_2((\{2\}\times\mathcal{E})\textcircled{\scriptsize T}(p, t^\frown\langle e\rangle))$

$\langle 2\rangle 1.$ $ev.(t^\frown\langle e\rangle)$
$= ev.t \cup ev.\langle e\rangle$ (lemma 3)
$= ev.t \cup \{e\}$ (def. of $ev.\_$)

$\langle 2\rangle 2.$ LET: $k \in \{1,2\}$

$\langle 2\rangle 3.$ $ev.\pi_2((\{k\}\times\mathcal{E})\textcircled{\scriptsize T}(p, t^\frown\langle e\rangle))$
$= ev.\pi_2((\{k\}\times\mathcal{E})\textcircled{\scriptsize T}(p|_{\#t+1}, t^\frown\langle e\rangle))$ (Def. 11 of $\_\textcircled{\scriptsize T}\_$)
$= ev.\pi_2((\{k\}\times\mathcal{E})\textcircled{\scriptsize T}(p|_{\#t}{}^\frown\langle j\rangle, t^\frown\langle e\rangle))$ ($j \in \{1,2\}$)
$= ev.\pi_2(((\{k\}\times\mathcal{E})\textcircled{\scriptsize T}(p|_{\#t}, t))$
${}^\frown((\{k\}\times\mathcal{E})\textcircled{\scriptsize T}(\langle j\rangle, \langle e\rangle)))$ (Defs. of $\_{}^\frown\_$ and $\_\textcircled{\scriptsize T}\_$)
$= ev.(\pi_2((\{k\}\times\mathcal{E})\textcircled{\scriptsize T}(p|_{\#t}, t))$
${}^\frown\pi_2((\{k\}\times\mathcal{E})\textcircled{\scriptsize T}(\langle j\rangle, \langle e\rangle)))$ (Lemma 8)
$= ev.\pi_2((\{k\}\times\mathcal{E})\textcircled{\scriptsize T}(p|_{\#t}, t))$
$\cup\, ev.\pi_2((\{k\}\times\mathcal{E})\textcircled{\scriptsize T}(\langle j\rangle, \langle e\rangle))$ (Lemma 3)
$= ev.\pi_2((\{k\}\times\mathcal{E})\textcircled{\scriptsize T}(p, t))$
$\cup\, ev.\pi_2((\{k\}\times\mathcal{E})\textcircled{\scriptsize T}(\langle j\rangle, \langle e\rangle))$ (Def. 11 of $\_\textcircled{\scriptsize T}\_$)
$= t_k \cup ev.\pi_2((\{k\}\times\mathcal{E})\textcircled{\scriptsize T}(\langle j\rangle, \langle e\rangle))$ (Assumptions 1 and 2)

$\langle 2\rangle 4.$ $ev.\pi_2((\{1\}\times\mathcal{E})\textcircled{\scriptsize T}(p, t^\frown\langle e\rangle))$
$\cup\, ev.\pi_2((\{2\}\times\mathcal{E})\textcircled{\scriptsize T}(p, t^\frown\langle e\rangle))$
$= t_1 \cup ev.\pi_2((\{1\}\times\mathcal{E})\textcircled{\scriptsize T}(\langle j\rangle, \langle e\rangle))$
$\cup\, t_2 \cup ev.\pi_2((\{2\}\times\mathcal{E})\textcircled{\scriptsize T}(\langle j\rangle, \langle e\rangle))$ ($\langle 2\rangle 3$, $j \in \{1,2\}$)
$= t \cup ev.\pi_2((\{1\}\times\mathcal{E})\textcircled{\scriptsize T}(\langle j\rangle, \langle e\rangle))$
$\cup\, ev.\pi_2((\{2\}\times\mathcal{E})\textcircled{\scriptsize T}(\langle j\rangle, \langle e\rangle))$ (induction hypothesis)

$\langle 2\rangle 5.$ CASE: $j = 1$
$t \cup ev.\pi_2((\{1\}\times\mathcal{E})\textcircled{\scriptsize T}(\langle 1\rangle, \langle e\rangle))$
$\cup\, ev.\pi_2((\{2\}\times\mathcal{E})\textcircled{\scriptsize T}(\langle 1\rangle, \langle e\rangle))$
$= t \cup ev.\pi_2(\langle 1\rangle, \langle e\rangle) \cup ev.\pi_2(\langle\rangle, \langle\rangle)$ (Def. 11 of $\_\textcircled{\scriptsize T}\_$)
$= ev.t \cup ev.\langle e\rangle \cup ev.\langle\rangle$ (Def. of π_2)
$= ev.t \cup \{e\} \cup \emptyset$ (Def. of $ev.\_$)
$= ev.t \cup \{e\}$

$\langle 2\rangle 6.$ CASE: $j = 2$
$t \cup ev.\pi_2((\{1\}\times\mathcal{E})\textcircled{\scriptsize T}(\langle 2\rangle, \langle e\rangle))$
$\cup\, ev.\pi_2((\{2\}\times\mathcal{E})\textcircled{\scriptsize T}(\langle 2\rangle, \langle e\rangle))$
$= t \cup ev.\pi_2(\langle\rangle, \langle\rangle) \cup ev.\pi_2(\langle 2\rangle, \langle e\rangle)$ (Def. 11 of $\_\textcircled{\scriptsize T}\_$)
$= ev.t \cup ev.\langle\rangle \cup ev.\langle e\rangle$ (Def. of π_2)
$= ev.t \cup \emptyset \cup \{e\}$ (Def. of $ev.\_$)
$= ev.t \cup \{e\}$

$\langle 2\rangle 7.$ Q.E.D.
PROOF: $\langle 2\rangle 1$-$\langle 2\rangle 6$

$\langle 1\rangle 3.$ Q.E.D.
PROOF: $\langle 1\rangle 1$ and $\langle 1\rangle 2$

□

Lemma 14 *For all t, if there exist traces t_1, t_2, and $\beta_1, \beta_2 \in \mathcal{B}, d_1, d_2 \in \mathcal{D}$, and*

$p \in \{1,2\}^\infty$ *such that*

$$\begin{aligned}&[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \mathsf{skip}] \wedge\\ &[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \mathsf{skip}] \wedge\\ &\pi_2((\{1\} \times \mathcal{E})Ⓣ(p,t)) = t_1 \wedge\\ &\pi_2((\{2\} \times \mathcal{E})Ⓣ(p,t)) = t_2 \wedge\\ &t \in \mathcal{H}\end{aligned}$$

then there exists $\beta \in \mathcal{B}$ *such that*

$$[env^!_{\mathcal{M}}.(d_1 \ \mathsf{par}\ d_2), d_1 \ \mathsf{par}\ d_2] \xrightarrow{t} [\beta, \mathsf{skip}]$$

Proof of lemma 14

ASSUME: t is given and there exist traces t_1, t_2, and $\beta_1, \beta_2 \in \mathcal{B}, d_1, d_2 \in \mathcal{D}$, and $p \in \{1,2\}^\infty$ such that
1. $[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \mathsf{skip}]$
2. $[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \mathsf{skip}]$
3. $\pi_2((\{1\} \times \mathcal{E})Ⓣ(p,t)) = t_1$
4. $\pi_2((\{2\} \times \mathcal{E})Ⓣ(p,t)) = t_2$
5. $t \in \mathcal{H}$

PROVE: There exists $\beta \in \mathcal{B}$ such that $[env^!_{\mathcal{M}}.(d_1 \ \mathsf{par}\ d_2), d_1 \ \mathsf{par}\ d_2] \xrightarrow{t} [\beta, \mathsf{skip}]$

$\langle 1\rangle 1.$ $ev.t = ev.t_1 \cup ev.t_2$
PROOF: Assumptions 3 and 4, and lemma 13.

$\langle 1\rangle 2.$ We describe a sequence of transitions such that
$[env^!_{\mathcal{M}}.(d_1 \ \mathsf{par}\ d_2), d_1 \ \mathsf{par}\ d_2] \xrightarrow{t} [\beta, \mathsf{skip}]$
We do this with guidance of the oracle p

$\langle 2\rangle 1.$ LET: $t = \langle e\rangle ^\frown t'$

$\langle 2\rangle 2.$ CASE: $p = \langle 1\rangle ^\frown p'$
PROVE: There exists $d_1' \in \mathcal{D}$ and $\beta' \in \mathcal{B}$ such that
$[env^!_{\mathcal{M}}.(d_1 \ \mathsf{par}\ d_2), d_1 \ \mathsf{par}\ d_2] \xrightarrow{e} [\beta', d_1' \ \mathsf{par}\ d_2]$

$\langle 3\rangle 1.$ $t_1 = \langle e\rangle ^\frown \pi_2((\{1\} \times \mathcal{E})Ⓣ(p', t'))$
PROOF:

$$\begin{aligned} t_1 &= \pi_2((\{1\} \times \mathcal{E})Ⓣ(p,t)) && \text{(Assumption 3)}\\ &= \pi_2((\{1\} \times \mathcal{E})Ⓣ(\langle 1\rangle ^\frown p', \langle e\rangle ^\frown t')) && \text{(}\langle 2\rangle 1 \text{ and } \langle 2\rangle 2\text{)}\\ &= \pi_2((\langle 1\rangle, \langle e\rangle)^\frown((\{1\} \times \mathcal{E})Ⓣ(p', t'))) && \text{(Def. 11 of \_Ⓣ\_)}\\ &= \pi_2((\langle 1\rangle, \langle e\rangle))^\frown \pi_2((\{1\} \times \mathcal{E})Ⓣ(p', t')) && \text{(Lemma 8)}\\ &= \langle e\rangle ^\frown \pi_2((\{1\} \times \mathcal{E})Ⓣ(p', t')) && \text{(Def. of } \pi_2\text{)}\end{aligned}$$

$\langle 3\rangle 2.$ There exists $d_1' \in \mathcal{D}$ such that
$\Pi(ll.d_1, env^!_{\mathcal{M}}.d_1, d_1) \xrightarrow{e} \Pi(ll.d_1, env^!_{\mathcal{M}}.d_1, d_1')$
PROOF: $\langle 3\rangle 1$, assumption 1 and rule (3).

$\langle 3\rangle 3.$ There exists $d_1' \in \mathcal{D}$ such that
$\Pi(ll.d_1 \cap ll.(d_1 \ \mathsf{par}\ d_2), env^!_{\mathcal{M}}.(d_1 \ \mathsf{par}\ d_2), d_1) \xrightarrow{e}$
$\Pi(ll.d_1 \cap ll.(d_1 \ \mathsf{par}\ d_2), env^!_{\mathcal{M}}.(d_1 \ \mathsf{par}\ d_2), d_1')$

$\langle 4\rangle 1.$ $ll.d_1 \cap ll.(d_1 \ \mathsf{par}\ d_2) = ll.d_1 \cap (ll.d_1 \cup ll.d_2) = ll.d_1$
PROOF: Def. of $ll.\_$.

$\langle 4\rangle 2.$ $k.e = !$ implies that the state of the communication medium is irrelevant. $k.e = ?$ implies that $m.e \in env^!_{\mathcal{M}}.(d_1 \ \mathsf{par}\ d_2)$

PROOF: Rule (8). $k.e = ?$ implies that $m.e \in env^!_{\mathcal{M}}.d_1$ and also that $(!, m.e) \notin ev.t'$ because $t \in \mathcal{H}$ (assumption 5), which means $(!, m.e) \notin ev.d_1' \cup ev.d_2$ ($\langle 1 \rangle 1$ and lemma 5). This implies $m.e \in env^!_{\mathcal{M}}.(d_1 \text{ par } d_2)$ because $m.e \in env^!_{\mathcal{M}}.(d_1 \text{ par } d_2) \Leftrightarrow m.e \in env^!_{\mathcal{M}}.d_1 \wedge (!, m.e) \notin ev.d_2$ (see proof of lemma 7).

$\langle 4 \rangle 3$. Q.E.D.

PROOF: $\langle 3 \rangle 2$, $\langle 4 \rangle 1$, $\langle 4 \rangle 2$.

$\langle 3 \rangle 4$. There exists $d_1' \in \mathcal{D}$ such that

$\Pi(ll.(d_1 \text{ par } d_2), env^!_{\mathcal{M}}.(d_1 \text{ par } d_2), d_1 \text{ par } d_2) \xrightarrow{e}$

$\Pi(ll.(d_1 \text{ par } d_2), env^!_{\mathcal{M}}.(d_1 \text{ par } d_2), d_1' \text{ par } d_2)$

PROOF: $\langle 3 \rangle 3$, and rule (11)

$\langle 3 \rangle 5$. Q.E.D.

PROOF: $\langle 3 \rangle 4$ and rule (3).

$\langle 2 \rangle 3$. CASE: $p = \langle 2 \rangle ^\frown p'$

PROVE: There exists $d_2' \in \mathcal{D}$ and $\beta' \in \mathcal{B}$ such that

$[env^!_{\mathcal{M}}.(d_1 \text{ par } d_2), d_1 \text{ par } d_2] \xrightarrow{e} [\beta', d_1 \text{ par } d_2']$

PROOF: Identical to proof of $\langle 2 \rangle 2$ with t_2 and rule (12) substituted for t_1 and rule (11).

$\langle 2 \rangle 4$. We may now substitute t for t' and repeat the argument until $t = \langle \rangle$. (For justification of this see proof of lemma 9.)

$\langle 2 \rangle 5$. Q.E.D.

PROOF: $\langle 2 \rangle 1$-$\langle 2 \rangle 3$

$\langle 1 \rangle 3$. Q.E.D.

PROOF: $\langle 1 \rangle 1$ and $\langle 1 \rangle 2$

□

Theorem 2 (Completeness) *Given a simple diagram d. Then, for all traces t, if*

$$t \in [\![d]\!]$$

then there exists $\beta \in \mathcal{B}$ such that

$$[env^!_{\mathcal{M}}.d, d] \xrightarrow{t} [\beta, \text{skip}]$$

Proof of theorem 2

ASSUME: $t \in [\![d]\!]$

PROVE: There exists $\beta \in \mathcal{B}$ such that

$[env^!_{\mathcal{M}}.d, d] \xrightarrow{t} [\beta, \text{skip}]$

PROOF: by induction on the structure of d.

$\langle 1 \rangle 1$. CASE: $d = \text{skip}$ (induction start: empty diagram)

ASSUME: $t \in [\![\text{skip}]\!]$

PROVE: There exists $\beta \in \mathcal{B}$ such that

$[env^!_{\mathcal{M}}.\text{skip}, \text{skip}] \xrightarrow{t} [\beta, \text{skip}]$

$\langle 2 \rangle 1$. $t = \langle \rangle$

$\langle 3 \rangle 1$. $[\![\text{skip}]\!] = \{\langle \rangle\}$

PROOF: Assumption that d is simple, and definition (32).

$\langle 3 \rangle 2$. Q.E.D.

PROOF: $\langle 3 \rangle 1$.

$\langle 2 \rangle 2$. $[env^!_{\mathcal{M}}.\text{skip}, \text{skip}] \xrightarrow{\langle \rangle} [\emptyset, \text{skip}]$

PROOF: Rule (7) and def. of $env^!_{\mathcal{M}}$.

$\langle 2\rangle 3.$ Q.E.D.

PROOF: $\langle 2\rangle 1$ and $\langle 2\rangle 2$.

$\langle 1\rangle 2.$ CASE: $d = (!, m)$ (induction start: single output)

ASSUME: $t \in [\![(!, m)]\!]$

PROVE: There exists $\beta \in \mathcal{B}$ such that $[env^!_{\mathcal{M}}.(!, m), (!, m)] \xrightarrow{t} [\beta, \mathsf{skip}]$

$\langle 2\rangle 1.$ $t = \langle (!, m)\rangle$

$\langle 3\rangle 1.$ $[\![(!, m)]\!] = \{\langle (!, m)\rangle\}$

PROOF: Assumption that d is simple and def. (33).

$\langle 3\rangle 2.$ Q.E.D.

PROOF: $\langle 3\rangle 1$

$\langle 2\rangle 2.$ $[env^!_{\mathcal{M}}.(!, m), (!, m)] \xrightarrow{(!,m)} [\{m\}, \mathsf{skip}]$

$\langle 3\rangle 1.$ $\Pi(ll.(!, m), env^!_{\mathcal{M}}.(!, m), (!, m)) \xrightarrow{(!,m)} \Pi(ll.(!, m), env^!_{\mathcal{M}}.(!, m), \mathsf{skip})$

PROOF: $l.(!, m) \in ll.(!, m) = \{l.(!, m)\}$ (def. of $ll.\_$), $k.(!, m) = !$ (def. of $k.\_$), and rule (8).

$\langle 3\rangle 2.$ Q.E.D.

PROOF: $\langle 3\rangle 1$, rule (3) and def. of $env^!_{\mathcal{M}}$.

$\langle 2\rangle 3.$ Q.E.D.

PROOF: $\langle 2\rangle 1$ and $\langle 2\rangle 2$.

$\langle 1\rangle 3.$ CASE: $d = (?, m)$ (induction start: single input)

ASSUME: $t \in [\![(?, m)]\!]$

PROVE: There exists $\beta \in \mathcal{B}$ such that $[env^!_{\mathcal{M}}.(?, m), (?, m)] \xrightarrow{t} [\beta, \mathsf{skip}]$

$\langle 2\rangle 1.$ $t = \langle (?, m)\rangle$

$\langle 3\rangle 1.$ $[\![(?, m)]\!] = \{\langle (?, m)\rangle\}$

PROOF: Assumption that d is simple and def. (33).

$\langle 3\rangle 2.$ Q.E.D.

PROOF: $\langle 3\rangle 1$

$\langle 2\rangle 2.$ $[env^!_{\mathcal{M}}.(?, m), (?, m)] \xrightarrow{(?,m)} [\emptyset, \mathsf{skip}]$

$\langle 3\rangle 1.$ $\Pi(ll.(?, m), env^!_{\mathcal{M}}.(?, m), (?, m)) \xrightarrow{(?,m)} \Pi(ll.(?, m), env^!_{\mathcal{M}}.(?, m), \mathsf{skip})$

PROOF: $l.(?, m) \in ll.(?, m) = \{l.(?, m)\}$ (def. of $ll.\_$), $ready(env^!_{\mathcal{M}}.(?, m))$, $m.(?, m) = ready(\{m\}, m) = m \in \{m\}$ (defs. of $ready$, $env^!_{\mathcal{M}}.\_$ and $m.\_$), and rule (8).

$\langle 3\rangle 2.$ Q.E.D.

PROOF: $\langle 3\rangle 1$, rule (3) and def. of $env^!_{\mathcal{M}}$.

$\langle 2\rangle 3.$ Q.E.D.

PROOF: $\langle 2\rangle 1$ and $\langle 2\rangle 2$.

$\langle 1\rangle 4.$ CASE: $d = d_1 \ \mathsf{seq}\ d_2$ (induction step)

ASSUME:
1. $t \in [\![d_1 \ \mathsf{seq}\ d_2]\!]$
2. For all t_1, if $t_1 \in [\![d_1]\!]$, then there exists $\beta_1 \in \mathcal{B}$ such that $[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \mathsf{skip}]$ (induction hypothesis)
3. For all t_2, if $t_2 \in [\![d_2]\!]$, then there exists $\beta_2 \in \mathcal{B}$ such that $[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \mathsf{skip}]$ (induction hypothesis)

PROVE: There exists $\beta \in \mathcal{B}$ such that

$[env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1 \textsf{ seq } d_2] \xrightarrow{t} [\beta, \textsf{skip}]$

$\langle 2 \rangle 1.$ $t \in \{h \in \mathcal{H} \mid \exists h_1 \in [\![d_1]\!], h_2 \in [\![d_2]\!] :$
$\forall l \in \mathcal{L} : e.l Ⓢ h = e.l Ⓢ h_1 \frown e.l Ⓢ h_2\}$

Proof: Assumption d is simple, assumption 1, and defs. (34), (35) and (36).

