**University of Oslo**
**Department of Informatics**

# Why timed sequence diagrams require three-event semantics

Øystein Haugen,
Knut Eilif Husa,
Ragnhild Kobro Runde, Ketil Stølen

**Research Report 309**
**ISBN 82-7368-261-7**
**ISSN 0806-3036**

**Revised December 2006**

List of main revisions made in December 2006:

- Changed the weak sequencing operator `seq` from binary to n-ary.
- Revised the syntactical well-formedness criteria.
- Corrected and simplified the example in Appendix A.3.
- Added Appendix B on trace completeness, and restructured the old appendix B into Appendices C–E.
- Corrected lemmas 7 and 11, and their proofs (using new lemmas).
- Added new lemmas and simplified some of the existing proofs.

Because of new lemmas, the original numbering found in previous versions of this report is no longer kept. An overview of all old lemmas and their new numbering are given in the following table:

| **Old number** | **New number** | **Result** |
|---|---|---|
| Lemma 1 | Lemma 3 | Commutativity of $\parallel$ on trace sets |
| Lemma 2 | Lemma 7 | Associativity of $\parallel$ on trace sets |
| Lemma 3 | Lemma 11 | Associativity of $\succsim$ on trace sets |
| Lemma 4 | Lemma 12 | Distributivity of $\parallel$ over $\cup$ on trace sets |
| Lemma 5 | Lemma 14 | Distributivity of $\succsim$ over $\cup$ on trace sets |
| Lemma 6 | Lemma 15 | Distributivity of $\succsim$ over $\cup$ on trace sets |
| Lemma 7 | Lemma 16 | Distributivity of $\wr$ over $\cup$ on trace sets |
| Lemma 8 | Lemma 17 | Commutativity of $\parallel$ on interaction obligations |
| Lemma 9 | Lemma 18 | Associativity of $\parallel$ on interaction obligations |
| Lemma 10 | Lemma 19 | Associativity of $\succsim$ on interaction obligations |
| Lemma 11 | Lemma 20 | Commutativity of $\uplus$ on sets of interaction obligations |
| Lemma 12 | Lemma 21 | Associativity of $\uplus$ on sets of interaction obligations |
| Lemma 13 | Lemma 22 | Commutativity of $\parallel$ on sets of interaction obligations |
| Lemma 14 | Lemma 23 | Associativity of $\parallel$ on sets of interaction obligations |
| Lemma 15 | Lemma 24 | Associativity of $\succsim$ on sets of interaction obligations |
| Lemma 16 | Lemma 25 | Reflexivity of $\rightsquigarrow_r$ |
| Lemma 17 | Lemma 30 | Monotonicity of $\rightsquigarrow_r$ with respect to $\succsim$ |
| Lemma 18 | Lemma 33 | Monotonicity of $\rightsquigarrow_g$ with respect to $\succsim$ on sets of interaction obligations |
| Lemma 19 | Lemma 36 | Monotonicity of $\rightsquigarrow_g$ with respect to $\uplus$ on sets of interaction obligations |
| Lemma 20 | Lemma 37 | Monotonicity of $\rightsquigarrow_g$ with respect to $\biguplus$ on sets of interaction obligations |
| Lemma 21 | Lemma 38 | Monotonicity of $\rightsquigarrow_g$ with respect to the inductive definition of $\mu_n$ |

All theorems have the same numbering as before.

# Why timed sequence diagrams require three-event semantics

Øystein Haugen[1], Knut Eilif Husa[1,2], Ragnhild Kobro Runde[1], Ketil Stølen[1,3]

[1] Department of Informatics, University of Oslo
[2] Ericsson
[3] SINTEF ICT, Norway

**Abstract.** STAIRS is an approach to the compositional development of sequence diagrams supporting the specification of mandatory as well as potential behavior. In order to express the necessary distinction between black-box and glass-box refinement, an extension of the semantic framework with three event messages is introduced. A concrete syntax is also proposed. The proposed extension is especially useful when describing time constraints. The resulting approach, referred to as Timed STAIRS, is formally underpinned by denotational trace semantics. A trace is a sequence built from three kinds of events: events for transmission, reception and consumption. We argue that such traces give the necessary expressiveness to capture the standard UML interpretation of sequence diagrams as well as the black-box interpretation found in classical formal methods.