$\langle 2 \rangle 2.$ $t \in \mathcal{H} \wedge \exists h_1 \in [\![d_1]\!], h_2 \in [\![d_2]\!] : \forall l \in \mathcal{L} : e.l Ⓢ t = e.l Ⓢ h_1 \frown e.l Ⓢ h_2$

Proof: $\langle 2 \rangle 1$

$\langle 2 \rangle 3.$ There exist traces h_1, h_2, and $\beta_1, \beta_2 \in \mathcal{B}$ such that

$[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{h_1} [\beta_1, \textsf{skip}] \wedge$
$[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{h_2} [\beta_2, \textsf{skip}] \wedge$
$\forall l \in \mathcal{L} : e.l Ⓢ t = e.l Ⓢ h_1 \frown e.l Ⓢ h_2 \wedge$
$t \in \mathcal{H}$

Proof: Assumptions 2 and 3 (induction hypothesis), and $\langle 2 \rangle 2$.

$\langle 2 \rangle 4.$ There exist $\beta \in \mathcal{B}$ such that

$[env^!_{\mathcal{M}}.(d_1 \textsf{ seq } d_2), d_1 \textsf{ seq } d_2] \xrightarrow{t} [\beta, \textsf{skip}]$

Proof: $\langle 2 \rangle 3$ and lemma 12.

$\langle 2 \rangle 5.$ Q.E.D.

Proof: $\langle 2 \rangle 4$.

$\langle 1 \rangle 5.$ Case: $d = d_1 \textsf{ par } d_2$ (induction step)

Assume: 1. $t \in [\![d_1 \textsf{ par } d_2]\!]$

2. For all t_1, if $t_1 \in [\![d_1]\!]$, then there exists $\beta_1 \in \mathcal{B}$ such that $[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \textsf{skip}]$ (induction hypothesis)

3. For all t_2, if $t_2 \in [\![d_2]\!]$, then there exists $\beta_2 \in \mathcal{B}$ such that $[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \textsf{skip}]$ (induction hypothesis)

Prove: There exists $\beta \in \mathcal{B}$ such that

$[env^!_{\mathcal{M}}.(d_1 \textsf{ par } d_2), d_1 \textsf{ par } d_2] \xrightarrow{t} [\beta, \textsf{skip}]$

$\langle 2 \rangle 1.$ $t \in \{h \in \mathcal{H} \mid \exists p \in \{1, 2\}^\infty :$
$\pi_2((\{1\} \times \mathcal{E}) Ⓣ (p, h)) \in [\![d_1]\!] \wedge$
$\pi_2((\{2\} \times \mathcal{E}) Ⓣ (p, h)) \in [\![d_2]\!]$

Proof: Assumption d is simple, assumption 1, and defs. (37), (38) and (39).

$\langle 2 \rangle 2.$ $t \in \mathcal{H} \wedge \exists p \in \{1, 2\}^\infty :$
$\pi_2((\{1\} \times \mathcal{E}) Ⓣ (p, t)) \in [\![d_1]\!] \wedge$
$\pi_2((\{2\} \times \mathcal{E}) Ⓣ (p, t)) \in [\![d_2]\!]$

Proof: $\langle 2 \rangle 1$

$\langle 2 \rangle 3.$ There exist $\beta_1, \beta_2 \in \mathcal{B}, p \in \{1, 2\}^\infty$ such that

$[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{\pi_2((\{1\} \times \mathcal{E}) Ⓣ (p, t))} [\beta_1, \textsf{skip}] \wedge$

$[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{\pi_2((\{2\} \times \mathcal{E}) Ⓣ (p, t))} [\beta_2, \textsf{skip}]$

Proof: $\langle 2 \rangle 2$ and assumptions 2 and 3 (induction hypothesis).

$\langle 2 \rangle 4.$ There exist traces t_1, t_2, and $\beta_1, \beta_2 \in \mathcal{B}, p \in \{1, 2\}^\infty$ such that

$[env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1} [\beta_1, \textsf{skip}] \wedge$
$[env^!_{\mathcal{M}}.d_2, d_2] \xrightarrow{t_2} [\beta_2, \textsf{skip}] \wedge$
$\pi_2((\{1\} \times \mathcal{E}) Ⓣ (p, t)) = t_1 \wedge$
$\pi_2((\{2\} \times \mathcal{E}) Ⓣ (p, t)) = t_2 \wedge$
$t \in \mathcal{H}$

PROOF: ⟨2⟩3

⟨2⟩5. There exists $\beta \in \mathcal{B}$ such that

$$[env^{!}_{\mathcal{M}}.(d_1 \;\textsf{par}\; d_2), d_1 \;\textsf{par}\; d_2] \xrightarrow{t} [\beta, \textsf{skip}]$$

PROOF: ⟨2⟩4 and lemma 14.

⟨2⟩6. Q.E.D.

PROOF: ⟨2⟩5.

⟨1⟩6. Q.E.D.

PROOF: ⟨1⟩1, ⟨1⟩2, ⟨1⟩3, ⟨1⟩4 and ⟨1⟩5

□

B.3 Sequence diagrams with high-level operators with finite behavior

In this section we prove termination, soundness and completeness of diagrams with high-level operators with finite behavior. This means the operators neg, alt, xalt, loop$\langle n \rangle$ with $n \neq \infty$ and loop I with $\infty \notin I$. Diagrams with infinite loops are addressed in section B.4.

In the above section we assumed $[\![d]\!] = \{(T, \emptyset)\}$ for any diagram d. In this section we assume the general semantics model $[\![d]\!] = \{(p_1, n_1), (p_2, n_2), \ldots, (p_m, p_m)\}$, but will write $t \in [\![d]\!]$ as a shorthand for $t \in \bigcup_{i=1,\ldots,m}(p_i \cup n_i)$ when we find this suitable.

By termination we mean a state $[\beta, d]$ where no further execution steps are possible. Usually this means that $d = \textsf{skip}$, but it is possible to make other diagrams where no execution is possible, e.g. the diagram $d = (?, (m, l, l))$ seq $(!, (m, l, l))$ which specify that lifeline l receives message m from itself before it sends it.

Theorem 3 (Termination) *Given a diagram $d \in \mathcal{D}$ without infinite loops. If we assume progression of execution, then execution of $[env^{!}_{\mathcal{M}}.d, d]$ will terminate.*

Proof of theorem 3

ASSUME: Diagram $d \in \mathcal{D}$ without infinite loop

PROVE: Execution of $[env^{!}_{\mathcal{M}}.d_1, d_1]$ terminates

⟨1⟩1. LET: $w \in \mathcal{D} \rightarrow \mathbb{N} \cup \{0\}$ be a weight function defined as

$$\begin{aligned}
w(\textsf{skip}) &\overset{\textsf{def}}{=} 0 \\
w(e) &\overset{\textsf{def}}{=} 1 \;(\textbf{for}\; e \in \mathcal{E}) \\
w(d_1 \;\textsf{seq}\; d_2) \overset{\textsf{def}}{=} w(d_1 \;\textsf{par}\; d_2) &\overset{\textsf{def}}{=} w(d_1) + w(d_2) + 1 \\
w(d_1 \;\textsf{alt}\; d_2) \overset{\textsf{def}}{=} w(d_1 \;\textsf{xalt}\; d_2) &\overset{\textsf{def}}{=} \textbf{max}(w(d_1), w(d_2)) + 1 \\
w(\textsf{neg}\; d_1) &\overset{\textsf{def}}{=} w(d_1) + 1 \\
w(\textsf{loop}\langle n \rangle\; d_1) &\overset{\textsf{def}}{=} n \cdot (w(d_1) + 2) \\
w(\textsf{loop}\; I\; d_1) &\overset{\textsf{def}}{=} \textbf{max}(I) \cdot (w(d_1) + 2) + 1
\end{aligned}$$

⟨1⟩2. $w(d) \geq 0$

PROOF: by induction on the structure of d.

Induction start:

⟨2⟩1. CASE: $d = \textsf{skip}$

⟨3⟩1. $w(d) = 0$

PROOF: Def. of w.

⟨3⟩2. Q.E.D.
PROOF: ⟨3⟩1.
⟨2⟩2. CASE: $d = e$ (single event)
⟨3⟩1. $w(d) = 1$
PROOF: Def. of w.
⟨3⟩2. Q.E.D.
PROOF: ⟨3⟩1.
Induction step:
ASSUME: $w(d_1) \geq 0 \wedge w(d_2) \geq 0$ (Induction hypothesis)
⟨2⟩3. CASE: $d = d_1$ seq d_2 or $d = d_1$ par d_2
⟨3⟩1. $w(d) = w(d_1) + w(d_2) + 1 > 0$
PROOF: Def. of w and induction hypothesis.
⟨3⟩2. Q.E.D.
PROOF: ⟨3⟩1
⟨2⟩4. CASE: $d = d_1$ alt d_2 or $d = d_1$ xalt d_2
⟨3⟩1. $w(d) = \mathbf{max}(w(d_1), w(d_2)) + 1 > 0$
PROOF: Def. of w and $\mathbf{max}(w(d_1), w(d_2)) \geq 0$ (induction hypothesis and properties of **max**).
⟨3⟩2. Q.E.D.
PROOF: ⟨3⟩1
⟨2⟩5. CASE: $d =$ neg d_1
⟨3⟩1. $w(d) = w(d_1) + 1 > 0$
PROOF: Def. of w and induction hypothesis.
⟨3⟩2. Q.E.D.
PROOF: ⟨3⟩1.
⟨2⟩6. CASE: $d =$ loop$\langle n \rangle$ d_1
⟨3⟩1. $w(d) = n \cdot (w(d_1) + 2)$
PROOF: Def. of w.
⟨3⟩2. $w(d_1) + 2 > 0$
PROOF: Induction hypothesis.
⟨3⟩3. $n \geq 0$
PROOF: Def. of $\mathcal{D}$.
⟨3⟩4. $w(d) \geq 0$
PROOF: ⟨3⟩1, ⟨3⟩2 and ⟨3⟩3.
⟨3⟩5. Q.E.D.
PROOF: ⟨3⟩4.
⟨2⟩7. CASE: $d =$ loop I d_1
⟨3⟩1. $w(d) = \mathbf{max}(I) \cdot (w(d_1) + 2) + 1$
PROOF: Def. of w.
⟨3⟩2. $w(d_1) + 2 > 0$
PROOF: Induction hypothesis.
⟨3⟩3. $\mathbf{max}(I) \geq 0$
PROOF: Def. of $\mathcal{D}$.
⟨3⟩4. $w(d) > 0$
PROOF: ⟨3⟩1, ⟨3⟩2 and ⟨3⟩3.
⟨3⟩5. Q.E.D.
PROOF: ⟨3⟩4.
⟨2⟩8. Q.E.D.
PROOF: ⟨2⟩1-⟨2⟩7.

⟨1⟩3. $[\beta, d] \xrightarrow{e} [\beta', d'] \wedge e \in \mathcal{E} \cup \mathcal{T} \Rightarrow w(d) > w(d')$

Assume: $[\beta, d] \xrightarrow{e} [\beta', d'] \wedge e \in \mathcal{E} \cup \mathcal{T}$

Prove: $w(d) > w(d')$

Proof: by induction on the structure of d

Induction start:

⟨2⟩1. Case: $d = e$ (single event)

⟨3⟩1. $d' = \mathsf{skip}$

Proof: Rules (3) and (8).

⟨3⟩2. $w(d) = 1 < 0 = w(d')$

Proof: ⟨2⟩1, ⟨3⟩1 and def. of w.

⟨3⟩3. Q.E.D.

Proof: ⟨3⟩2.

Induction step:

Assume: $[\beta_k, d_k] \xrightarrow{e} [\beta_k, d'_k] \wedge e \in \mathcal{E} \cup \mathcal{T} \Rightarrow w(d_k) > w(d'_k),\ k \in \{1, 2\}$ (induction hypothesis)

⟨2⟩2. Case: $d = d_1$ alt d_2 or $d = d_1$ xalt d_2

⟨3⟩1. $(d' = d_1 \vee d' = d_2) \wedge e \in \mathcal{T}$

Proof: Rules (6), (13) and (14).

⟨3⟩2. Case: $d' = d_1$

⟨4⟩1. $w(d) = \mathbf{max}(w(d_1), w(d_2)) + 1 > w(d_1) = w(d')$

Proof: Def. of w and $\mathbf{max}(w(d_1), w(d_2)) \geq w(d_1)$.

⟨4⟩2. Q.E.D.

Proof: ⟨4⟩1.

⟨3⟩3. Case: $d' = d_2$

⟨4⟩1. $w(d) = \mathbf{max}(w(d_1), w(d_2)) + 1 > w(d_2) = w(d')$

Proof: Def. of w and $\mathbf{max}(w(d_1), w(d_2)) \geq w(d_2)$.

⟨4⟩2. Q.E.D.

Proof: ⟨4⟩1.

⟨3⟩4. Q.E.D.

Proof: ⟨3⟩1, ⟨3⟩2 and ⟨3⟩3.

⟨2⟩3. Case: $d = \mathsf{neg}\ d_1$

⟨3⟩1. $(d' = \mathsf{skip} \vee d' = d_1) \wedge e \in \mathcal{T}$

Proof: Rules (6), (15), (16).

⟨3⟩2. Case: $d' = \mathsf{skip}$

⟨4⟩1. $w(d) = w(d_1) + 1 > 0 = w(\mathsf{skip}) = w(d')$

Proof: Def. w, ⟨1⟩2.

⟨4⟩2. Q.E.D.

Proof: ⟨4⟩1.

⟨3⟩3. Case: $d' = d_1$

⟨4⟩1. $w(d) = w(d_1) + 1 > w(d_1) = w(d')$

Proof: Def. of w.

⟨4⟩2. Q.E.D.

Proof: ⟨4⟩1.

⟨3⟩4. Q.E.D.

Proof: ⟨3⟩1, ⟨3⟩2 and ⟨3⟩3.

⟨2⟩4. Case: $d = \mathsf{loop}\langle n \rangle\ d_1$

⟨3⟩1. $n > 0$

Proof: Assumption and $n = 0 \Rightarrow d = \mathsf{skip}$ (see section 4.2.7).

⟨3⟩2. $d = d_1$ seq $\mathsf{loop}\langle n - 1 \rangle\ d_1 \wedge e \in \mathcal{T}$

PROOF: ⟨3⟩1 and rules (6) and (18).

⟨3⟩3. $w(d) = n \cdot (w(d_1) + 2) = n \cdot w(d_1) + 2 \cdot n$

PROOF: Def. of w.

⟨3⟩4. $w(d') = w(d_1) + (n-1) \cdot (w(d_1) + 2) + 1$
$= w(d_1) + n \cdot w(d_1) + 2 \cdot n - w(d_1) - 2 + 1 = n \cdot w(d_1) + 2 \cdot n - 1$

PROOF: ⟨3⟩2 and def. of w.

⟨3⟩5. $w(d) = n \cdot w(d_1) + 2 \cdot n > n \cdot w(d_1) + 2 \cdot n - 1 = w(d')$

PROOF: ⟨3⟩3 and ⟨3⟩4.

⟨3⟩6. Q.E.D.

PROOF: ⟨3⟩5

⟨2⟩5. CASE: $d = \mathsf{loop}\ I\ d_1$

⟨3⟩1. $d' = \mathsf{loop}\langle n\rangle\ \wedge e \in \mathcal{T}$ for some $n \in I$

PROOF: (6) and (17).

⟨3⟩2. $w(d) = \mathbf{max}(I) \cdot (w(d_1) + 2) + 1$

PROOF: Def. of w.

⟨3⟩3. $w(d') = n \cdot (w(d_1) + 2)$ for some $n \in I$

PROOF: ⟨3⟩1 and def. of w.

⟨3⟩4. $w(d) > w(d')$

PROOF: ⟨3⟩2, ⟨3⟩3 and $\forall n \in I : \mathbf{max}(I) \geq n$.

⟨3⟩5. Q.E.D.

PROOF: ⟨3⟩4.

⟨2⟩6. CASE: $d = d_1\ \mathsf{seq}\ d_2$

⟨3⟩1. $((d' = d_1'\ \mathsf{seq}\ d_2 \wedge [\beta_1, d_1] \xrightarrow{e} [\beta_1', d_1'])$
$\vee(d' = d_1\ \mathsf{seq}\ d_2' \wedge [\beta_2, d_2] \xrightarrow{e} [\beta_2', d_2']))$
$\wedge e \in \mathcal{E} \cup \mathcal{T}$

PROOF: By rules (3), (9) and (10) either d_1 or d_2 is executed (see also proof of lemma 7).

⟨3⟩2. CASE: $d' = d_1'\ \mathsf{seq}\ d_2 \wedge [\beta_1, d_1] \xrightarrow{e} [\beta_1', d_1'] \wedge\ e \in \mathcal{E} \cup \mathcal{T}$

⟨4⟩1. $w(d_1) > w(d_1')$

PROOF: Case assumption ⟨3⟩2 and induction hypothesis.

⟨4⟩2. $w(d) = w(d_1) + w(d_2) + 1 > w(d_1') + w(d_2) + 1 = w(d')$

PROOF: Case assumption ⟨2⟩6, case assumption ⟨3⟩2, ⟨4⟩1 and def. of w.

⟨4⟩3. Q.E.D.

PROOF: ⟨4⟩2

⟨3⟩3. CASE: $d' = d_1\ \mathsf{seq}\ d_2' \wedge [\beta_2, d_2] \xrightarrow{e} [\beta_2', d_2'] \wedge\ e \in \mathcal{E} \cup \mathcal{T}$

⟨4⟩1. $w(d_2) > w(d_2')$

PROOF: Case assumption ⟨3⟩3 and induction hypothesis.

⟨4⟩2. $w(d) = w(d_1) + w(d_2) + 1 > w(d_1) + w(d_2') + 1 = w(d')$

PROOF: Case assumption ⟨2⟩6, case assumption ⟨3⟩3, ⟨4⟩1 and def. of w.

⟨4⟩3. Q.E.D.

PROOF: ⟨4⟩2

⟨3⟩4. Q.E.D.

PROOF: ⟨3⟩1, ⟨3⟩2 and ⟨3⟩3.

⟨2⟩7. CASE: $d = d_1\ \mathsf{par}\ d_2$

⟨3⟩1. $((d' = d_1'\ \mathsf{par}\ d_2 \wedge [\beta_1, d_1] \xrightarrow{e} [\beta_1', d_1'])$
$\vee(d' = d_1\ \mathsf{par}\ d_2' \wedge [\beta_2, d_2] \xrightarrow{e} [\beta_2', d_2']))$
$\wedge e \in \mathcal{E} \cup \mathcal{T}$

PROOF: By rules (3), (11) and (12) either d_1 or d_2 is executed (see also proof of lemma 9).

⟨3⟩2. CASE: $d' = d_1'$ par $d_2 \wedge [\beta_1, d_1] \xrightarrow{e} [\beta_1', d_1'] \wedge e \in \mathcal{E} \cup \mathcal{T}$

⟨4⟩1. $w(d_1) > w(d_1')$

PROOF: Case assumption ⟨3⟩2 and induction hypothesis.

⟨4⟩2. $w(d) = w(d_1) + w(d_2) + 1 > w(d_1') + w(d_2) + 1 = w(d')$

PROOF: Case assumption ⟨2⟩7, case assumption ⟨3⟩2, ⟨4⟩1 and def. of w.

⟨4⟩3. Q.E.D.

PROOF: ⟨4⟩2

⟨3⟩3. CASE: $d' = d_1$ par $d_2' \wedge [\beta_2, d_2] \xrightarrow{e} [\beta_2', d_2'] \wedge e \in \mathcal{E} \cup \mathcal{T}$

⟨4⟩1. $w(d_2) > w(d_2')$

PROOF: Case assumption ⟨3⟩3 and induction hypothesis.

⟨4⟩2. $w(d) = w(d_1) + w(d_2) + 1 > w(d_1) + w(d_2') + 1 = w(d')$

PROOF: Case assumption ⟨2⟩7, case assumption ⟨3⟩3, ⟨4⟩1 and def. of w.

⟨4⟩3. Q.E.D.

PROOF: ⟨4⟩2

⟨3⟩4. Q.E.D.

PROOF: ⟨3⟩1, ⟨3⟩2 and ⟨3⟩3.

⟨2⟩8. Q.E.D.

PROOF: ⟨2⟩1-⟨2⟩7.

⟨1⟩4. Q.E.D.

PROOF: ⟨1⟩1, ⟨1⟩2 and ⟨1⟩3.

□

Definition 12 *Following [46] we define an operator* refuse*:*

$$d \in \mathcal{D} \Rightarrow \mathsf{refuse}\ d \in \mathcal{D}$$

The denotational semantics of refuse *is:*

$$[\![\, \mathsf{refuse}\ d \,]\!] \overset{\mathsf{def}}{=} \{(\emptyset, p \cup n) \mid (p, n) \in [\![\, d \,]\!]\} \tag{52}$$

For the operator define the following Π*-rule:*

$$\Pi(L, \beta, \mathsf{refuse}\ d) \xrightarrow{\tau_{refuse}} \Pi(L, \beta, d) \tag{53}$$

with $\tau_{refuse} \in \mathcal{T}$.

Lemma 15 *Given diagram* $d \in \mathcal{D}$.

1. $[\![\, \mathsf{neg}\ d \,]\!] = [\![\, \mathsf{skip\ alt\ refuse}\ d \,]\!]$
2. $[\beta, \mathsf{neg}\ d] \xrightarrow{\tau_{pos}} [\beta, \mathsf{skip}] \iff [\beta, \mathsf{skip\ alt\ refuse}\ d] \xrightarrow{\tau_{alt}} [\beta, \mathsf{skip}]$
3. $[\beta, \mathsf{neg}\ d] \xrightarrow{\langle \tau_{neg} \rangle \frown t} [\beta', \mathsf{skip}] \iff$

   $[\beta, \mathsf{skip\ alt\ refuse}\ d] \xrightarrow{\langle \tau_{alt}, \tau_{refuse} \rangle \frown t} [\beta', \mathsf{skip}]$

Proof of lemma 15

$\langle 1\rangle 1.$ PROOF: of 1

ASSUME: $[\![\, d\,]\!] = \{(p_1, n_1), (p_2, n_2), \ldots, (p_m, n_m)\}$

$\langle 2\rangle 1.$ $[\![\, \mathsf{neg}\ d\,]\!] = \{(\{\langle\rangle\}, p_1 \cup n_1), (\{\langle\rangle\}, p_2 \cup n_2), \ldots (\{\langle\rangle\}, p_m \cup n_m)\}$

PROOF: Assumption and def. (42).

$\langle 2\rangle 2.$ $[\![\, \mathsf{skip\ alt\ refuse}\ d\,]\!] = [\![\, \mathsf{skip}\,]\!] \uplus [\![\, \mathsf{refuse}\ d\,]\!] =$
$\{(\{\langle\rangle\}, \emptyset)\} \uplus \{(\emptyset, p_1 \cup n_1), (\emptyset, p_2 \cup n_2), \ldots, (\emptyset, p_m \cup n_m)\} =$
$\{(\{\langle\rangle\}, p_1 \cup n_1), (\{\langle\rangle\}, p_2 \cup n_2), \ldots (\{\langle\rangle\}, p_m \cup n_m)\}$

PROOF: Assumption and defs. (40), (43) and (52).

$\langle 2\rangle 3.$ Q.E.D.

PROOF: $\langle 2\rangle 1$ and $\langle 2\rangle 2$.

$\langle 1\rangle 2.$ PROOF: of 2

$\langle 2\rangle 1.$ $[\beta, \mathsf{neg}\ d] \xrightarrow{\tau_{pos}} [\beta, \mathsf{skip}]$

PROOF: Rules (6) and (15).

$\langle 2\rangle 2.$ $[\beta, \mathsf{skip\ alt\ refuse}\ d] \xrightarrow{\tau_{alt}} [\beta, \mathsf{skip}]$

PROOF: Rules (6) and (13).

$\langle 2\rangle 3.$ Q.E.D.

PROOF: $\langle 2\rangle 1$ and $\langle 2\rangle 2$.

$\langle 1\rangle 3.$ PROOF: of 3

ASSUME: $[\beta, d] \xrightarrow{t} [\beta', \mathsf{skip}]$

$\langle 2\rangle 1.$ $[\beta, \mathsf{neg}\ d] \xrightarrow{\tau_{neg}} [\beta, d]$

PROOF: Rules (16) and (6).

$\langle 2\rangle 2.$ $[\beta, \mathsf{skip\ alt\ refuse}\ d] \xrightarrow{\tau_{alt}} [\beta, \mathsf{refuse}\ d] \xrightarrow{\tau_{refuse}} [\beta, d]$

PROOF: Rules (6), (13) and (53).

$\langle 2\rangle 3.$ Q.E.D.

PROOF: Assumption and $\langle 2\rangle 1$ and $\langle 2\rangle 2$

$\langle 1\rangle 4.$ Q.E.D.

PROOF: $\langle 1\rangle 1$-$\langle 1\rangle 3$.

□

Definition 13 *We define* neg d *as*

$$\mathsf{neg}\ d \overset{\mathsf{def}}{=} \mathsf{skip\ alt\ refuse}\ d$$

Definition 14 *We define the denotation of* $\mathsf{loop}\langle n\rangle\ d$ *as*

$$[\![\, \mathsf{loop}\langle n\rangle\ d\,]\!] \overset{\mathsf{def}}{=} \mu_n\ [\![\, d\,]\!] \tag{54}$$

Lemma 16 *Given diagram $d \in \mathcal{D}$.*

1. $[\![\, \mathsf{loop}\ I\ d\,]\!] = [\![\, \mathsf{alt}_{i\in I}\ \mathsf{loop}\langle i\rangle\ d\,]\!]$

2. $[\beta, \mathsf{loop}\ I\ d] \xrightarrow{t} [\beta', \mathsf{skip}] \iff [\beta, \mathsf{alt}_{i\in I}\ \mathsf{loop}\langle i\rangle\ d] \xrightarrow{t} [\beta', \mathsf{skip}]$

where $\mathsf{alt}_{i\in I}$ *is a generalized* alt.

Proof of lemma 16

$\langle 1\rangle 1.$ PROOF: of 1

$\langle 2\rangle 1.$ $[\![\, \mathsf{loop}\ I\ d\,]\!] = \biguplus_{i\in I} \mu_i [\![\, d\,]\!]$

Proof: Def. (44).

⟨2⟩2. $[\![\,\mathsf{alt}_{i\in I}\ \mathsf{loop}\langle i\rangle\ d\,]\!] = \biguplus_{i\in I}[\![\,\mathsf{loop}\langle i\rangle\ d\,]\!] = \biguplus_{i\in I}\mu_i[\![\,d\,]\!]$

Proof: Defs. (40), (43) (54).

⟨2⟩3. Q.E.D.

Proof: ⟨2⟩1 and ⟨2⟩2.

⟨1⟩2. Proof: of 2

Assume: $[\beta_j, \mathsf{loop}\langle j\rangle\ d] \xrightarrow{t_j} [\beta'_j, \mathsf{skip}]$ for $j \in I$

⟨2⟩1. $[\beta, \mathsf{loop}\ I\ d] \xrightarrow{\tau_{alt}} [\beta, \mathsf{loop}\langle n\rangle\ d]$ for $n \in I$

Proof: Rules (6) and (17).

⟨2⟩2. $[\beta, \mathsf{alt}_{i\in I}\mathsf{loop}\langle i\rangle\ d] \xrightarrow{\tau_{alt}} [\beta, \mathsf{loop}\langle n\rangle\ d]$ for $n \in I$.

Proof: Rules (6) and (13), and associativity and commutativity of alt.

⟨2⟩3. Q.E.D.

Proof: Assumption and ⟨2⟩1 and ⟨2⟩2 (by setting $n = j$, $\beta = \beta_j$, $\beta' = \beta'_j$ and $t = \langle\tau_{alt}\rangle^\frown t_j$).

⟨1⟩3. Q.E.D.

Proof: ⟨1⟩1 and ⟨1⟩2.

□

Definition 15 *We define* loop I d *as*

$$\mathsf{loop}\ I\ d \overset{\mathsf{def}}{=} \mathsf{alt}_{i\in I}\ \mathsf{loop}\langle i\rangle\ d$$

To simplify the proofs of soundness and completeness we now apply definitions 13 and 15 as the definitions of neg and loop I. By lemmas 15 and 16 we may do this without any change in the semantics of neg and loop I. (It would also be possible to define loop$\langle n\rangle$ by means of $n-1$ seqs. Because this would remove the silent events τ_{loop} we choose not to do this.) This means that in the following we will only be concerned with diagrams built from the operators seq, par, refuse, alt, xalt and loop$\langle n\rangle$.

In the previous section we assumed that diagrams do not have repetition of events. We make the same assumption for diagrams with high-level operators. For the operators refuse, alt and xalt this is straight forward by assuming that for a diagram d:

$$d = d_1\ \mathsf{alt}\ d_2 \vee d = d_1\ \mathsf{xalt}\ d_2 \Rightarrow ev.d_1 \cap ev.d_2 = \emptyset$$

and extending the definition of $msg.\_$ and $ev.\_$ with

$$msg.\mathsf{refuse}\ d \overset{\mathsf{def}}{=} msg.d$$

$$ev.\mathsf{refuse}\ d \overset{\mathsf{def}}{=} ev.d$$

However, the loop is a bit more problematic. Given a diagram d, assume that we rename every signal s in d and give it the new name s^k where $k \in \mathbb{N}$. Let d^k denote the resulting diagram. We now assume, both in the denotational and the operational semantics, that we have an implicit counter and a way of doing this renaming every time the body of a loop is "copied" out of the loop. A way to do this could be to redefine the Π-rule as

$$\Pi(L, \beta, \mathsf{loop}^c\langle n\rangle\ d) \xrightarrow{\tau_{loop}} \Pi(L, \beta, d^c\ \mathsf{seq}\ \mathsf{loop}^{c+1}\langle n-1\rangle\ d) \tag{55}$$

and the definition of semantic loop to

$$\mu_n^c \, [\![\, d \,]\!] \overset{\mathsf{def}}{=} [\![\, d^c \,]\!] \succsim \mu_{n-1}^{c+1} \, [\![\, d \,]\!] \tag{56}$$

where $c \in \mathbb{N}$ is a counter. Further we could need to redefine $ev.\_$ and $msg.\_$ as follows

$$msg.\mathsf{loop}^c\langle n \rangle \; d \overset{\mathsf{def}}{=} msg.d^c \cup msg.\mathsf{loop}^{c+1}\langle n-1 \rangle \; d$$

$$ev.\mathsf{loop}^c\langle n \rangle \; d \overset{\mathsf{def}}{=} ev.d^c \cup ev.\mathsf{loop}^{c+1}\langle n-1 \rangle \; d$$

In the following we assume these new definitions, also when not writing it out explicitly.

Lemma 17 *Given a diagram d. For all traces t such that (there exists β such that)*

$$[env^!_{\mathcal{M}}.d, d] \xrightarrow{t} [\beta, \mathsf{skip}]$$

we may find a simple diagram d' (and β') such that

$$[env^!_{\mathcal{M}}.d', d'] \xrightarrow{\mathcal{E}\circledS t} [\beta', \mathsf{skip}]$$

Proof of lemma 17
ASSUME: $\exists \beta : [env^!_{\mathcal{M}}.d, d] \xrightarrow{t} [\beta, \mathsf{skip}]$
PROVE: $\exists d', \beta' : [env^!_{\mathcal{M}}.d', d'] \xrightarrow{\mathcal{E}\circledS t} [\beta', \mathsf{skip}]$, d' simple
PROOF: by induction on d.
⟨1⟩1. Induction start: d simple
PROOF: Trivial by letting $d = d'$ and $\beta = \beta'$. By lemma 5, $\mathcal{E}\circledS t = t$.
Induction step:
ASSUME: $\forall d_k, t_k : \exists \beta_k : [env^!_{\mathcal{M}}.d_k, d_k] \xrightarrow{t_k} [\beta_k, \mathsf{skip}] \Rightarrow$
$\exists d'_k, \beta'_k : [env^!_{\mathcal{M}}.d'_k, d'_k] \xrightarrow{\mathcal{E}\circledS t_k} [\beta'_k, \mathsf{skip}]$, d'_k simple (induction hypothesis)
⟨1⟩2. CASE: $d = d_1$ seq d_2 ($d = d_1$ par d_2)
PROOF: By rules (3), (9) and (10) ((3), (11) and (12)), t is obtained by alternately executing d_1 and d_2 in some suitable fashion. This means there must exist traces t_1 and t_2 (and β_1 and β_2) such that $[env^!_{\mathcal{M}}.d_k, d_k] \xrightarrow{t_k} [\beta_k, \mathsf{skip}]$ for $k \in \{1, 2\}$ and t is (some kind of) merge of t_1 and t_2. By the induction hypothesis we may find d'_k such that it produces $\mathcal{E}\circledS t_k$ (for $k \in \{1, 2\}$). Let $d' = d'_1$ seq d'_2 ($d' = d'_1$ par d'_2). Then clearly d' is simple. Further, by applying the same execution scheme (i.e., alternation between the left and right hand side of the operator), except for the silent events in t, d' produces $\mathcal{E}\circledS t$.
⟨1⟩3. CASE: $d = \mathsf{refuse}$ d_1
PROOF: By assumption and rules (6) and (53), $t = \langle \tau_{refuse} \rangle ^\frown t_1$ such that $[env^!_{\mathcal{M}}.\mathsf{refuse}\ d_1, d_1] \xrightarrow{t_1} [\beta, \mathsf{skip}]$. By $env^!_{\mathcal{M}}.\mathsf{refuse}\ d_1 = env^!_{\mathcal{M}}.d_1$ and the induction hypothesis, we may find simple d'_1 such that $[env^!_{\mathcal{M}}.d'_1, d'_1] \xrightarrow{\mathcal{E}\circledS t_1} [\beta'_1, \mathsf{skip}]$. Let $d' = d'_1$. Clearly d' is simple. Further, we have that $[env^!_{\mathcal{M}}.d', d'] \xrightarrow{\mathcal{E}\circledS t} [\beta'_1, \mathsf{skip}]$ since $\mathcal{E}\circledS \langle \tau_{refuse} \rangle ^\frown t_1 = \mathcal{E}\circledS t_1$ (by def. of $\circledS$ and $\tau_{refuse} \notin \mathcal{E}$).
⟨1⟩4. CASE: $d = d_1$ alt d_2

PROOF: By assumption and rules (6) and (13), $t = \langle \tau_{alt} \rangle^\frown t_k$ such that $[env^!_{\mathcal{M}}.d, d_k] \xrightarrow{t_k} [\beta, \mathsf{skip}]$, for $k = 1$ or $k = 2$. Because $env^!_{\mathcal{M}}.d = env^!_{\mathcal{M}}.d_1 \cup env^!_{\mathcal{M}}.d_2$, $env^!_{\mathcal{M}}.d_1 \cap env^!_{\mathcal{M}}.d_2 = \emptyset$ and by the induction hypothesis we may find simple d'_k such that $[env^!_{\mathcal{M}}.d'_k, d'_k] \xrightarrow{\mathcal{E} \circledS t_k} [\beta'_k, \mathsf{skip}]$ for the appropriate choice of k. Let $d' = d'_k$ (for this choice of k). Clearly d' is simple. Further, we have that $[env^!_{\mathcal{M}}.d', d'] \xrightarrow{\mathcal{E} \circledS t} [\beta'_k, \mathsf{skip}]$ since $\mathcal{E} \circledS t = \mathcal{E} \circledS \langle \tau_{alt} \rangle^\frown t_k = \mathcal{E} \circledS t_k$ (by def. of $\circledS$ and $\tau_{alt} \notin \mathcal{E}$).

$\langle 1 \rangle 5$. CASE: $d = d_1$ xalt d_2

PROOF: By assumption and rules (6) and (14), $t = \langle \tau_{xalt} \rangle^\frown t_k$ such that $[env^!_{\mathcal{M}}.d, d_k] \xrightarrow{t_k} [\beta, \mathsf{skip}]$, for $k = 1$ or $k = 2$. Because $env^!_{\mathcal{M}}.d = env^!_{\mathcal{M}}.d_1 \cup env^!_{\mathcal{M}}.d_2$, $env^!_{\mathcal{M}}.d_1 \cap env^!_{\mathcal{M}}.d_2 = \emptyset$ and by the induction hypothesis we may find simple d'_k such that $[env^!_{\mathcal{M}}.d'_k, d'_k] \xrightarrow{\mathcal{E} \circledS t_k} [\beta'_k, \mathsf{skip}]$ for the appropriate choice of k. Let $d' = d'_k$ (for this choice of k). Clearly d' is simple. Further, we have that $[env^!_{\mathcal{M}}.d', d'] \xrightarrow{\mathcal{E} \circledS t} [\beta'_k, \mathsf{skip}]$ since $\mathcal{E} \circledS t = \mathcal{E} \circledS \langle \tau_{xalt} \rangle^\frown t_k = \mathcal{E} \circledS t_k$ (by def. of $\circledS$ and $\tau_{xalt} \notin \mathcal{E}$).

$\langle 1 \rangle 6$. CASE: $d = \mathsf{loop}\langle n \rangle\ d_1$

PROOF: by induction on n.

$\langle 2 \rangle 1$. Induction start: $n = 1$

PROOF: By assumption and rules (6) and (18), $t = \langle \tau_{loop} \rangle^\frown t_1$ such that $[env^!_{\mathcal{M}}.\mathsf{loop}\langle 1 \rangle\ d_1, \mathsf{loop}\langle 1 \rangle\ d_1] \xrightarrow{\tau_{loop}} [env^!_{\mathcal{M}}.\mathsf{loop}\langle 1 \rangle\ d_1, d_1\ \mathsf{seq}\ \mathsf{loop}\langle 0 \rangle\ d_1] = [env^!_{\mathcal{M}}.d_1 \cup env^!_{\mathcal{M}}.\mathsf{skip}, d_1\ \mathsf{seq}\ \mathsf{skip}] = [env^!_{\mathcal{M}}.d_1, d_1]$. By the induction hypothesis we may find simple d'_1 such that $[env^!_{\mathcal{M}}.d'_1, d'_1] \xrightarrow{\mathcal{E} \circledS t_1} [\beta'_1, \mathsf{skip}]$. Let $d' = d'_1$. Clearly d' is simple. Further, we have that $[env^!_{\mathcal{M}}.d', d'] \xrightarrow{\mathcal{E} \circledS t}$ since $\mathcal{E} \circledS t = \mathcal{E} \circledS \langle \tau_{loop} \rangle^\frown t_1 = \mathcal{E} \circledS t_1$ (by def. of $\circledS$ and $\tau_{loop} \notin \mathcal{E}$).