## 1 Introduction to STAIRS

Sequence diagrams have been used informally for several decades. The first standardization of sequence diagrams came in 1992 [ITU93] – often referred to as MSC-92. Later we have seen several dialects and variations. The sequence diagrams of UML 1.4 [OMG00] were comparable to those of MSC-92, while the recent UML 2.0 [OMG04] has upgraded sequence diagrams to conform well to MSC-2000 [ITU99].

Sequence diagrams show how messages are sent between objects or other instances to perform a task. They are used in a number of different situations. They are for example used by an individual designer to get a better grip of a communication scenario or by a group to achieve a common understanding of the situation. Sequence diagrams are also used during the more detailed design phase where the precise inter-process communication must be set up according to formal protocols. When testing is performed, the behavior of the system can be described as sequence diagrams and compared with those of the earlier phases.

Sequence diagrams seem to have the ability to be understood and produced by professionals of computer systems design as well as potential end-users and stakeholders of the (future) systems. Even though sequence diagrams are intuitive – a property which is always exploited – which diagrams to make is not

always so intuitive. It is also the case that intuition is not always the best guide for a precise interpretation of a complicated scenario. Therefore we have brought forth an approach for reaching a sensible and fruitful set of sequence diagrams, supported by formal reasoning. We called this approach STAIRS – Steps To Analyze Interactions with Refinement Semantics [HS03].

STAIRS distinguishes between positive and negative traces and accepts that some traces may be inconclusive meaning that they have not yet or should not be characterized as positive or negative. STAIRS views the process of developing the interactions as a process of learning through describing. From a fuzzy, rough sketch, the aim is to reach a precise and detailed description applicable for formal handling. To come from the rough and fuzzy to the precise and detailed, STAIRS distinguishes between three sub-activities: (1) supplementing, (2) narrowing and (3) detailing.

Supplementing categorizes inconclusive behavior as either positive or negative. The initial requirements concentrate on the most obvious normal situations and the most obvious exceptional ones. Supplementing supports this by allowing less obvious situations to be treated later. Narrowing reduces the allowed behavior to match the problem better. Detailing involves introducing a more detailed description without significantly altering the externally observable behavior.

STAIRS distinguishes between potential alternatives and mandatory or obligatory alternatives. A special composition operator named xalt facilitates the specification of mandatory alternatives.

Figure 1 shows our STAIRS example – an interaction overview diagram description of the making of a dinner at an ethnic restaurant.

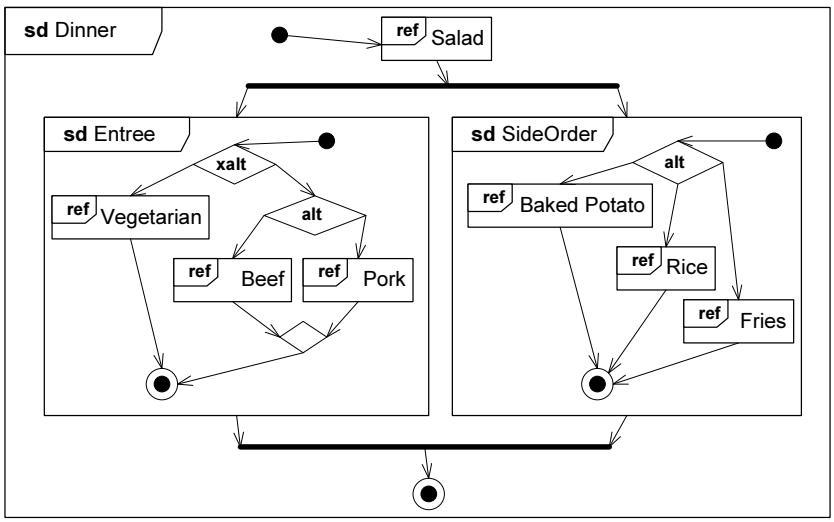


**Fig. 1.** Interaction overview diagram of a dinner

The dinner starts with a salad and continues with a main course that consists of an entree and a side order, which are made in parallel. For the side order there is a simple choice between three alternatives and the restaurant is not obliged to have any particular of them available. Supplementing side orders could be to offer soya beans in addition, while narrowing would mean that the restaurant could choose only to serve rice and never potatoes nor fries. It would still be consistent with the specification and a valid refinement. On the other hand, the entree has more absolute requirements. The restaurant is obliged to offer vegetarian as well as meat, but it does not have to serve both beef and pork. This means that Indian as well as Jewish restaurants are refinements (narrowing) of our dinner concept, while a pure vegetarian restaurant is not valid according to our specification.

The remainder of the paper is divided into six sections: Section 2 motivates the need for a three event semantics for sequence diagrams. Section 3 introduces the formal machinery; in particular, it defines the syntax and semantics of sequence diagrams. Section 4 defines two special interpretations of sequence diagrams, referred to as the standard and the black-box interpretation, respectively. Section 5 demonstrates the full power of Timed STAIRS as specification formalism. Section 6 introduces glass-box and black-box refinement and demonstrates the use of these notions. Section 7 provides a brief conclusion and compares Timed STAIRS to other approaches known from the literature.

## 2 Motivating Timed STAIRS

STAIRS works well for its purpose. However, there are certain aspects that cannot be expressed within the framework as presented in [HS03]. For instance time constraints and the difference between glass-box and black-box view of a system. This section motivates the need for this extra expressiveness.

Let us now look closer at the details of making the Beef in Figure 1. From Figure 2[4] it is intuitive to assume that the working of the Cook making Beef can be explained by the following scheme: The Cook receives an order for main dish (of type Beef) and then turns on the heat and waits until the heat is adequate. Then he fetches the sirloin meat from the refrigerator before putting it on the grill. Then he fetches the sirloin from the stove (hopefully when it is adequately grilled). He then sends the steak to the customer.

[4] This sequence diagram is not a complete specification of Beef. The supplementing has not yet been finished. From a methodological point of view, the diagram should be "closed" with an assert when the supplementing has been finished. This to state that what is still left as inconclusive behavior should from now on be understood as negative. Otherwise, we do not get the semantics intended by Figure 1.

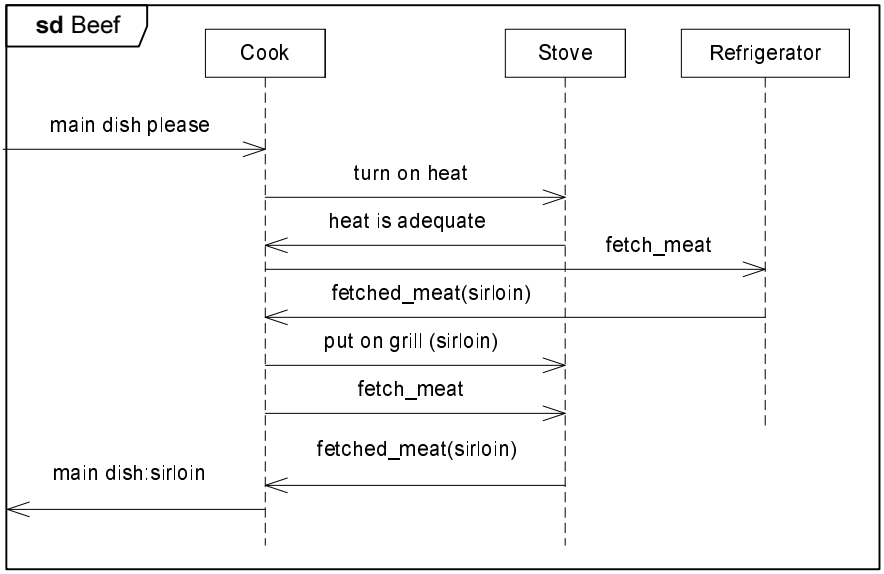


**Fig. 2.** Sequence diagram of Beef

We reached this explanation of the procedures of the cook from looking locally at the cook's lifeline in the Beef diagram. The input event led to one or more outputs, before he again would wait for an input. We found it natural to assume that the input event meant that the cook handled this event, consumed it and acted upon it. This intuition gives rise to what we here will call the standard interpretation of sequence diagrams where an input event is seen as consumption of the event, and where the directly following output events of the trace are causally linked to the consumption. Thus, we can by considering each separate lifeline locally determine the transitions of a state machine describing the lifeline.