$\langle 2 \rangle 2$. Induction step: $n = k + 1$

ASSUME: For all t and d_1, if these exists β such that $[env^!_{\mathcal{M}}.\mathsf{loop}\langle k \rangle\ d_1, \mathsf{loop}\langle k \rangle\ d_1] \xrightarrow{t} [\beta, \mathsf{skip}]$ then we may find simple d' (and β') such that $[env^!_{\mathcal{M}}.d', d'] \xrightarrow{\mathcal{E} \circledS t} [\beta', \mathsf{skip}]$ (induction hypothesis 2)

PROOF: By assumption and rules (6) and (18), $t = \langle \tau_{loop} \rangle^\frown t'$ such that $[env^!_{\mathcal{M}}.\mathsf{loop}\langle k+1 \rangle\ d_1, \mathsf{loop}\langle k+1 \rangle\ d_1] \xrightarrow{\tau_{loop}} [env^!_{\mathcal{M}}.\mathsf{loop}\langle k+1 \rangle\ d_1, d_1\ \mathsf{seq}\ \mathsf{loop}\langle k \rangle\ d_1] \xrightarrow{t'} [\beta, \mathsf{skip}]$. We have that $env^!_{\mathcal{M}}.\mathsf{loop}\langle k+1 \rangle\ d_1 = env^!_{\mathcal{M}}.d_1 \cup env^!_{\mathcal{M}}.\mathsf{loop}\langle k \rangle\ d_1 = env^!_{\mathcal{M}}.d_1\ \mathsf{seq}\ \mathsf{loop}\langle k \rangle\ d_1$. By this, the induction hypotheses and $\langle 1 \rangle 2$ we may find simple d'' such that $[env^!_{\mathcal{M}}.d'', d''] \xrightarrow{\mathcal{E} \circledS t'} [\beta', \mathsf{skip}]$. Let $d' = d''$. Clearly d' is simple. Further, we have that $[env^!_{\mathcal{M}}.d', d'] \xrightarrow{\mathcal{E} \circledS t}$ since $\mathcal{E} \circledS t = \mathcal{E} \circledS \langle \tau_{loop} \rangle^\frown t' = \mathcal{E} \circledS t'$ (by def. of $\circledS$ and $\tau_{loop} \notin \mathcal{E}$).

$\langle 2 \rangle 3$. Q.E.D.

PROOF: $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$.

$\langle 1 \rangle 7$. Q.E.D.

PROOF: $\langle 1 \rangle 1$-$\langle 1 \rangle 6$.

□

Lemma 18 *Given a diagram d. For all traces t such that*

$$t \in [\![\, d \,]\!]$$

we may find a simple diagram d' such that

$$t \in [\![d']\!]$$

Proof of lemma 18
ASSUME: $t \in [\![d]\!]$
PROVE: $\exists d' : t \in [\![d']\!]$, d' simple.
PROOF: by induction on d.
$\langle 1 \rangle 1$. Induction start: d simple
PROOF: Trivial: Let $d = d'$.
Induction step:
ASSUME: If $t_k \in [\![d_k]\!]$ then there exists simple d'_k such that $t_k \in [\![d'_k]\!]$ (induction hypothesis)
$\langle 1 \rangle 2$. $d = d_1$ seq d_2 ($d = d_1$ par d_2)
PROOF: By definitions (34)-(36) ((37)-(39)) there exist traces t_1 and t_2 such that t is obtained from t_1 and t_2. By the induction hypothesis we may find simple d'_1 and d'_2 such that $t_1 \in [\![d'_1]\!]$ and $t_2 \in [\![d'_2]\!]$. Let $d' = d'_1$ seq d'_2 ($d' = d'_1$ par d'_2). Clearly d' is simple and $t \in [\![d']\!]$.
$\langle 1 \rangle 3$. $d = \mathsf{refuse}$ d_1
PROOF: By def. $[\![d_1]\!] = \{(p_1, n_1), \ldots, (p_m, n_m)\}$ iff $[\![d]\!] = \{(\emptyset, p_1 \cup n_1), \ldots, (\emptyset, p_m \cup n_m)\}$, so $t \in [\![d]\!]$ iff $t \in p_i$ or $t \in n_i$ (for some $i \in \{1, \ldots, m\}$) iff $t \in [\![d_1]\!]$. By induction hypothesis there exists simple d'_1 such that $t \in [\![d'_1]\!]$. Let $d' = d'_1$. Clearly d' is simple and $t \in [\![d']\!]$.
$\langle 1 \rangle 4$. $d = d_1$ alt d_2 ($d = d_1$ xalt d_2)
PROOF: By def. $t \in [\![d_1 \ \mathsf{alt} \ d_2]\!]$ ($t \in [\![d_1 \ \mathsf{xalt} \ d_2]\!]$) iff $t \in [\![d_1]\!] \uplus [\![d_2]\!]$ ($t \in [\![d_1]\!] \cup [\![d_2]\!]$) iff $t \in [\![d_1]\!]$ or $t \in [\![d_2]\!]$. Chose the appropriate $k \in \{1, 2\}$. By induction hypothesis there exist simple d'_k such that $t \in [\![d'_k]\!]$. Let $d' = d'_k$. Clearly d' is simple and $t \in [\![d']\!]$.
$\langle 1 \rangle 5$. $d = \mathsf{loop}\langle n \rangle$ d_1
PROOF: By def. $[\![d]\!] = [\![\mathsf{loop}\langle n \rangle \ d_1]\!] = \mu_n [\![d_1]\!] = [\![d_1^1]\!] \succsim [\![d_1^2]\!] \succsim \cdots \succsim [\![d_1^n]\!]$. By this and defs. (34)-(36) we have that $[\![d]\!] = [\![d_1^1 \ \mathsf{seq} \ d_1^2 \ \mathsf{seq} \ \cdots \ \mathsf{seq} \ d_1^n]\!]$. By the induction hypothesis and $\langle 1 \rangle 2$ we may find simple $d_1^{1\prime}, d_2^{2\prime}, \ldots, d_2^{n\prime}$ such that $t \in [\![d_1^{1\prime} \ \mathsf{seq} \ d_1^{2\prime} \ \mathsf{seq} \ \cdots \ \mathsf{seq} \ d_1^{n\prime}]\!]$. Let $d' = d_1^{1\prime} \ \mathsf{seq} \ d_1^{2\prime} \ \mathsf{seq} \ \cdots \ \mathsf{seq} \ d_1^{n\prime}$. Clearly d' is simple and $t \in [\![d']\!]$.
$\langle 1 \rangle 6$. Q.E.D.
PROOF: $\langle 1 \rangle 1$-$\langle 1 \rangle 5$.

□

Theorem 4 (Soundness) *Given diagram $d \in \mathcal{D}$ without infinite loop. For all traces $t \in (\mathcal{E} \cup \mathcal{T})^*$, if there exists $\beta \in \mathcal{B}$ such that*

$$[env^!_{\mathcal{M}}.d, d] \xrightarrow{t} [\beta, \mathsf{skip}]$$

then

$$\mathcal{E} \circledS t \in [\![d]\!]$$

Proof of theorem 4
ASSUME: $\exists \beta : [env^!_{\mathcal{M}}.d, d] \xrightarrow{t} [\beta, \mathsf{skip}]$
PROVE: $\mathcal{E} \circledS t \in [\![d]\!]$

PROOF: by induction on d.

$\langle 1\rangle 1.$ Induction start: $d = \mathsf{skip},\ d = (!, m),\ d = (?, m)$

PROOF: This carries over from theorem 1.

Induction step:

ASSUME: $\exists \beta_k : [env^!_{\mathcal{M}}.d_k, d_k] \xrightarrow{t_k} [\beta_k, \mathsf{skip}] \Rightarrow \mathcal{E} \circledS t_k \in [\![\, d \,]\!]$

$\langle 1\rangle 2.$ CASE: $d = d_1\ \mathsf{seq}\ d_2,\ d = d_1\ \mathsf{par}\ d_2$

PROOF: By lemmas 17 and 18 this follows from theorem 1

$\langle 1\rangle 3.$ CASE: $d = \mathsf{refuse}\ d_1$

$\langle 2\rangle 1.$ $t = \langle \tau_{refuse} \rangle ^\frown t_1$ such that $[env^!_{\mathcal{M}}.d, d] \xrightarrow{\tau_{refuse}} [env^!_{\mathcal{M}}.d, d_1] \xrightarrow{t_1} [\beta, \mathsf{skip}]$

PROOF: Assumptions and rules (6) and (53).

$\langle 2\rangle 2.$ $\mathcal{E} \circledS t = \mathcal{E} \circledS \langle \tau_{refuse} \rangle ^\frown t_1 = \mathcal{E} \circledS t_1$

PROOF: $\langle 2\rangle 1$, def. of $\circledS$ and $\tau_{refuse} \notin \mathcal{E}$.

$\langle 2\rangle 3.$ $\mathcal{E} \circledS t \in [\![\, d_1 \,]\!]$

PROOF: $\langle 2\rangle 2$ and induction hypothesis (by $env^!_{\mathcal{M}}.\mathsf{refuse}\ d_1 = env^!_{\mathcal{M}}.d_1$).

$\langle 2\rangle 4.$ $\mathcal{E} \circledS t \in [\![\, \mathsf{refuse}\ d_1 \,]\!]$

PROOF: Def. 12 of the denotation of refuse.

$\langle 2\rangle 5.$ Q.E.D.

PROOF: $\langle 2\rangle 4$.

$\langle 1\rangle 4.$ CASE: $d = d_1\ \mathsf{alt}\ d_2$

$\langle 2\rangle 1.$ $t = \langle \tau_{alt} \rangle ^\frown t_k$ such that $[env^!_{\mathcal{M}}.d, d] \xrightarrow{\tau_{alt}} [env^!_{\mathcal{M}}.d, d_k] \xrightarrow{t_k} [\beta, \mathsf{skip}]$ for $k = 1$ or $k = 2$

PROOF: Assumptions and rules (6) and (13).

$\langle 2\rangle 2.$ $\mathcal{E} \circledS t = \mathcal{E} \circledS \langle \tau_{alt} \rangle ^\frown t_k = \mathcal{E} \circledS t_k$

PROOF: $\langle 2\rangle 1$, def. of $\circledS$ and $\tau_{alt} \notin \mathcal{E}$.

$\langle 2\rangle 3.$ $\mathcal{E} \circledS t \in [\![\, d_k \,]\!]$ for $k = 1$ or $k = 2$

PROOF: $\langle 2\rangle 2$ and induction hypothesis (by $env^!_{\mathcal{M}}.d = env^!_{\mathcal{M}}.d_1 \cup env^!_{\mathcal{M}}.d_2$, $env^!_{\mathcal{M}}.d_1 \cap env^!_{\mathcal{M}}.d_2 = \emptyset$).

$\langle 2\rangle 4.$ $\mathcal{E} \circledS t \in [\![\, d_1\ \mathsf{alt}\ d_2 \,]\!]$

PROOF: $\langle 2\rangle 3$ and def. (40).

$\langle 2\rangle 5.$ Q.E.D.

PROOF: $\langle 2\rangle 4$.

$\langle 1\rangle 5.$ CASE: $d = d_1\ \mathsf{xalt}\ d_2$

$\langle 2\rangle 1.$ $t = \langle \tau_{xalt} \rangle ^\frown t_k$ such that $[env^!_{\mathcal{M}}.d, d] \xrightarrow{\tau_{xalt}} [env^!_{\mathcal{M}}.d, d_k] \xrightarrow{t_k} [\beta, \mathsf{skip}]$ for $k = 1$ or $k = 2$

PROOF: Assumptions and rules (6) and (14).

$\langle 2\rangle 2.$ $\mathcal{E} \circledS t = \mathcal{E} \circledS \langle \tau_{xalt} \rangle ^\frown t_k = \mathcal{E} \circledS t_k$

PROOF: $\langle 2\rangle 1$, def. of $\circledS$ and $\tau_{xalt} \notin \mathcal{E}$.

$\langle 2\rangle 3.$ $\mathcal{E} \circledS t \in [\![\, d_k \,]\!]$ for $k = 1$ or $k = 2$

PROOF: $\langle 2\rangle 2$ and induction hypothesis (by $env^!_{\mathcal{M}}.d = env^!_{\mathcal{M}}.d_1 \cup env^!_{\mathcal{M}}.d_2$, $env^!_{\mathcal{M}}.d_1 \cap env^!_{\mathcal{M}}.d_2 = \emptyset$).

$\langle 2\rangle 4.$ $\mathcal{E} \circledS t \in [\![\, d_1\ \mathsf{xalt}\ d_2 \,]\!]$

PROOF: $\langle 2\rangle 3$ and def. (41).

$\langle 2\rangle 5.$ Q.E.D.

PROOF: $\langle 2\rangle 4$.

$\langle 1\rangle 6.$ CASE: $d = \mathsf{loop}\langle n \rangle\ d_1$

PROOF: Induction on n

$\langle 2\rangle 1.$ Induction start: $n = 1$

$\langle 3\rangle 1.$ $t = \langle \tau_{loop} \rangle ^\frown t_1$ such that $[env^!_{\mathcal{M}}.\mathsf{loop}\langle 1 \rangle\ d_1, \mathsf{loop}\langle 1 \rangle\ d_1] \xrightarrow{\tau_{loop}}$

$[env^!_{\mathcal{M}}.\mathsf{loop}\langle 1\rangle\ d_1, d_1\ \mathsf{seq}\ \mathsf{loop}\langle 0\rangle\ d_1] \xrightarrow{t_1} [\beta, \mathsf{skip}]$

PROOF: Assumptions and rules (6) and (18).

$\langle 3\rangle 2.$ $\mathcal{E}ⓢt_1 \in [\![d_1]\!]$

PROOF: $\langle 3\rangle 1$ and induction hypothesis (by $d_1\ \mathsf{seq}\ \mathsf{loop}\langle 0\rangle\ d_1 = d_1\ \mathsf{seq}\ \mathsf{skip} = d_1$ and $env^!_{\mathcal{M}}.\mathsf{loop}\langle 1\rangle\ d_1 = env^!_{\mathcal{M}}.d_1 \cup env^!_{\mathcal{M}}.\mathsf{skip} = env^!_{\mathcal{M}}.d_1$).

$\langle 3\rangle 3.$ $\mathcal{E}ⓢt = \mathcal{E}ⓢ\langle \tau_{loop}\rangle ^\frown t_1 = \mathcal{E}ⓢt_1$

PROOF: $\langle 3\rangle 1$, def. of ⓢ and $\tau_{loop} \notin \mathcal{E}$.

$\langle 3\rangle 4.$ $[\![\mathsf{loop}\langle 1\rangle\ d_1]\!] = \mu_1 [\![d_1]\!] = [\![d_1]\!]$

PROOF: Defs. (45), (54).

$\langle 3\rangle 5.$ $\mathcal{E}ⓢt = \mathcal{E}ⓢt_1 \in [\![d_1]\!] = [\![\mathsf{loop}\langle 1\rangle\ d_1]\!] = [\![d]\!]$

PROOF: $\langle 3\rangle 2$, $\langle 3\rangle 3$ and $\langle 3\rangle 4$.

$\langle 3\rangle 6.$ Q.E.D.

PROOF: $\langle 3\rangle 5$

$\langle 2\rangle 2.$ Induction step: $n = k+1$

ASSUME: $\forall d_1, s : \exists \beta : [env^!_{\mathcal{M}}.\mathsf{loop}\langle k\rangle\ d_1, \mathsf{loop}\langle k\rangle\ d_1] \xrightarrow{s} [\beta, \mathsf{skip}] \Rightarrow \mathcal{E}ⓢs \in [\![\mathsf{loop}\langle k\rangle\ ]\!]$ (induction hypothesis 2)

$\langle 3\rangle 1.$ $t = \langle \tau_{loop}\rangle ^\frown t'$ such that $[env^!_{\mathcal{M}}.\mathsf{loop}\langle k+1\rangle\ d_1, \mathsf{loop}\langle k+1\rangle\ d_1] \xrightarrow{\tau_{loop}}$

$[env^!_{\mathcal{M}}.\mathsf{loop}\langle k+1\rangle\ , d_1\ \mathsf{seq}\ \mathsf{loop}\langle k\rangle\ d_1] \xrightarrow{t'} [\beta, \mathsf{skip}]$

PROOF: Assumptions and rules (6) and (18).

$\langle 3\rangle 2.$ $\mathcal{E}ⓢt' \in [\![d_1\ \mathsf{seq}\ \mathsf{loop}\langle k\rangle\ d_1]\!]$

PROOF: $\langle 3\rangle 1$, induction hypotheses (by $env^!_{\mathcal{M}}.\mathsf{loop}\langle k+1\rangle\ d_1 = env^!_{\mathcal{M}}.d_1 \cup env^!_{\mathcal{M}}.\mathsf{loop}\langle k\rangle\ d_1 = env^!_{\mathcal{M}}.d_1\ \mathsf{seq}\ \mathsf{loop}\langle k\rangle\ d_1$) and $\langle 1\rangle 2$.

$\langle 3\rangle 3.$ $\mathcal{E}ⓢt = \mathcal{E}ⓢ\langle \tau_{loop}\rangle ^\frown t' = \mathcal{E}ⓢt'$

PROOF: $\langle 3\rangle 1$, def. of ⓢ and $\tau_{loop} \notin \mathcal{E}$.

$\langle 3\rangle 4.$ $[\![\mathsf{loop}\langle k+1\rangle\ d_1]\!] = \mu_{k+1} [\![d_1]\!] = [\![d_1]\!] \succsim \mu_k [\![d_1]\!]$
$= [\![d_1]\!] \succsim [\![\mathsf{loop}\langle k\rangle\ d_1]\!] = [\![d_1\ \mathsf{seq}\ \mathsf{loop}\langle k\rangle\ d_1]\!]$

PROOF: Defs. (34), (35), (36), (45), (54).

$\langle 3\rangle 5.$ $\mathcal{E}ⓢt = \mathcal{E}ⓢt' \in [\![d_1\ \mathsf{seq}\ \mathsf{loop}\langle k\rangle\ d_1]\!] = [\![\mathsf{loop}\langle k+1\rangle\ d_1]\!] = [\![d]\!]$

PROOF: $\langle 3\rangle 2$, $\langle 3\rangle 3$ and $\langle 3\rangle 4$.

$\langle 3\rangle 6.$ Q.E.D.

PROOF: $\langle 3\rangle 5$.

$\langle 2\rangle 3.$ Q.E.D.

PROOF: $\langle 2\rangle 1$ and $\langle 2\rangle 2$.

$\langle 1\rangle 7.$ Q.E.D.

PROOF: $\langle 1\rangle 1$-$\langle 1\rangle 6$.

□

Theorem 5 (Completeness) *Given a diagram $d \in \mathcal{D}$ without infinite loop. For all traces $t \in \mathcal{E}^*$, if*

$$t \in [\![d]\!]$$

then there exist trace $t' \in (\mathcal{E} \cup \mathcal{T})^*$ and $\beta \in \mathcal{B}$ such that

$$[env^!_{\mathcal{M}}.d, d] \xrightarrow{t'} [\beta, \mathsf{skip}] \text{ and } \mathcal{E}ⓢt' = t$$

Proof of theorem 5

ASSUME: $t \in [\![d]\!]$

PROVE: $\exists t', \beta : [env^!_{\mathcal{M}}.d, d] \xrightarrow{t'} [\beta, \mathsf{skip}] \wedge \mathcal{E}ⓢt' = t$

PROOF: by induction on d.

$\langle 1\rangle 1$. Induction start: $d = \mathsf{skip}$, $d = (!, m)$, $d = (?, m)$

PROOF: This carries over from theorem 2.

Induction step:

ASSUME: $t_k \in [\![d_k]\!] \Rightarrow \exists t'_k, \beta_k : [env^!_{\mathcal{M}}.d_k, d_k] \xrightarrow{t'_k} [\beta_k, \mathsf{skip}] \wedge \mathcal{E}Ⓢt'_k = t_k$ (induction hypothesis)

$\langle 1\rangle 2$. CASE: $d = d_1$ seq d_2, $d = d_1$ par d_2

PROOF: By lemmas 17 and 18 this follows from theorem 2.

$\langle 1\rangle 3$. $d = \mathsf{refuse}\ d_1$

$\langle 2\rangle 1$. $t \in [\![d_1]\!]$

PROOF: Assumption and def. (52).

$\langle 2\rangle 2$. $\exists t'_1, \beta_1 : [env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t'_1} [\beta_1, \mathsf{skip}] \ \wedge\ \mathcal{E}Ⓢt'_1 = t$

PROOF: $\langle 2\rangle 1$ and induction hypothesis.