Our description of how the beef is made comes from a quiet day, or early in the evening when there were not so many customers and the kitchen had no problems to take care of each order immediately. Furthermore our description was probably made for one of those restaurants where the customers can look into the kitchen through glass. It was a glass-box description. We want, however, to be able to describe the situation later in the evening when the restaurant is crammed with customers and in a restaurant where there is only a black door to the kitchen. We would like to assume that even though the restaurant is full, the kitchen will handle our order immediately, but alas this is of course not the case. We can only observe the kitchen as a black-box. We observe the waiters coming through the door as messengers – orders one way and dishes the other. From these observations we could make estimates of the efficiency of the kitchen. Notice that the efficiency of the kitchen cannot be derived from when the customers placed the orders because the waiters may stop at several tables before they enter the kitchen. Comparing black-box observations of the kitchen

with our glass-box one, we realize that in the glass-box description no event was attached to passing through the door. The order was sent by the customer and consumed by the chef. The passing through the door represents that the kitchen is receiving the message but not necessarily doing something with it. As long as you are not interested in timing matters, the difference is seldom practically significant, but when time matters, the difference between when a message is received and when it is consumed is crucial. How is the kitchen organized to handle the orders in a swift and fair manner?
Motivated by this we will use three events to represent the communication of a message: the sending event, the receiving event and the consumption event, and each of these events may have a timestamp associated. We will introduce concrete syntax for sequence diagrams to capture this and the distinction is also reflected in the semantics. This will give us sufficient expressiveness to describe a black-box interpretation as well as the standard glass-box interpretation.

## 3 Formal foundation

In the following we define the notion of sequence diagram. In particular, we formalize the meaning of sequence diagrams in denotational trace semantics.

### 3.1 Mathematical background on sequences

We will define the semantics of sequence diagrams by using sequences of events. $\mathbb{N}$ denotes the set of natural numbers, while $\mathbb{N}_0$ denotes the set of natural numbers including 0.
By $A^\infty$ and $A^\omega$ we denote the set of all infinite sequences and the set of all finite and infinite sequences over the set $A$, respectively. We define the functions

$$\#\_ \in A^\omega \to \mathbb{N}_0 \cup \{\infty\}, \quad \_[\_] \in A^\omega \times \mathbb{N} \to A \quad \_ \sqsubseteq \_ \in A^\omega \times A^\omega \to \mathbb{B}\mathrm{ool}$$

to yield the length, the $n$th element of a sequence, and the prefix ordering on sequences. Hence, $\#a$ yields the number of elements in $a$, $a[n]$ yields $a$'s $n$th element if $n \leq \#a$, and $a_1 \sqsubseteq a_2$ evaluates to true if $a_1$ is an initial segment of $a_2$, or if $a_1 = a_2$.
We also need functions for concatenation, truncation and filtering:

$$\_ \frown \_ \in A^\omega \times A^\omega \to A^\omega, \quad \_\downarrow\_ \in A^\omega \times \mathbb{N}_0 \to A^\omega,$$
$$\_ \circledS \_ \in \mathbb{P}(A) \times A^\omega \to A^\omega, \quad \_ \circledT \_ \in \mathbb{P}(A \times B) \times (A^\omega \times B^\omega) \to A^\omega \times B^\omega$$

Concatenating two sequences implies gluing them together. Hence, $a_1 \frown a_2$ denotes a sequence of length $\#a_1 + \#a_2$ that equals $a_1$ if $a_1$ is infinite, and is prefixed by $a_1$ and suffixed by $a_2$, otherwise. For any $0 \leq i \leq \#a$, we define $a|_i$ to denote the prefix of $a$ of length $i$.
The filtering function $\circledS$ is used to filter away elements. By $B \circledS a$ we denote the sequence obtained from the sequence $a$ by removing all elements in $a$ that are not in the set of elements $B$. For example, we have that