$\langle 2\rangle 3$. LET: $t' = \langle \tau_{refuse}\rangle ^\frown t'_1$

$\langle 2\rangle 4$. $[env^!_{\mathcal{M}}.d, d] \xrightarrow{\tau_{refuse}} [env^!_{\mathcal{M}}.d, d_1] \xrightarrow{t'_1} [\beta_1, \mathsf{skip}]$

PROOF: $\langle 2\rangle 2$, $\langle 2\rangle 3$, rules (6) and (53) and $env^!_{\mathcal{M}}.d = env^!_{\mathcal{M}}.\mathsf{refuse}\ d_1 = env^!_{\mathcal{M}}.d_1$.

$\langle 2\rangle 5$. $\mathcal{E}Ⓢt' = \mathcal{E}Ⓢ\langle \tau_{refuse}\rangle ^\frown t'_1 = \mathcal{E}Ⓢt'_1 = t$

PROOF: $\langle 2\rangle 2$, $\langle 2\rangle 3$, def. of Ⓢ and $\tau_{refuse} \notin \mathcal{E}$.

$\langle 2\rangle 6$. Q.E.D.

PROOF: $\langle 2\rangle 3$, $\langle 2\rangle 4$ and $\langle 2\rangle 5$.

$\langle 1\rangle 4$. $d = d_1$ alt d_2

$\langle 2\rangle 1$. $t \in [\![d_1]\!] \vee t \in [\![d_2]\!]$

PROOF: $\langle 1\rangle 4$ and def. (40).

$\langle 2\rangle 2$. $\exists t'_k, \beta_k : [env^!_{\mathcal{M}}.d_k, d_k] \xrightarrow{t'_k} [\beta_k, \mathsf{skip}] \ \wedge\ \mathcal{E}Ⓢt'_k = t$ for $k = 1$ or $k = 2$

PROOF: $\langle 2\rangle 1$ and induction hypothesis.

$\langle 2\rangle 3$. $[env^!_{\mathcal{M}}.d, d] \xrightarrow{\tau_{alt}} [env^!_{\mathcal{M}}.d, d_k] \xrightarrow{t'_k} [\beta_k, \mathsf{skip}]$ for $k = 1$ or $k = 2$

PROOF: $\langle 2\rangle 2$, rules (6) and (13), and $env^!_{\mathcal{M}}.d = env^!_{\mathcal{M}}.d_1 \cup env^!_{\mathcal{M}}.d_2$, $env^!_{\mathcal{M}}.d_1 \cap env^!_{\mathcal{M}}.d_2 = \emptyset$.

$\langle 2\rangle 4$. LET: $t' = \langle \tau_{alt}\rangle ^\frown t'_k$ for the appropriate choice of $k \in \{1, 2\}$

$\langle 2\rangle 5$. $\mathcal{E}Ⓢt' = \mathcal{E}Ⓢ\langle \tau_{alt}\rangle ^\frown t'_k = \mathcal{E}Ⓢt'_k = t$

PROOF: $\langle 2\rangle 2$, $\langle 2\rangle 4$, def. of Ⓢ and $\tau_{alt} \notin \mathcal{E}$.

$\langle 2\rangle 6$. Q.E.D.

PROOF: $\langle 2\rangle 3$, $\langle 2\rangle 4$ and $\langle 2\rangle 5$.

$\langle 1\rangle 5$. $d = d_1$ xalt d_2

$\langle 2\rangle 1$. $t \in [\![d_1]\!] \vee t \in [\![d_2]\!]$

PROOF: $\langle 1\rangle 5$ and def. (41).

$\langle 2\rangle 2$. $\exists t'_k, \beta_k : [env^!_{\mathcal{M}}.d_k, d_k] \xrightarrow{t'_k} [\beta_k, \mathsf{skip}] \ \wedge\ \mathcal{E}Ⓢt'_k = t$ for $k = 1$ or $k = 2$

PROOF: $\langle 2\rangle 1$ and induction hypothesis.

$\langle 2\rangle 3$. $[env^!_{\mathcal{M}}.d, d] \xrightarrow{\tau_{xalt}} [env^!_{\mathcal{M}}.d, d_k] \xrightarrow{t'_k} [\beta_k, \mathsf{skip}]$ for $k = 1$ or $k = 2$

PROOF: $\langle 2\rangle 2$, rules (6) and (14), and $env^!_{\mathcal{M}}.d = env^!_{\mathcal{M}}.d_1 \cup env^!_{\mathcal{M}}.d_2$, $env^!_{\mathcal{M}}.d_1 \cap env^!_{\mathcal{M}}.d_2 = \emptyset$.

$\langle 2\rangle 4$. LET: $t' = \langle \tau_{xalt}\rangle ^\frown t'_k$ for the appropriate choice of $k \in \{1, 2\}$

$\langle 2\rangle 5$. $\mathcal{E}Ⓢt' = \mathcal{E}Ⓢ\langle \tau_{xalt}\rangle ^\frown t'_k = \mathcal{E}Ⓢt'_k = t$

PROOF: $\langle 2\rangle 2$, $\langle 2\rangle 4$, def. of Ⓢ and $\tau_{xalt} \notin \mathcal{E}$.

⟨2⟩6. Q.E.D.
PROOF: ⟨2⟩3, ⟨2⟩4 and ⟨2⟩5.

⟨1⟩6. $d = \mathsf{loop}\langle n\rangle\ d_1$
PROOF: by induction on n

⟨2⟩1. Induction start: $n = 1$

⟨3⟩1. $t \in [\![\, \mathsf{loop}\langle 1\rangle\ d_1 \,]\!] = \mu_1[\![\, d_1 \,]\!] = [\![\, d_1 \,]\!]$
PROOF: ⟨1⟩6, ⟨2⟩1 and defs. (45), (54).

⟨3⟩2. $\exists t_1', \beta_1 : [env^!_{\mathcal{M}}.d_1, d_1] \xrightarrow{t_1'} [\beta_1, \mathsf{skip}]\ \wedge\ \mathcal{E}Ⓢt_1' = t$
PROOF: ⟨3⟩1 and induction hypothesis.

⟨3⟩3. $[env^!_{\mathcal{M}}.\mathsf{loop}\langle 1\rangle\ d_1, \mathsf{loop}\langle 1\rangle\ d_1] \xrightarrow{\tau_{loop}}$
$[env^!_{\mathcal{M}}.\mathsf{loop}\langle 1\rangle\ d_1, d_1\ \mathsf{seq}\ \mathsf{loop}\langle 0\rangle\ d_1] \xrightarrow{t_1'} [\beta_1, \mathsf{skip}]$
PROOF: ⟨3⟩2 and $d_1\ \mathsf{seq}\ \mathsf{loop}\langle 0\rangle\ d_1 = d_1\ \mathsf{seq}\ \mathsf{skip} = d_1$, $env^!_{\mathcal{M}}.\mathsf{loop}\langle 1\rangle\ d_1 = env^!_{\mathcal{M}}.d_1 \cup env^!_{\mathcal{M}}.\mathsf{skip} = env^!_{\mathcal{M}}.d_1$.

⟨3⟩4. LET: $t' = \langle \tau_{loop}\rangle ^\frown t_1'$

⟨3⟩5. $\mathcal{E}Ⓢt' = \mathcal{E}Ⓢ\langle \tau_{loop}\rangle ^\frown t_1' = \mathcal{E}Ⓢt_1' = t$
PROOF: ⟨3⟩2, ⟨3⟩4, def. of Ⓢ and $\tau_{loop} \notin \mathcal{E}$.

⟨3⟩6. Q.E.D.
PROOF: ⟨3⟩3, ⟨3⟩4 and ⟨3⟩5.

⟨2⟩2. Induction step: $n = k + 1$

ASSUME: $s \in [\![\, \mathsf{loop}\langle k\rangle\ d_1 \,]\!] \Rightarrow \exists s', \beta' : [env^!_{\mathcal{M}}.\mathsf{loop}\langle k\rangle\ d_1, \mathsf{loop}\langle k\rangle\ d_1] \xrightarrow{s'} [\beta', \mathsf{skip}] \wedge \mathcal{E}Ⓢs' = s$ (induction hypothesis 2)

⟨3⟩1. $t \in [\![\, \mathsf{loop}\langle k+1\rangle\ d_1 \,]\!] = \mu_{k+1}[\![\, d_1 \,]\!] = [\![\, d_1 \,]\!] \succsim \mu_k[\![\, d_1 \,]\!]$
$= [\![\, d_1 \,]\!] \succsim [\![\, \mathsf{loop}\langle k\rangle\ d_1 \,]\!] = [\![\, d_1\ \mathsf{seq}\ \mathsf{loop}\langle k\rangle\ d_1 \,]\!]$
PROOF: Assumption, ⟨2⟩2 and definitions (54), (45), (34), (35) and (36).

⟨3⟩2. $\exists t'', \beta' : [env^!_{\mathcal{M}}.(d_1\ \mathsf{seq}\ \mathsf{loop}\langle k\rangle\ d_1), d_1\ \mathsf{seq}\ \mathsf{loop}\langle k\rangle\ d_1] \xrightarrow{t''} [\beta', \mathsf{skip}] \wedge$
$t = \mathcal{E}Ⓢt''$
PROOF: ⟨1⟩2, ⟨3⟩1 and induction hypotheses.

⟨3⟩3. $[env^!_{\mathcal{M}}.\mathsf{loop}\langle k+1\rangle\ d_1, \mathsf{loop}\langle k+1\rangle\ d_1] \xrightarrow{\tau_{loop}}$
$[env^!_{\mathcal{M}}.\mathsf{loop}\langle k+1\rangle\ d_1, d_1\ \mathsf{seq}\ \mathsf{loop}\langle k\rangle\ d_1] \xrightarrow{t''} [\beta', \mathsf{skip}]$
PROOF: ⟨3⟩2, rules (6) and (18), and $env^!_{\mathcal{M}}.(d_1\ \mathsf{seq}\ \mathsf{loop}\langle k\rangle\ d_1) = env^!_{\mathcal{M}}.d_1 \cup env^!_{\mathcal{M}}.\mathsf{loop}\langle k\rangle\ d_1 = env^!_{\mathcal{M}}.\mathsf{loop}\langle k+1\rangle\ d_1$.

⟨3⟩4. LET: $t' = \langle \tau_{loop}\rangle ^\frown t''$

⟨3⟩5. $\mathcal{E}Ⓢt' = \mathcal{E}Ⓢ\langle \tau_{loop}\rangle ^\frown t'' = \mathcal{E}Ⓢt'' = t$
PROOF: ⟨3⟩2, ⟨3⟩4, def. of Ⓢ and $\tau_{loop} \notin \mathcal{E}$.

⟨3⟩6. Q.E.D.
PROOF: ⟨3⟩3, ⟨3⟩4 and ⟨3⟩5.

⟨2⟩3. Q.E.D.
PROOF: ⟨2⟩1 and ⟨2⟩2.

⟨1⟩7. Q.E.D.
PROOF: ⟨1⟩1-⟨1⟩6.

□

B.4 Sequence diagrams with infinite loops

In this section we prove soundness and completeness of diagrams that also contain the infinite loop, $\mathsf{loop}\langle\infty\rangle\ d$. As in the previous sections we treat the denotation $[\![d]\!]$ of a diagram d as a flat set, i.e. we write $t \in [\![d]\!]$ for $t \in \bigcup_{(p,n)\in[\![d]\!]}(p \cup n)$. This means we disregard negative behavior and interaction obligations. For simplicity we assume that $env.d = \emptyset$.

We let d^k for $k \in \mathbb{N}$ denote the diagram d with every signal s renamed to s^k and apply the loop with implicit counter $\mathsf{loop}^c\langle n\rangle\ d$ as described by rule (55) and definition (56). Further we define:

$$\begin{aligned} [\![d]\!]^{\leq 1} &\overset{\mathsf{def}}{=} [\![d^1]\!] \\ [\![d]\!]^{\leq n+1} &\overset{\mathsf{def}}{=} [\![d]\!]^{\leq n} \succsim [\![d^{n+1}]\!] \textbf{ for } n \geq 1 \end{aligned} \tag{57}$$

Let $\mathcal{U} = \mathbb{P}(\mathcal{H})$ be the powerset of $\mathcal{H}$. We have that $[\![d]\!] \in \mathcal{U}$ and define the chains of trace sets of $[\![d]\!]$ as

$$chains([\![d]\!]) \overset{\mathsf{def}}{=} \{\bar{u} \in \mathcal{U}^\infty \mid \bar{u}[1] = [\![d^1]\!] \wedge \forall j \in \mathbb{N} : \bar{u}[j+1] = \bar{u}[j] \succsim [\![d^{j+1}]\!]\}$$

Because we treat $[\![d]\!]$ as a flat set, we see that $chains([\![d]\!])$ is a singleton set $\{\bar{u}\}$ where

$$\bar{u} = [\![d]\!]^{\leq 1}, [\![d]\!]^{\leq 2}, [\![d]\!]^{\leq 3}, \ldots$$

which means we have that $\bar{u}[j] = [\![d]\!]^{\leq j}$ for all $j \in \mathbb{N}$. The function pos may then be simplified to

$$pos(\bar{u}) \overset{\mathsf{def}}{=} \{\bar{t} \in \mathcal{H}^\infty \mid \forall j \in \mathbb{N} : \bar{t}[j] \in \bar{u}[j] \wedge \exists h \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \succsim \{h\}\}$$

The approximation $\sqcup\bar{u}$ becomes

$$\sqcup\bar{u} \overset{\mathsf{def}}{=} \bigcup_{\bar{t}\in pos(\bar{u})} \sqcup\bar{t}$$

where $\sqcup\bar{t}$ is defined as in appendix A. Because $chains([\![d]\!])$ is a singleton set we can let

$$\mu_\infty[\![d]\!] = \sqcup\bar{u} \quad \textbf{for} \quad chains([\![d]\!]) = \{\bar{u}\}$$

which means:

$$\mu_\infty[\![d]\!] = \bigcup_{\bar{t}\in pos(\bar{u})} \sqcup\bar{t} \quad \textbf{for} \quad chains([\![d]\!]) = \{\bar{u}\}$$

In fact, if we define

$$\begin{aligned} &tra([\![d]\!]) \overset{\mathsf{def}}{=} \\ &\quad \{\bar{t} \in \mathcal{H}^\infty \mid \forall j \in \mathbb{N} : \bar{t}[j] \in [\![d]\!]^{\leq j} \wedge \exists h \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \succsim \{h\}\} \end{aligned} \tag{58}$$

we get that

$$\mu_\infty[\![d]\!] \overset{\mathsf{def}}{=} \bigcup_{\bar{t}\in tra([\![d]\!])} \sqcup\bar{t} \tag{59}$$

In the following we use this definition of $\mu_\infty[\![d]\!]$.

We start by assuming that the body d of an infinite loop $\mathsf{loop}\langle\infty\rangle\ d$ only characterizes finite behavior, i.e., we have no nesting of infinite loops. Further we assume that $\mathsf{loop}\langle\infty\rangle$ is the outermost operator of a diagram. When soundness and completeness of this case are established we consider combinations of infinite loop with other operators.

Lemma 19 $t \in [\![\mathsf{loop}\langle\infty\rangle\ d]\!] \iff t \in \mathcal{H} \wedge \exists \bar{t} \in \mathcal{H}^\infty : (\forall j \in \mathbb{N} : (\bar{t}[j] \in [\![d]\!]^{\leq j} \wedge$ $\exists h \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \succsim \{h\}) \wedge \forall l \in \mathcal{L} : e.l \circledS t = \sqcup_l \bar{t})$

Proof of lemma 19

PROVE: $t \in [\![\,\mathsf{loop}\langle\infty\rangle\ d\,]\!] \iff t \in \mathcal{H} \wedge \exists \bar{t} \in \mathcal{H}^\infty : (\forall j \in \mathbb{N} : (\bar{t}[j] \in [\![\,d\,]\!]^{\le j} \wedge \exists h \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \precsim \{h\}) \wedge \forall l \in \mathcal{L} : e.l \circledS t = \sqcup_l \bar{t})$

$\langle 1\rangle 1.$ $t \in [\![\,\mathsf{loop}\langle\infty\rangle\ d\,]\!] = \mu_\infty [\![\,d\,]\!] = \bigcup_{\bar{t} \in tra([\![\,d\,]\!])} \sqcup \bar{t}$

PROOF: Defs. (54) and (59).

$\langle 1\rangle 2.$ $t \in \sqcup \bar{t}$ for some $\bar{t} \in tra([\![\,d\,]\!])$

PROOF: $\langle 1\rangle 1$.

$\langle 1\rangle 3.$ $t \in \mathcal{H} \wedge \forall l \in \mathcal{L} : e.l \circledS t = \sqcup_l \bar{t}$ for some $\bar{t}$ such that $\bar{t} \in \mathcal{H}^\infty \wedge \forall j \in \mathbb{N} : \bar{t}[j] \in [\![\,d\,]\!]^{\le j} \wedge \exists h \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \precsim \{h\}$

PROOF: $\langle 1\rangle 2$ and defs. (49) and (58).

$\langle 1\rangle 4.$ Q.E.D.

PROOF: $\langle 1\rangle 3$.

□

Lemma 20 *Given diagram $d \in \mathcal{D}$ without infinite loops. If $[\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d]$ may produce trace t, then, for any k, there exists a trace t' such that $[\emptyset, d^1\ \mathsf{seq}\ d^2\ \mathsf{seq}\ \cdots\ \mathsf{seq}\ d^k\ \mathsf{seq}\ \mathsf{loop}^{k+1}\langle\infty\rangle\ d]$ may produce t' and $\mathcal{E} \circledS t = \mathcal{E} \circledS t'$.*

Proof of lemma 20

ASSUME: t is produced by $[\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d]$

PROVE: There exists a trace t' such that t' is produced by $[\emptyset, d^1\ \mathsf{seq}\ d^2\ \mathsf{seq}\ \cdots\ \mathsf{seq}\ d^k\ \mathsf{seq}\ \mathsf{loop}^{k+1}\langle\infty\rangle\ d]$ and $\mathcal{E} \circledS t = \mathcal{E} \circledS t'$.

PROOF: by induction on k

$\langle 1\rangle 1.$ CASE: $k = 1$
(induction start)

$\langle 2\rangle 1.$ The only possible execution step from $[\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d]$ is $[\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d] \xrightarrow{\tau_{loop}} [\emptyset, d^1\ \mathsf{seq}\ \mathsf{loop}^2\langle\infty\rangle\ d]$

PROOF: Rules (6) and (55).

$\langle 2\rangle 2.$ There exists trace s such that s is produced by $[\emptyset, d^1\ \mathsf{seq}\ \mathsf{loop}^2\langle\infty\rangle\ d]$ and $t = \langle \tau_{loop} \rangle ^\frown s$

PROOF: Assumption and $\langle 2\rangle 1$.

$\langle 2\rangle 3.$ LET: $t' = s$

$\langle 2\rangle 4.$ $\mathcal{E} \circledS t = \mathcal{E} \circledS (\langle \tau_{loop} \rangle ^\frown t') = \mathcal{E} \circledS t'$

PROOF: $\langle 2\rangle 1$, $\langle 2\rangle 2$, $\langle 2\rangle 3$, def. of $\circledS$ and $\tau_{loop} \notin \mathcal{E}$.

$\langle 2\rangle 5.$ Q.E.D.

PROOF: $\langle 2\rangle 4$.

$\langle 1\rangle 2.$ CASE: $k = j + 1$ (induction step)

ASSUME: t' is produced by $[\emptyset, d^1\ \mathsf{seq}\ d^2\ \mathsf{seq}\ \cdots\ \mathsf{seq}\ d^j\ \mathsf{seq}\ \mathsf{loop}^{j+1}\langle\infty\rangle\ d]$ and $\mathcal{E} \circledS t = \mathcal{E} \circledS t'$ (induction hypothesis)

PROVE: t'' is produced by
$[\emptyset, d^1\ \mathsf{seq}\ d^2\ \mathsf{seq}\ \cdots\ \mathsf{seq}\ d^j\ \mathsf{seq}\ d^{j+1}\ \mathsf{seq}\ \mathsf{loop}^{j+2}\langle\infty\rangle\ d]$
and $\mathcal{E} \circledS t = \mathcal{E} \circledS t''$

$\langle 2\rangle 1.$ There exist traces s, s', diagram $d' \in \mathcal{D}$, and $\beta \in \mathcal{B}$ such that

$$[\emptyset, d^1\ \mathsf{seq}\ d^2\ \mathsf{seq}\ \cdots\ \mathsf{seq}\ d^j\ \mathsf{seq}\ \mathsf{loop}^{j+1}\langle\infty\rangle \xrightarrow{s}$$
$$[\beta, d'\ \mathsf{seq}\ \mathsf{loop}^{j+1}\langle\infty\rangle\ d] \xrightarrow{\tau_{loop}}$$
$$[\beta, d'\ \mathsf{seq}\ d^{j+2}\ \mathsf{seq}\ \mathsf{loop}^{j+1}\langle\infty\rangle\ d] \xrightarrow{s'}$$

and $t' = s ^\frown \langle \tau_{loop} \rangle ^\frown s'$

PROOF: Induction hypothesis and rules (6), (9), (10) and (55).

$\langle 2\rangle 2.$ $[\emptyset, d^1 \textsf{ seq } d^2 \textsf{ seq } \cdots \textsf{ seq } d^j \textsf{ seq } d^{j+1}\textsf{loop}^{j+2}\langle\infty\rangle\ d] \xrightarrow{s}$
$[\beta, d' \textsf{ seq } d^{j+1} \textsf{ seq loop}^{j+2}\langle\infty\rangle\ d] \xrightarrow{s'}$

PROOF: $\langle 2\rangle 1$.

$\langle 2\rangle 3.$ LET: $t'' = s ^\frown s'$

$\langle 2\rangle 4.$ $\mathcal{E} \circledS t' = \mathcal{E} \circledS (s ^\frown \langle \tau_{loop}\rangle ^\frown s') = \mathcal{E} \circledS (s ^\frown s') = \mathcal{E} \circledS t''$

PROOF: $\langle 2\rangle 1$, $\langle 2\rangle 3$, properties of $\circledS$ and $\tau_{loop} \notin \mathcal{E}$.

$\langle 2\rangle 5.$ $\mathcal{E} \circledS t = \mathcal{E} \circledS t''$

PROOF: Induction hypothesis and $\langle 2\rangle 4$.

$\langle 2\rangle 6.$ Q.E.D.

PROOF: $\langle 2\rangle 2$, $\langle 2\rangle 4$ and $\langle 2\rangle 5$.

$\langle 1\rangle 3.$ Q.E.D.

PROOF: $\langle 1\rangle 1$ and $\langle 1\rangle 2$.

□

Theorem 6 (Soundness) *Let d be a diagram without infinite loops. If $[\emptyset, \textsf{loop}\langle\infty\rangle\ d]$ produces the trace t, then $\mathcal{E} \circledS t \in [\![\ \textsf{loop}\langle\infty\rangle\ d\ ]\!]$.*

Proof of theorem 6

ASSUME: 1. d is a diagram without infinite loops
2. $[\emptyset, \textsf{loop}^1\langle\infty\rangle\ d]$ produces trace t

LET: $t' = \mathcal{E} \circledS t$

PROVE: $t' \in [\![\ \textsf{loop}\langle\infty\rangle\ d\ ]\!]$

$\langle 1\rangle 1.$ $t' \in [\![\ \textsf{loop}\langle\infty\rangle\ d\ ]\!] \iff t' \in \mathcal{H} \wedge \exists \bar{t} \in \mathcal{H}^\infty : (\forall j \in \mathbb{N} : (\bar{t}[j] \in [\![\ d\ ]\!]^{\leq j} \wedge \exists h \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \succsim \{h\}) \wedge \forall l \in \mathcal{L} : e.l \circledS t' = \sqcup_l \bar{t})$

PROOF: Lemma 19.

$\langle 1\rangle 2.$ $t' \in \mathcal{H} \wedge \exists \bar{t} \in \mathcal{H}^\infty : (\forall j \in \mathbb{N} : (\bar{t}[j] \in [\![\ d\ ]\!]^{\leq j} \wedge \exists h \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \succsim \{h\}) \wedge \forall l \in \mathcal{L} : e.l \circledS t' = \sqcup_l \bar{t})$

PROOF: by contradiction

ASSUME: $t' \notin \mathcal{H} \vee \forall \bar{t} \in \mathcal{H}^\infty : (\exists j \in \mathbb{N} : (\bar{t}[j] \notin [\![\ d\ ]\!]^{\leq j} \vee \forall h \in \mathcal{H} : \bar{t}[j+1] \notin \{\bar{t}[j]\} \succsim \{h\}) \vee \exists l \in \mathcal{L} : e.l \circledS t' \neq \sqcup_l \bar{t})$

PROVE: Contradiction

$\langle 2\rangle 1.$ CASE: $t' \notin \mathcal{H}$

PROOF: We have assumed that all messages are unique, so this must mean that there exist a message m, traces s, s', diagrams d_1, d_2, d_3, d_4 and states of the communication medium $\beta'_1, \beta'_2, \beta'_3, \beta'_4$ such that

$$[\emptyset, \textsf{loop}^1\langle\infty\rangle\ d] \xrightarrow{s} [\beta'_1, d_1] \xrightarrow{(?,m)} [\beta'_2, d_2] \xrightarrow{s'} [\beta'_3, d_3] \xrightarrow{(!,m)} [\beta'_4, d_4]$$

and

$$\mathcal{E} \circledS (s ^\frown \langle (?,m)\rangle ^\frown s' ^\frown \langle (!,m)\rangle) \sqsubseteq t'$$

By rules (3) and (8) this is clearly not possible and we must have that $t' \in \mathcal{H}$ and the case assumption is impossible.

$\langle 2\rangle 2.$ CASE: $\forall \bar{t} \in \mathcal{H}^\infty : (\exists j \in \mathbb{N} : (\bar{t}[j] \notin [\![\ d\ ]\!]^{\leq j} \vee \forall h \in \mathcal{H} : \bar{t}[j+1] \notin \{\bar{t}[j]\} \succsim \{h\}) \vee \exists l \in \mathcal{L} : e.l \circledS t' \neq \sqcup_l \bar{t})$

$\langle 3\rangle 1.$ LET: $s_1, s_2, s_3, \ldots, s_i, \ldots$ be traces and $\beta_1, \beta_2, \beta_3, \ldots, \beta_i, \ldots$ be states of the communication medium such that

$[\emptyset, d^1] \xrightarrow{s_1} [\beta_1, \mathsf{skip}]$

$[\emptyset, d^1 \text{ seq } d^2] \xrightarrow{s_1} [\beta_1, d^2] \xrightarrow{s_2} [\beta_2, \mathsf{skip}]$

$[\emptyset, d^1 \text{ seq } d^2 \text{ seq } d^3] \xrightarrow{s_1 \frown s_2} [\beta_2, d^3] \xrightarrow{s_3} [\beta_3, \mathsf{skip}]$

$\vdots$

$[\emptyset, d^1 \text{ seq } d^2 \cdots \text{ seq } d^i] \xrightarrow{s_1 \frown s_2 \frown \cdots \frown s_{i-1}} [\beta_{i-1}, d^i] \xrightarrow{s_i} [\beta_i, \mathsf{skip}]$

$\vdots$

where we choose $s_1, s_2, \ldots, s_i, \ldots$ such that $ev.d^i Ⓢ s_i = ev.d^i Ⓢ t$. Further we let $t_1, t_2, t_3, \ldots, t_i, \ldots$ be traces such that

$$\begin{aligned} t_1 &= \mathcal{E} Ⓢ s_1 \\ t_2 &= t_1 \frown (\mathcal{E} Ⓢ s_2) \\ t_3 &= t_2 \frown (\mathcal{E} Ⓢ s_3) \\ &\vdots \\ t_i &= t_{i-1} \frown (\mathcal{E} Ⓢ s_i) \\ &\vdots \end{aligned}$$

⟨3⟩2. $t_j \in [\![d]\!]^{\leq j}$ for all $j \in \mathbb{N}$
PROOF: ⟨3⟩1 and theorem 4.

⟨3⟩3. LET: $\bar{t} = t_1, t_2, t_3, \ldots, t_i, \ldots$

⟨3⟩4. CASE: $\exists j \in \mathbb{N} : (\bar{t}[j] \notin [\![d]\!]^{\leq j} \vee \forall h \in \mathcal{H} : \bar{t}[j+1] \notin \{\bar{t}[j]\} \precsim \{h\})$

⟨4⟩1. Choose arbitrary $j \in \mathbb{N}$

⟨4⟩2. CASE: $\bar{t}[j] \notin [\![d]\!]^{\leq j}$

⟨5⟩1. $\bar{t}[j] \in [\![d]\!]^{\leq j}$
PROOF: By ⟨3⟩3, $\bar{t}[j] = t_j$ and by ⟨3⟩2, $t_j \in [\![d]\!]^{\leq j}$.

⟨5⟩2. Q.E.D.
PROOF: By ⟨5⟩1, ⟨4⟩2 is impossible.

⟨4⟩3. CASE: $\forall h \in \mathcal{H} : \bar{t}[j+1] \notin \{\bar{t}[j]\} \precsim \{h\}$

⟨5⟩1. $\bar{t}[j+1] = t_{j+1} = t_j \frown (\mathcal{E} Ⓢ s_{j+1}) = \bar{t}[j] \frown (\mathcal{E} Ⓢ s_{j+1})$
PROOF: ⟨3⟩1 and ⟨3⟩3.

⟨5⟩2. $\bar{t}[j+1] \in \{\bar{t}[j]\} \precsim \{\mathcal{E} Ⓢ s_{j+1}\}$
PROOF: ⟨5⟩1.

⟨5⟩3. $\mathcal{E} Ⓢ s_{j+1} \in \mathcal{H}$
PROOF: By ⟨3⟩1, $[\beta_j, d^{j+1}] \xrightarrow{s_{j+1}} [\beta_{j+1}, \mathsf{skip}]$, so this follows from lemma 6 and theorem 4.

⟨5⟩4. Q.E.D.
PROOF: Let $h = \mathcal{E} Ⓢ s_{j+1}$. By ⟨5⟩2 and ⟨5⟩3 it is clear that ⟨4⟩3 is impossible.

⟨4⟩4. Q.E.D.
PROOF: Since j is chosen arbitrarily, and ⟨4⟩2 and ⟨4⟩3 are impossible for this arbitrarily chosen j, ⟨3⟩4 is impossible.

⟨3⟩5. CASE: $\exists l \in \mathcal{L} : e.l Ⓢ t' \neq \sqcup_l \bar{t}$
This means that there must exist l such that $e.l Ⓢ t' \sqsubset \sqcup_l \bar{t}$ or $\sqcup_l \bar{t} \sqsubset e.l Ⓢ t'$ or $\exists j \in \mathbb{N} : (e.l Ⓢ t')[j] \neq (\sqcup_l \bar{t})[j]$.

⟨4⟩1. Choose arbitrary $l \in \mathcal{L}$.

⟨4⟩2. For any k, $e.l Ⓢ s_1 \frown e.l Ⓢ s_2 \frown \cdots \frown e.l Ⓢ s_k \sqsubseteq e.l Ⓢ t$

⟨5⟩1. There exists t'' such that t'' is produced by $[\emptyset, d^1 \ \mathsf{seq}\ d^2\ \mathsf{seq} \cdots \mathsf{seq}\ d^k\ \mathsf{seq}\ \ \mathsf{loop}^{k+1}\langle\infty\rangle\ d]$ and $\mathcal{E}Ⓢt = \mathcal{E}Ⓢt''$

PROOF: Lemma 20.

⟨5⟩2. $(e.l \cap ev.d^1)Ⓢs_1{}^\frown(e.l \cap ev.d^2)Ⓢs_2{}^\frown \cdots {}^\frown(e.l \cap d^k)Ⓢs_k$
$= e.l \cap (ev.d^1 \cup ev.d^2 \cup \cdots \cup ev.d^k)Ⓢt''$

PROOF: ⟨3⟩1, $ev.d^m \cap ev.d^n = \emptyset$ for $m \neq n$, and rules (3), (9) and (10) (which handles the ordering on l).

⟨5⟩3. $(e.l \cap ev.d^1)Ⓢs_1{}^\frown(e.l \cap ev.d^2)Ⓢs_2{}^\frown \cdots {}^\frown(e.l \cap ev.d^k)Ⓢs_k$
$= e.lⓈs_1{}^\frown e.lⓈs_2{}^\frown \cdots {}^\frown e.lⓈs_k$

PROOF: $ev.d^iⓈs_i = \mathcal{E}Ⓢs_i$ for $1 \leq i \leq k$ (⟨3⟩1) and $e.l \subseteq \mathcal{E}$ (by def. of $e.\_$), which implies that $(e.l \cap ev.d^i)Ⓢs_i = e.lⓈ(ev.d^iⓈs_i) = e.lⓈ(\mathcal{E}Ⓢs_i) = (e.l \cap \mathcal{E})Ⓢs_i = e.lⓈs_i$.

⟨5⟩4. $e.l \cap (ev.d^1 \cup ev.d^2 \cup \cdots \cup ev.d^k)Ⓢt'' \sqsubseteq e.lⓈt''$

PROOF: $(ev.d^1 \cup ev.d^2 \cup \cdots \cup ev.d^k) \cap (ev.\mathsf{loop}^{k+1}\langle\infty\rangle\ d) = \emptyset$ and rules (3), (9) and (10).

⟨5⟩5. Q.E.D.

PROOF: $e.lⓈs_1{}^\frown e.lⓈs_2{}^\frown \cdots {}^\frown e.lⓈs_k$

$$\begin{array}{ll}
= (e.l \cap ev.d^1)Ⓢs_1{}^\frown(e.l \cap ev.d^2)Ⓢs_2{}^\frown & \\
\quad \cdots {}^\frown(e.l \cap ev.d^k)Ⓢs_k & (\langle 5\rangle 3) \\
= e.l \cap (ev.d^1 \cup ev.d^2 \cup \cdots \cup ev.d^k)Ⓢt'' & (\langle 5\rangle 2) \\
\sqsubseteq e.lⓈt'' & (\langle 5\rangle 4) \\
= e.lⓈt & (\langle 5\rangle 1 \text{ and } e.l \subseteq \mathcal{E})
\end{array}$$

⟨4⟩3. $e.lⓈt' = e.lⓈt$

PROOF: Assumption that $t' = \mathcal{E}Ⓢt$ and $e.l \subseteq \mathcal{E}$.

⟨4⟩4. CASE: $e.lⓈt' \sqsubset \sqcup_l \bar{t}$

⟨5⟩1. $\exists h \in \mathcal{H} : h \neq \langle\rangle \wedge \sqcup_l \bar{t} = (e.lⓈt')^\frown h$

PROOF: Case assumption (⟨4⟩4) and the properties of $\sqsubset$.

⟨5⟩2. $\#(e.lⓈt') \neq \infty$ (i.e. $e.lⓈt'$ is finite)

PROOF: ⟨5⟩1 and $\#s = \infty \Rightarrow s^\frown r = s$

⟨5⟩3. LET: k be the smallest number such that

$$e.lⓈt_k = e.lⓈs_1{}^\frown e.lⓈs_2{}^\frown \cdots {}^\frown e.lⓈs_k = e.lⓈt'$$

PROOF: By ⟨4⟩2, ⟨4⟩3 and ⟨5⟩2 such k must exist (because at some point the s_i stops containing events from $e.l$ or else $e.lⓈt'$ would not be finite).

⟨5⟩4. $e.lⓈs_i = \langle\rangle$ for all $i > k$

PROOF: This follows direclty from ⟨5⟩3.

⟨5⟩5. $\sqcup_l \bar{t} = e.lⓈt_k$

PROOF: By ⟨3⟩1, ⟨3⟩3 and def. of $\sqcup_l \bar{t}$, $\sqcup_l \bar{t}$ is the least upper bound of

$$e.lⓈt_1, e.lⓈt_2, \ldots, e.lⓈt_k, e.lⓈt_{k+1}, \ldots = e.lⓈs_1, e.lⓈ(s_1{}^\frown s_2), \ldots,$$
$$e.lⓈ(s_1{}^\frown s_2{}^\frown \cdots {}^\frown s_k), e.lⓈ(s_1{}^\frown s_2{}^\frown \cdots {}^\frown s_k{}^\frown s_{k+1}), \ldots$$

By ⟨5⟩4 we must have that $\sqcup_l \bar{t} = e.lⓈt_k$ or else it would not be the *least* upper bound.

⟨5⟩6. $\sqcup_l \bar{t} = e.lⓈt'$

PROOF: ⟨5⟩3 and ⟨5⟩5.

⟨5⟩7. Q.E.D.

PROOF: ⟨5⟩5 contradicts ⟨5⟩1, and hence $e.lⓈt' \sqsubset \sqcup_l \bar{t}$ is impossible.

$\langle 4\rangle 5$. CASE: $\sqcup_l \bar{t} \sqsubset e.l \circledS t'$

$\langle 5\rangle 1$. $\exists h \in \mathcal{H} : h \neq \langle\rangle \wedge e.l \circledS t' = (\sqcup_l \bar{t})^\frown h$
PROOF: Case assumption ($\langle 4\rangle 5$) and properties of $\sqsubset$.

$\langle 5\rangle 2$. $\#(\sqcup_l \bar{t}) \neq \infty$ (i.e. $\sqcup_l \bar{t}$ is finite)
PROOF: $\langle 5\rangle 1$ and $\#s = \infty \Rightarrow s^\frown r = s$.

$\langle 5\rangle 3$. There exists k such that $e.l \circledS t_k = e.l \circledS t_i$ for all $i > k$
PROOF: By $\langle 5\rangle 2$, $\sqcup_l \bar{t}$ is finite and by def. $e.l \circledS t_i \sqsubseteq \sqcup_l \bar{t}$ for all i, so there exists a finite bound on $e.l \circledS t_i$ and a place where the $e.l \circledS t_i$ for $i \in \mathbb{N}$ stop growing.

$\langle 5\rangle 4$. $e.l \circledS s_i = \langle\rangle$ for all $i > k$
PROOF: $\langle 3\rangle 1$ and $\langle 5\rangle 3$.

$\langle 5\rangle 5$. $\sqcup_l \bar{t} = e.l \circledS t_k$
PROOF: $\langle 3\rangle 3$, $\langle 5\rangle 3$ and the def. of $\sqcup_l \bar{t}$ as the least upperbound of

$$e.l \circledS \bar{t}[1], e.l \circledS \bar{t}[2], \ldots, e.l \circledS \bar{t}[k], \ldots$$

$\langle 5\rangle 6$. LET: $h[1] = e$. Then $(\sqcup_l \bar{t})^\frown \langle e\rangle \sqsubseteq e.l \circledS t'$
PROOF: This follows directly from $\langle 5\rangle 1$.

$\langle 5\rangle 7$. There exists $n \in \mathbb{N}$ such that $t'[n] = e$
PROOF: $\langle 5\rangle 6$.

$\langle 5\rangle 8$. There exists $j \in \mathbb{N}$ such that $e \in ev.d^j$
PROOF: $\langle 5\rangle 7$, assumption 2, the assumption that $t' = \mathcal{E} \circledS t$ and lemma 20.

$\langle 5\rangle 9$. $e.l \circledS t_j \sqsubseteq e.l \circledS t'$
PROOF: $\langle 4\rangle 2$ and $\langle 4\rangle 3$.

$\langle 5\rangle 10$. $j > k$
PROOF: By $\langle 5\rangle 5$ and $\langle 5\rangle 6$, $(e.l \circledS t_k)^\frown \langle e\rangle \sqsubseteq e.l \circledS t'$, so by $\langle 5\rangle 8$ and $\langle 5\rangle 9$ this must be the case (or else rule (10) is violated).

$\langle 5\rangle 11$. $(e.l \circledS t_k)^\frown \langle e\rangle \sqsubseteq e.l \circledS t_j = e.l \circledS s_1{}^\frown \cdots {}^\frown e.l \circledS s_k{}^\frown \cdots {}^\frown e.l \circledS s_j$
PROOF: $\langle 3\rangle 1$, $\langle 4\rangle 3$, $\langle 5\rangle 5$, $\langle 5\rangle 6$, $\langle 5\rangle 9$ and $\langle 5\rangle 10$.

$\langle 5\rangle 12$. $e.l \circledS s_j \neq \langle\rangle$
PROOF: $\langle 3\rangle 1$, $\langle 5\rangle 8$ and $\langle 5\rangle 11$.

$\langle 5\rangle 13$. Q.E.D.
PROOF: $\langle 5\rangle 10$ and $\langle 5\rangle 12$ contradicts $\langle 5\rangle 4$, so we must have that $\langle 4\rangle 5$ is impossible.

$\langle 4\rangle 6$. CASE: $\exists j \in \mathbb{N} : (e.l \circledS t')[j] \neq (\sqcup_l \bar{t})[j]$
ASSUME: j is the smallest such number, i.e.:

$$(e.l \circledS t')|_{j-1} = (\sqcup_l \bar{t})|_{j-1} \wedge (e.l \circledS t')[j] \neq (\sqcup_l \bar{t})[j]$$

$\langle 5\rangle 1$. LET: k be the smallest number such that $\#(e.l \circledS \bar{t}[k]) \geq j$
PROOF: Such k must exist. $\sqcup_l \bar{t}$ is defined as the least upper bound of $e.l \circledS \bar{t}[1], e.l \circledS \bar{t}[2], \ldots, e.l \circledS \bar{t}[i], \ldots$ If $\sqcup_l \bar{t}$ is finite, then by the case assumption ($\langle 4\rangle 6$), $\#(\sqcup_l \bar{t}) \geq j$, and there must exist k such that $e.l \circledS \bar{t}[k] = \sqcup_l \bar{t}$ (or else it is not the least upper bound), which implies $\#(e.l \circledS \bar{t}[k]) \geq j$. If $\sqcup_l \bar{t}$ is infinite this means there is no finite bound on the length of the $e.l \circledS \bar{t}[i]$ and we may find k such that $\#(e.l \circledS \bar{t}[k]) \geq n$ for any $n \in \mathbb{N}$.

$\langle 5\rangle 2$. $e.l \circledS \bar{t}[k] \sqsubseteq e.l \circledS t'$

$\langle 6\rangle 1$. $e.l \circledS \bar{t}[k] = e.l \circledS s_1{}^\frown e.l \circledS s_2{}^\frown \cdots {}^\frown e.l \circledS s_k$

$\langle 7\rangle 1$. $\bar{t}[k] = t_k = \mathcal{E} \circledS s_1{}^\frown \mathcal{E} \circledS s_2{}^\frown \cdots {}^\frown \mathcal{E} \circledS s_k$
PROOF: $\langle 3\rangle 1$ and $\langle 3\rangle 3$.

$\langle 7\rangle 2.$ Q.E.D.
PROOF: $\langle 7\rangle 1$ and $e.l \subseteq \mathcal{E}$.
$\langle 6\rangle 2.$ $e.l \circledS s_1 \frown e.l \circledS s_2 \frown \cdots \frown e.l \circledS s_k \sqsubseteq e.l \circledS t$
PROOF: $\langle 4\rangle 2$.
$\langle 6\rangle 3.$ Q.E.D.
PROOF: $\langle 4\rangle 3$, $\langle 6\rangle 1$ and $\langle 6\rangle 2$.
$\langle 5\rangle 3.$ $e.l \circledS \bar{t}[k] \sqsubseteq \sqcup_l \bar{t}$
PROOF: This follow directly from the definition of $\sqcup_l \bar{t}$.
$\langle 5\rangle 4.$ $(e.l \circledS t')[j] = (\sqcup_l \bar{t})[j]$
PROOF: From $\langle 5\rangle 2$ and $\langle 5\rangle 3$ we have that $e.l \circledS \bar{t}[k] \sqsubseteq e.l \circledS t'$ and $e.l \circledS \bar{t}[k] \sqsubseteq \sqcup_l \bar{t}$. By assumption $j \leq \#(e.l \circledS \bar{t}[k])$, so we must have that $(e.l \circledS t')[j] = (e.l \circledS \bar{t}[k])[j] = (\sqcup_l \bar{t})[j]$.
$\langle 5\rangle 5.$ Q.E.D.
PROOF: From $\langle 5\rangle 4$ it is clear that $\langle 4\rangle 6$ is impossible.
$\langle 4\rangle 7.$ Q.E.D.
PROOF: Because $\langle 4\rangle 4$, $\langle 4\rangle 5$ and $\langle 4\rangle 6$ is impossible for an arbitrarily chosen l, $\langle 3\rangle 5$ is also impossible.
$\langle 3\rangle 6.$ Q.E.D.
PROOF: Because $\langle 3\rangle 4$ and $\langle 3\rangle 5$ are impossible, we have found a $\bar{t}$ that contradicts $\langle 2\rangle 2$.
$\langle 2\rangle 3.$ Q.E.D.
PROOF: Because $\langle 2\rangle 1$ and $\langle 2\rangle 2$ are impossible we must have that $\langle 1\rangle 2$ is true.
$\langle 1\rangle 3.$ Q.E.D.
PROOF: $\langle 1\rangle 1$ and $\langle 1\rangle 2$.

□

Lemma 21 *Let d be a diagram without infinite loops and with $env.d = \emptyset$ (i.e. no external communication). If $t \in [\![\mathsf{loop}\langle\infty\rangle\ d]\!]$, then for all $k \in \mathbb{N} \cup \{0\}$ there exist trace t', $\beta \in \mathcal{B}$ and $d' \in \mathcal{D}$ such that*

$$[\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d] \xrightarrow{t'} [\beta, d'] \wedge \mathcal{E} \circledS t' = t|_k$$

Proof of lemma 21
ASSUME: 1. d is a diagram without infinite loops
2. $env.d = \emptyset$
3. $t \in [\![\mathsf{loop}\langle\infty\rangle\ d]\!]$
PROVE: For all $k \in \mathbb{N} \cup \{0\}$ there exist t', β, d' such that
$[\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d] \xrightarrow{t'} [\beta, d'] \wedge \mathcal{E} \circledS t' = t|_k$
PROOF: by induction on k
$\langle 1\rangle 1.$ CASE: $k = 0$ (induction start)
$\langle 2\rangle 1.$ $t|_k = t|_0 = \langle\rangle$
PROOF: $\langle 1\rangle 1$ and properties of $\_|\_$.
$\langle 2\rangle 2.$ LET: $t' = \langle \tau_{loop} \rangle$
$\langle 2\rangle 3.$ $\mathcal{E} \circledS \langle \tau_{loop} \rangle = \langle\rangle$
PROOF: $\tau_{loop} \notin \mathcal{E}$ and properties of $\_\circledS\_$.
$\langle 2\rangle 4.$ $[\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d] \xrightarrow{t'} [\emptyset, d^1\ \mathsf{seq}\ \mathsf{loop}^2\langle\infty\rangle\ d] \wedge \mathcal{E} \circledS t' = t|_k$
PROOF: This follows trivially from $\langle 2\rangle 1$, $\langle 2\rangle 2$, and rule (55).

$\langle 2\rangle$5. Q.E.D.

PROOF: This follows from $\langle 2\rangle$3 and $\langle 2\rangle$4 by letting $\beta = \emptyset$ and $d' = d^1$ seq $\mathsf{loop}^2\langle\infty\rangle\ d$.

$\langle 1\rangle$2. CASE: $k+1$ (induction step)

ASSUME: $\exists t'', \beta', d'' : [\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d] \xrightarrow{t''} [\beta', d''] \wedge \mathcal{E} \circledS t'' = t|_k$ (induction hypothesis)

PROVE: $\exists t', \beta, d' : [\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d] \xrightarrow{t'} [\beta, d'] \wedge \mathcal{E} \circledS t' = t|_{k+1}$

$\langle 2\rangle$1. LET: $t[k+1] = e$

$\langle 2\rangle$2. $t|_{k+1} = t|_k \frown \langle e\rangle \sqsubset t$

PROOF: $\langle 2\rangle$1 (here we assume t to be infinite, but the case $t|_{k+1} = t$ would not change the following argument).

$\langle 2\rangle$3. $\forall l \in \mathcal{L} : e.l \circledS t|_{k+1} = e.l \circledS (t|_k \frown \langle e\rangle) \sqsubseteq e.l \circledS t$

PROOF: $\langle 2\rangle$1, $\langle 2\rangle$2 and properties of $l.\_$ and $\_\circledS\_$.

$\langle 2\rangle$4. $t \in \mathcal{H} \wedge \exists \bar{t} \in \mathcal{H}^\infty : (\forall j \in \mathbb{N} : (\bar{t}[j] \in [\![\, d\,]\!]^{\leq j} \wedge \exists h \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \succsim \{h\}) \wedge \forall l \in \mathcal{L} : e.l \circledS t = \sqcup_l \bar{t})$

PROOF: Assumption 3 and lemma 19.

$\langle 2\rangle$5. LET: $n \in \mathbb{N}$ be the number such that $e \in d^n$ and let $m \in \mathbb{N}$ be the largest number such that $ev.d^m \cap ev.t|_k \neq \emptyset$

PROOF: By our assumption that all messages are unique and are enumerated by the iterations of the loop (cf. rule (55)), both n and m are uniquely identified.

$\langle 2\rangle$6. LET: $j = \mathbf{max}(m, n)$

$\langle 2\rangle$7. $\exists \bar{t} \in \mathcal{H}^\infty : \bar{t}[j] \in [\![\, d\,]\!]^{\leq j} \wedge \forall l \in \mathcal{L} : e.l \circledS t|_{k+1} \sqsubseteq e.l \circledS \bar{t}[j] \sqsubseteq e.l \circledS t$

$\langle 3\rangle$1. $\exists \bar{t} \in \mathcal{H}^\infty : \bar{t}[j] \in [\![\, d\,]\!]^{\leq j} \wedge \forall l \in \mathcal{L} : e.l \circledS t = \sqcup_l \bar{t}$

PROOF: $\langle 2\rangle$4.

$\langle 3\rangle$2. $\exists \bar{t} \in \mathcal{H}^\infty : \bar{t}[j] \in [\![\, d\,]\!]^{\leq j} \wedge \forall l \in \mathcal{L} : e.l \circledS \bar{t}[j] \sqsubseteq \sqcup_l \bar{t} = e.l \circledS t$

PROOF: $\langle 3\rangle$1 and def. of $\sqcup_l$.

$\langle 3\rangle$3. $\forall l \in \mathcal{L} : e.l \circledS t|_{k+1} \sqsubseteq e.l \circledS t$

PROOF: $\langle 2\rangle$3.

$\langle 3\rangle$4. $\exists \bar{t} \in \mathcal{H}^\infty : \bar{t}[j] \in [\![\, d\,]\!]^{\leq j} \wedge \forall l \in \mathcal{L} : e.l \circledS t|_{k+1} \sqsubseteq e.l \circledS \bar{t}[j]$

PROOF: By $\langle 3\rangle$2 and $\langle 3\rangle$3 we must have that, for all l, $e.l \circledS t|_{k+1} \sqsubset e.l \circledS \bar{t}[j]$, $e.l \circledS t|_{k+1} = e.l \circledS \bar{t}[j]$ or $e.l \circledS \bar{t}[j] \sqsubset e.l \circledS t|_{k+1}$. But by $\langle 2\rangle$5 and $\langle 2\rangle$6 we must have that $\#\bar{t}[j] \geq \#t|_{k+1}$, so the latter is not possible.

$\langle 3\rangle$5. Q.E.D.

PROOF: $\langle 3\rangle$2, $\langle 3\rangle$3 and $\langle 3\rangle$4.

$\langle 2\rangle$8. LET: $\bar{t}$ be such that $\bar{t} \in \mathcal{H}^\infty \wedge \bar{t}[j] \in [\![\, d\,]\!]^{\leq j} \wedge \forall l \in \mathcal{L} : e.l \circledS t|_{k+1} \sqsubseteq e.l \circledS \bar{t}[j] \sqsubseteq e.l \circledS t$

PROOF: By $\langle 2\rangle$6, such $\bar{t}$ exists.

$\langle 2\rangle$9. $\exists \beta', t'', d'' : \mathcal{E} \circledS t'' = t|_k \wedge [\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d] \xrightarrow{t''} [\beta', d''\ \mathsf{seq}\ \mathsf{loop}^{j+1}\langle\infty\rangle\ d] \wedge t''[\#t''] \in \mathcal{E}$

PROOF: This follows from the induction hypothesis, $\langle 2\rangle$5 and $\langle 2\rangle$6. (Because m is the largest number such that $ev.d^m \cap ev.t|_k \neq \emptyset$ and $j \geq m$, j iterations of the loop is sufficient to produce t''.) $t''[\#t''] \in \mathcal{E}$ is obtained by not execution silent events after the execution of $t[k]$.

$\langle 2\rangle$10. $\exists \beta', t'', d'' : \mathcal{E} \circledS t'' = t|_k \wedge [\emptyset, \mathsf{loop}^1\langle j\rangle\ d] \xrightarrow{t''} [\beta', d''] \wedge t''[\#t''] \in \mathcal{E}$

PROOF: This follows directly from $\langle 2\rangle$9.

$\langle 2\rangle$11. LET: β', t'' and d'' be such that $\mathcal{E} \circledS t'' = t|_k \wedge [\emptyset, \mathsf{loop}^1\langle j\rangle\ d] \xrightarrow{t''}$

$[\beta', d''] \wedge t''[\#t''] \in \mathcal{E}$

PROOF: By $\langle 2\rangle 10$, such β', t'' and d'' must exist.

$\langle 2\rangle 12$. $\bar{t}[j] \in [\![\, \mathsf{loop}\langle j\rangle\ d\,]\!]$

PROOF: This follows from $\langle 2\rangle 8$.

$\langle 2\rangle 13$. $\exists t_j, \beta_j : \mathcal{E}\circledS t_j = \bar{t}[j] \wedge [\emptyset, \mathsf{loop}^1\langle j\rangle\ d] \xrightarrow{t_j} [\beta_j, \mathsf{skip}]$

PROOF: $\langle 2\rangle 12$ and theorem 5.

$\langle 2\rangle 14$. LET: t_j, and β_j be such that $\mathcal{E}\circledS t_j = \bar{t}[j] \wedge [\emptyset, \mathsf{loop}^1\langle j\rangle\ d] \xrightarrow{t_j} [\beta_j, \mathsf{skip}]$

PROOF: By $\langle 2\rangle 13$ such t_j and β_j exist.

$\langle 2\rangle 15$. LET: $l.e = l$

$\langle 2\rangle 16$. $e.l\circledS t|_{k+1} = e.l\circledS(t|_k{}^\frown\langle e\rangle) = e.l\circledS(t|_k)^\frown\langle e\rangle \sqsubseteq e.l\circledS\bar{t}[j]$

PROOF: This follows from $\langle 2\rangle 2$ and $\langle 2\rangle 7$ by $\langle 2\rangle 2$, $e.l \subseteq \mathcal{E}$ and the properties of_$\circledS$_.

$\langle 2\rangle 17$. $e.l\circledS t''{}^\frown\langle e\rangle \sqsubseteq e.l\circledS t_j$

PROOF: This follow from $\langle 2\rangle 11$, $\langle 2\rangle 14$ and $\langle 2\rangle 16$ by $e.l \subseteq \mathcal{E}$ and properties of _$\circledS$_.

$\langle 2\rangle 18$. $\exists s, \beta'', d''' : s \in \mathcal{T}^* \wedge [\beta', d''] \xrightarrow{s^\frown\langle e\rangle} [\beta'', d''']$

PROOF: There are three way this may not be the case. 1) e is blocked by an event on lifeline e. But by $\langle 2\rangle 14$ and $\langle 2\rangle 17$ we know that a execution with e following $l.e\circledS t''$ is possible. 2) e is a receive event and its message is not available in β'. But we have assumed no external communication, t must obey the message invariant and we have assumed all messages to be unique, so this is not possible. 3) e is selected away in the execution of a choice operator. But, by $\langle 2\rangle 8$ and $\langle 2\rangle 14$ we know that all events of $t|_k$ can be executing without choosing away e, and after the execution of $t[k]$ we have full control over the choices by choosing s in the appropriate way.

$\langle 2\rangle 19$. LET: s, β'' and d''' be such that $s \in \mathcal{T}^* \wedge [\beta', d''] \xrightarrow{s^\frown\langle e\rangle} [\beta'', d''']$

PROOF: By $\langle 2\rangle 18$ such s, β'' and d''' exist.

$\langle 2\rangle 20$. LET: $t' = t''{}^\frown s^\frown\langle e\rangle$, $\beta = \beta''$ and $d' = d'''\ \mathsf{seq}\ \mathsf{loop}^{j+1}\langle\infty\rangle\ d$

$\langle 2\rangle 21$. $[\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d] \xrightarrow{t'} [\beta, d]$

PROOF: This follows from $\langle 2\rangle 20$, because from $\langle 2\rangle 9$, $\langle 2\rangle 11$ and $\langle 2\rangle 18$ we have

$[\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d] \xrightarrow{t''} [\beta', d''\ \mathsf{seq}\ \mathsf{loop}^{j+1}\langle\infty\rangle\ d] \xrightarrow{s^\frown\langle e\rangle} [\beta'', d'''\ \mathsf{seq}\ \mathsf{loop}^{j+1}\langle\infty\rangle\ d]$

$\langle 2\rangle 22$. $\mathcal{E}\circledS t' = t|_{k+1}$

PROOF: By $\langle 2\rangle 1$, $\langle 2\rangle 11$, $\langle 2\rangle 19$ and $\langle 2\rangle 20$ we have

$\mathcal{E}\circledS t' = \mathcal{E}\circledS(t''{}^\frown s^\frown\langle e\rangle) = (\mathcal{E}\circledS t'')^\frown(\mathcal{E}\circledS s)^\frown(\mathcal{E}\circledS\langle e\rangle) = t|_k{}^\frown\langle\rangle^\frown\langle e\rangle =$

$t|_k{}^\frown t[k+1] = t|_{k+1}$.

$\langle 2\rangle 23$. Q.E.D.

PROOF: $\langle 2\rangle 21$ and $\langle 2\rangle 22$.

$\langle 1\rangle 3$. Q.E.D.

PROOF: $\langle 1\rangle 1$ and $\langle 1\rangle 2$.

□

Theorem 7 (Completeness) *Let d be a diagram without infinite loops, and $env.d = \emptyset$ (i.e. without external communication). If $t \in [\![\, \mathsf{loop}\langle\infty\rangle\ d\,]\!]$ then there exists trace t' such that $\mathcal{E}\circledS t' = t$ and $[\emptyset, \mathsf{loop}\langle\infty\rangle\ d]$ produces t'.*

Proof of theorem 7

ASSUME: 1. d is a diagram without infinite loops
2. $env.d = \emptyset$
3. $t \in [\![\, \mathsf{loop}\langle\infty\rangle\ d \,]\!]$

PROVE: There exists t' such that $\mathcal{E}Ⓢt' = t$ and $[\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d]$ produces t'

PROOF: by contradiction

There are two cases that must be considered: 1) There exists some point where further execution of t is impossible. 2) t does not satisfy the fairness constraints of the operational semantics.

$\langle 1\rangle 1.$ CASE: $\exists k \in \mathbb{N} : \forall s, \beta, d' : (\mathcal{E}Ⓢs = t|_k \wedge [\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d] \xrightarrow{s} [\beta, d']) \Rightarrow [\beta, d'] \overset{t[k+1]}{\not\to}$

$\langle 2\rangle 1.$ $t|_{k+1} = t|_k \frown \langle t[k+1]\rangle \sqsubseteq t$

PROOF: Properties of $\_|\_$ and $\_[\_]$.

$\langle 2\rangle 2.$ Q.E.D.

PROOF: By $\langle 2\rangle 1$, $t|_{k+1}$ is a finite prefix of t. But by assumptions 1, 2 and 3, and lemma 21 we are able to produce any finite prefix, so $\langle 1\rangle 1$ is impossible.

$\langle 1\rangle 2.$ CASE: $\forall k \in \mathbb{N} : \exists s, \beta, d' : (\mathcal{E}Ⓢs = t|_k \wedge [\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d] \xrightarrow{s} [\beta, d']$
$\wedge\ [\beta, d'] \xrightarrow{t[k+1]}) \wedge \forall t' : (\mathcal{E}Ⓢt' = t \Rightarrow \neg \mathit{wft}(t', \mathsf{loop}^1\langle\infty\rangle\ d))$

$\langle 2\rangle 1.$ $\forall k \in \mathbb{N} : \exists s, \beta, d' : (\mathcal{E}Ⓢs = t|_k \wedge [\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d] \xrightarrow{s} [\beta, d'] \wedge [\beta, d'] \xrightarrow{t[k+1]}$
$)$

PROOF: This follows from the fact that $\langle 1\rangle 1$ is impossible.

$\langle 2\rangle 2.$ $\forall t' : (\mathcal{E}Ⓢt' = t \Rightarrow \neg \mathit{wft}(t', \mathsf{loop}^1\langle\infty\rangle\ d))$

PROOF: $\langle 1\rangle 2$ and $\langle 2\rangle 1$.

$\langle 2\rangle 3.$ ASSUME: t' is such that $\mathcal{E}Ⓢt' = t$

$\langle 2\rangle 4.$ $\neg \mathit{wft}(\mathsf{loop}^1\langle\infty\rangle\ d, t')$

PROOF: $\langle 2\rangle 2$ and $\langle 2\rangle 3$.

$\langle 2\rangle 5.$ $\neg\exists \sigma \in \Xi : \pi_2(\mathit{head}(\sigma)) = \mathsf{loop}^1\langle\infty\rangle\ d \wedge \mathit{tr}(\sigma) = t' \wedge \mathit{wfe}(\sigma)$

PROOF: $\langle 2\rangle 4$ and def. (25) of *wft*.

$\langle 2\rangle 6.$ $\forall \sigma \in \Xi : \pi_2(\mathit{head}(\sigma)) \neq \mathsf{loop}^1\langle\infty\rangle\ d \vee \mathit{tr}(\sigma) \neq t' \vee \neg\mathit{wfe}(\sigma)$

PROOF: $\langle 2\rangle 5$.

$\langle 2\rangle 7.$ ASSUME: σ is an execution such that
$\sigma = [\beta_1, d_1] \xrightarrow{x_1} [\beta_2, d_2] \xrightarrow{x_2} [\beta_3, d_3] \xrightarrow{x_3} \cdots\ \wedge$
$[\beta_1, d_1] = [\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d] \wedge \mathit{tr}(\sigma) = \langle x_1, x_2, x_3, \ldots\rangle = t'$

PROOF: By $\langle 2\rangle 1$ such σ must exist.

$\langle 2\rangle 8.$ $\neg\mathit{wfe}(\sigma)$

PROOF: $\langle 2\rangle 6$ and $\langle 2\rangle 7$.

$\langle 2\rangle 9.$ $\exists d' \in \mathcal{D}, i \in \mathbb{N} : (\forall j \in \mathbb{N} \cup \{0\} : \mathit{enabled}(d', [\beta_{i+j}, d_{i+j}]) \wedge \neg\exists k \in \mathbb{N} \cup \{0\} : \mathit{executed}(d', [\beta_{i+k}, d_{i+k}], x_{i+k}, [\beta_{i+k+1}, d_{i+k+1}]))$

PROOF: $\langle 2\rangle 8$ and def. (23) of *wfe*.

$\langle 2\rangle 10.$ LET: d' be the least complex diagram projection part (as by the weight function w defined in the proof of theorem 3) such that
$\exists i \in \mathbb{N} : (\forall j \in \mathbb{N} \cup \{0\} : \mathit{enabled}(d', [\beta_{i+j}, d_{i+j}]) \wedge \forall k \in \mathbb{N} \cup \{0\} : \neg\mathit{executed}(d', [\beta_{i+k}, d_{i+k}], x_{i+k}, [\beta_{i+k+1}, d_{i+k+1}]))$

PROOF: By $\langle 2\rangle 9$ such d' exists, and we must then be able to find the least complex d'.

$\langle 2\rangle 11.$ LET: i be the smallest number such that
$\forall j \in \mathbb{N} \cup \{0\} : \mathit{enabled}(d', [\beta_{i+j}, d_{i+j}]) \wedge$
$\forall k \in \mathbb{N} \cup \{0\} : \neg\mathit{executed}(d', [\beta_{i+k}, d_{i+k}], x_{i+k}, [\beta_{i+k+1}, d_{i+k+1}])$

PROOF: By ⟨2⟩10 such i exists, and we may then find the smallest.

⟨2⟩12. $enabled(d', [\beta_i, d_i])$

PROOF: ⟨2⟩11.

⟨2⟩13. d' is a single event or d' is a diagram with a high-level operator as its most significant operator

PROOF: Assume not. Then there exists d'' and d''' such that $d' = d''$ seq d''' or $d' = d''$ par d'''. But then by defs. (19), (20) and (21) we must have $enabled(d', [\beta_n, d_n]) \Rightarrow enabled(d'', [\beta_n, d_n]) \vee enabled(d''', [\beta_n, d_n])$ and likewise for *executed*. But d'' and d''' are less complex that d', so then d' is not the least complex diagram projection part that satisfy ⟨2⟩10.

⟨2⟩14. CASE: d' is a single event

⟨3⟩1. LET: $d' = e \in \mathcal{E}$

⟨3⟩2. $e \lhd d_1$

PROOF: ⟨2⟩12 and defs. (20) and (21) of *executed* and *enabled*.

⟨3⟩3. $\exists c \in \mathbb{N}, d'' \in \mathcal{D} : d_i = d'' \text{ seq loop}^c\langle\infty\rangle\ d \wedge e \lhd d''$

PROOF: By ⟨2⟩7, rules (55) and (6), and def. (19) of $\lhd$ this is the only way to obtain ⟨3⟩2.

⟨3⟩4. LET: c be the smallest number and d'' be such that
$d_i = d'' \text{ seq loop}^c\langle\infty\rangle\ d \wedge e \lhd d''$

PROOF: By ⟨3⟩3 such c and d'' exists.

⟨3⟩5. $e \in ev.d^{c-1}$

PROOF: By ⟨3⟩4, c is the smallest number such that $d_i = d'' \text{ seq loop}^c\langle\infty\rangle\ d$ and $e \lhd d''$. If there existed $n < c-1$ such that $e \in ev.d^n$ there would have existed d''' such that $d_i = d''' \text{ seq loop}^{n+1}\langle\infty\rangle\ d$, but then c would not be the smallest number to satisfy ⟨3⟩3.

⟨3⟩6. $[\emptyset, \text{loop}^1\langle\infty\rangle\ d] \xrightarrow{\langle x_1, x_2, \ldots, x_{i-1}\rangle} [\beta_i, d'' \text{ seq loop}^c\langle\infty\rangle\ d]$

PROOF: ⟨2⟩7 and ⟨3⟩4.

⟨3⟩7. $[\emptyset, \text{loop}^1\langle c-1\rangle\ d] \xrightarrow{\langle x_1, x_2, \ldots, x_{i-1}\rangle} [\beta_i, d'']$

PROOF: By ⟨3⟩6 and definition of the loop rules.

⟨3⟩8. There exist s, β such that $[\beta_i, d''] \xrightarrow{s} [\beta, \text{skip}]$

PROOF: Because of the syntactical constraints we can assume this to be the case, i.e. that there is nothing in d'' that prevent the execution to reach skip.

⟨3⟩9. LET: s and β be such that $[\beta_i, d''] \xrightarrow{s} [\beta, \text{skip}]$

PROOF: By ⟨3⟩8, s and β must exist.

⟨3⟩10. $e \in ev.(\mathcal{E} Ⓢ s)$

PROOF: By ⟨3⟩3, $e \lhd d''$, so by the rules of the operational semantics this must be the case in an execution from d'' to skip.

⟨3⟩11. $\mathcal{E} Ⓢ (\langle x_1, x_2, \ldots, x_{i-1}\rangle ^\frown s) \in [\![d]\!]^{\leq c-1}$

PROOF: $\text{loop}^1\langle c-1\rangle\ d$ and be considered as d^1 seq d^2 seq $\cdots$ seq d^{c-1}, so by ⟨3⟩7, ⟨3⟩9, defs. (54), (56) and (57), and theorem 4 this must be the case.

⟨3⟩12. $t \in \mathcal{H} \wedge \exists \bar{t} \in \mathcal{H}^\infty : (\forall j \in \mathbb{N} : (\bar{t}[j] \in [\![d]\!]^{\leq j} \wedge \exists h \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \succsim \{h\}) \wedge \forall l \in \mathcal{L} : e.l Ⓢ t = \sqcup_l \bar{t})$

PROOF: Assumption 3 and lemma 19.

⟨3⟩13. LET: $\bar{t} \in \mathcal{H}^\infty$ be such that
$\forall j \in \mathbb{N} : \bar{t}[j] \in [\![d]\!]^{\leq j} \wedge \exists h \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \succsim \{h\}) \wedge \forall l \in \mathcal{L} : e.l Ⓢ t = \sqcup_l \bar{t}$

PROOF: By ⟨3⟩12 such $\bar{t}$ exists.

⟨3⟩14. $e \in ev.\bar{t}[c-1]$

PROOF: We have established that $\mathcal{E} \circledS (\langle x_1, x_2, \ldots, x_{i-1}\rangle \frown s) \in [\![d]\!]^{\leq c-1}$ with $e \in ev.(\mathcal{E} \circledS s)$ (⟨3⟩10 and ⟨3⟩11), $\bar{t}[c-1] \in [\![d]\!]^{\leq c-1}$ (⟨3⟩13) and $e \in ev.d^{c-1}$ (⟨3⟩5). The only way we may have $e \notin ev.\bar{t}[c-1]$ is that e is inside and alt or xalt in d^{c-1}. But then, by theorem 4, there does not exist any finite execution

$$\sigma' = [\beta'_1, d'_1] \xrightarrow{x'_1} [\beta'_2, d'_2] \xrightarrow{x'_2} \cdots \xrightarrow{x'_{n-1}} [\beta'_n, d'_n]$$

with $\beta'_1 = \emptyset$, $d'_1 = \mathsf{loop}^1\langle c-1\rangle\ d$ and $d'_n = \mathsf{skip}$ such that $\exists j \in [1..n-1] : \mathit{executed}(e, [\beta'_j, d'_j], e, [\beta'_{j+1}, d'_{j+1}])$. I.e. we have that

$$\forall j \in [1..n-1] : \neg \mathit{executed}(e, [\beta_j,' d'_j], e, [\beta'_{j+1}, d'_{j+1}])$$

but then we we also have that

$$\forall j \in [1..n] : \neg \mathit{enabled}(e, [\beta'_j, d'_j])$$

because reaching skip means either that e is executed or is never enabled, and $\mathit{enabled}(e, [\beta'_n, d'_n])$ is impossible since $d'_n = \mathsf{skip}$. Because we have chosen an execution $\sigma = [\beta_1, d_1] \xrightarrow{x_1} [\beta_2, d_2] \xrightarrow{x_2} \cdots$ with $\mathit{enabled}(e, [\beta_i, d_i])$ (⟨2⟩7, ⟨2⟩12 and ⟨3⟩1) this is not an interesting situation, and we may safely assume ⟨3⟩14.

⟨3⟩15. $\exists m \in \mathbb{N} : t[m] = e$

⟨4⟩1. LET: $l.e = l$

⟨4⟩2. $\exists p \in \mathbb{N} : (\sqcup_l \bar{t})[p] = e$

PROOF: ⟨3⟩14, def. of $\sqcup_l \bar{t}$ (because $e.l \circledS \bar{t}[c-1] \sqsubseteq \sqcup_l \bar{t}$) and ⟨4⟩1 (because $l.e = l \Leftrightarrow e \in e.l$)

⟨4⟩3. $\exists p \in \mathbb{N} : (e.l \circledS t)[p] = e$

PROOF: $\sqcup_l \bar{t} = e.l \circledS t$ (by ⟨3⟩13) and ⟨4⟩2.

⟨4⟩4. Q.E.D.

PROOF: By ⟨4⟩3 and properties of $e.\_$, $l.\_$ and $\_\circledS\_$ such m must exist.

⟨3⟩16. $\exists q \in \mathbb{N} : q \geq i \wedge x_q = e$

PROOF: ⟨2⟩3, ⟨2⟩7, ⟨2⟩12 and ⟨3⟩15.

⟨3⟩17. $\mathit{executed}(e, [\beta_q, d_q], e, [\beta_{q+1}, d_{q+1}])$

PROOF: ⟨3⟩16 and the assumption that all events are unique.

⟨3⟩18. Q.E.D.

PROOF: ⟨3⟩17 contradict ⟨2⟩10.

⟨2⟩15. CASE: d' is a diagram with a high-level operator as its most significant operator

⟨3⟩1. $\exists x, \beta^\dagger, d^\dagger : \mathit{executed}(d', [\beta_i, d_i], x, [\beta^\dagger, d^\dagger])$

PROOF: ⟨2⟩12 and def. (21) of *enabled*.

⟨3⟩2. LET: x, $\beta^\dagger$ and $d^\dagger$ be such that $\mathit{executed}(d', [\beta_i, d_i], x, [\beta^\dagger, d^\dagger])$

PROOF: By ⟨3⟩1 such x, $\beta^\dagger$ and $d^\dagger$ must exist.

⟨3⟩3. $x \in \mathcal{T} \wedge \beta^\dagger = \beta_i$

PROOF: By ⟨2⟩15 and rule (6) this must be the case.

⟨3⟩4. LET: $x_i = x$, $\beta_{i+i} = \beta_i = \beta^\dagger$ and $d_{i+1} = d^\dagger$

PROOF: This not affect t because, by ⟨2⟩3 and ⟨2⟩7, $t = \mathcal{E} \circledS tr(\sigma)$ and, by ⟨3⟩3, $x_i = x \in \mathcal{T}$. By ⟨2⟩2 and ⟨2⟩6 we can use any σ and t' that have the property that $tr(\sigma) = t'$ and $\mathcal{E} \circledS t' = t$, and hence, by ⟨3⟩2 and ⟨3⟩3, we can safely do the assumption that $x_i = x$, $\beta_{i+1} = \beta_i$ and $d_{i+1} = d^\dagger$.

⟨3⟩5. $\mathit{executed}(d', [\beta_i, d_i], x_i, [\beta_{i+1}, d_{i+1}])$

PROOF: ⟨3⟩2, ⟨3⟩3 and ⟨3⟩4.

⟨3⟩6. Q.E.D.

PROOF: ⟨3⟩5 contradicts ⟨2⟩10.

⟨2⟩16. Q.E.D.

PROOF: Both ⟨2⟩14 and ⟨2⟩15 leads to contradictions, so ⟨1⟩2 is impossible.

⟨1⟩3. Q.E.D.

PROOF: Both ⟨1⟩1 and ⟨1⟩2 are impossible, so we must have that there exists trace t' such that $\mathcal{E} \circledS t' = t$ and $[\emptyset, \mathsf{loop}^1\langle\infty\rangle\ d]$ produces t'.

□

The above soundness and completeness theorems are concerned with diagrams that have only one infinite loop, and the infinite loop as the most significant operator. In the following we make some reflections on how these results carry over to other more complex cases. There are three main cases that must be considered:

1. Combinations d_1 op d_2 where op is an simple operator and d_1 or d_2 or both characterize infinite behavior.
2. Infinite loop inside other high-level operators.
3. The body d of an infinite loop $\mathsf{loop}\langle\infty\rangle\ d$ itself contains infinite loop(s).

In the first case, we have one simple sub-cases and one more complicated. The simple sub-case is the diagram $(\mathsf{loop}\langle\infty\rangle\ d_1)$ par $(\mathsf{loop}\langle\infty\rangle\ d_2)$, which simply yields the merges of the traces of $\mathsf{loop}\langle\infty\rangle\ d_1$ and $\mathsf{loop}\langle\infty\rangle\ d_2$. The weak fairness ensures that none of the par operands are starved infinitely long. Soundness and completeness considerations are then reduced to considerations of the oracle p in the denotational definition of par with respect to our definition of weak fairness. The complicated sub-case is the case of $(\mathsf{loop}\langle\infty\rangle\ d_1)$ seq $(\mathsf{loop}\langle\infty\rangle\ d_2)$. If we have that $ll.d_2 \subseteq ll.d_1$ this means we have a strict sequencing of $\mathsf{loop}\langle\infty\rangle\ d_1$ and $\mathsf{loop}\langle\infty\rangle\ d_2$ such that right operand is blocked by the left operand, and $(\mathsf{loop}\langle\infty\rangle\ d_1)$ seq $(\mathsf{loop}\langle\infty\rangle\ d_2)$ is equivalent to $\mathsf{loop}\langle\infty\rangle\ d_1$. If, on the other hand $ll.d_1 \cap ll.d_2 = \emptyset$, the case is equivalent to $(\mathsf{loop}\langle\infty\rangle\ d_1)$ par $(\mathsf{loop}\langle\infty\rangle\ d_2)$. The difficult situation is when $ll.d_1 \neq ll.d_2$ and $ll.d_1 \cap ll.d_2 \neq \emptyset$, because d_1 will block some, but not all of the lifelines, in d_2. However, this means that the result of $(\mathsf{loop}\langle\infty\rangle\ d_1)$ seq $(\mathsf{loop}\langle\infty\rangle\ d_2)$ is the behavior of $\mathsf{loop}\langle\infty\rangle\ d_1$ merged with the behavior of $\mathsf{loop}\langle\infty\rangle\ d_2$ not blocked by $\mathsf{loop}\langle\infty\rangle\ d_1$, which probably can be handled easily.

The case of a infinite loop inside the high-level operators refuse, assert, alt and xalt poses no problems as its result is just the execution of a silent event before execution of the loop itself. The diagram $\mathsf{loop}\langle n\rangle$ $(\mathsf{loop}\langle\infty\rangle\ d)$ is equivalent to $\mathsf{loop}\langle\infty\rangle\ d$, because the first iteration of the outer loop will block all subsequent iterations.

Finally, consider a diagram of the form $\mathsf{loop}\langle\infty\rangle$ $(d_1$ seq $(\mathsf{loop}\langle\infty\rangle\ d_2)$ seq $d_3)$. The behavior characterized by this diagram is actually the same behavior as characterized by a diagram d_1 seq $((\mathsf{loop}\langle\infty\rangle\ d_2)$ par $(\mathsf{loop}\langle\infty\rangle$ $(d_3'$ seq $d_1'))$ where d_1' and d_3' are the parts of d_1 and d_3 not blocked by $\mathsf{loop}\langle\infty\rangle\ d_2$.

Because diagrams with nested infinite loops can be reduced to diagrams without nested infinite loops, and because composition of diagrams with infinite loop can be accounted for, there are strong indications that the soundness and completeness results carry over to these more complex situations.