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

The filtering function $\textcircled{T}$ may be understood as a generalization of $\textcircled{S}$. The function $\textcircled{T}$ filters pairs of sequences with respect to pairs of elements in the same way as $\textcircled{S}$ filters sequences with respect to elements. For any set of pairs of elements $P$ and pair of sequences $t$, by $P \textcircled{T} t$ we denote the pair of sequences obtained from $t$ by

- truncating the longest sequence in $t$ at the length of the shortest sequence in $t$ if the two sequences are of unequal length;
- for each $j \in [1 \ldots k]$, where $k$ is the length of the shortest sequence in $t$, selecting or deleting the two elements at index $j$ in the two sequences, depending on whether the pair of these elements is in the set $P$.

For example, we have that

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

For a formal definition of $\textcircled{T}$, see [BS01].

### 3.2 Syntax of sequence diagrams

A message is a triple $(s, tr, re)$ of a signal $s$, a transmitter $tr$, and a receiver $re$. $\mathcal{M}$ denotes the set of all messages. The transmitters and receivers are lifelines. $\mathcal{L}$ denotes the set of all lifelines.
We distinguish between three kinds of events; a transmission event tagged by an exclamation mark "!", a reception event tagged by a tilde "$\sim$", or a consumption event tagged by a question mark "?". $\mathcal{K}$ denotes $\{!, \sim, ?\}$.
Every event occurring in a sequence diagram is decorated with a unique timestamp. $\mathcal{T}$ denotes the set of timestamp tags. We use logical formulas with timestamp tags as free variables to impose constraints on the timing of events. By $\mathbb{F}(v)$ we denote the set of logical formulas whose free variables are contained in the set of timestamp tags $v$.
$\mathcal{E}$ denotes the set of all events. Formally, an event is a triple of kind, message and timestamp tag

$$\mathcal{E} = \mathcal{K} \times \mathcal{M} \times \mathcal{T}$$

We define the functions

$$k.\_ \in \mathcal{E} \to \mathcal{K}, \quad m.\_ \in \mathcal{E} \to \mathcal{M}, \quad t.\_ \in \mathcal{E} \to \mathcal{T}, \quad tr.\_, re.\_ \in \mathcal{E} \to \mathcal{L}$$

to yield the kind, message, timestamp tag, transmitter and receiver of an event, respectively. We also overload $tr$ and $re$ to yield the transmitter and receiver of a message.
We define the functions

$$tt.\_ \in \mathcal{D} \to \mathbb{P}(\mathcal{T}), \quad ll.\_ \in \mathcal{D} \to \mathbb{P}(\mathcal{L}), \quad ev.\_ \in \mathcal{D} \to \mathbb{P}(\mathcal{E}),$$
$$msg.\_ \in \mathcal{D} \to \mathbb{P}(\mathcal{M})$$

to yield the timestamp tags, lifelines, events and messages of a sequence diagram, respectively.
The set of syntactically correct sequence diagrams $\mathcal{D}$ is defined inductively. $\mathcal{D}$ is the least set such that:

- $\mathcal{E} \subset \mathcal{D}$
- $d \in \mathcal{D} \Rightarrow \mathsf{neg}\ d \in \mathcal{D} \wedge \mathsf{assert}\ d \in \mathcal{D}$
- $d \in \mathcal{D} \wedge I \subseteq \mathbb{N}_0 \cup \{\infty\} \Rightarrow (\mathsf{loop}\ I\ d) \in \mathcal{D}$
- $d_1, d_2 \in \mathcal{D} \Rightarrow d_1\ \mathsf{alt}\ d_2 \in \mathcal{D} \wedge d_1\ \mathsf{xalt}\ d_2 \in \mathcal{D} \wedge d_1\ \mathsf{par}\ d_2 \in \mathcal{D}$
- $d_1, \ldots, d_n \in \mathcal{D} \Rightarrow \mathsf{seq}\ [d_1, \ldots, d_n] \in \mathcal{D}$
- $d \in \mathcal{D} \wedge C \in \mathbb{F}(tt.d) \Rightarrow d\ \mathsf{tc}\ C \in \mathcal{D}$

The base case implies that any event is a sequence diagram. Any other sequence diagram is constructed from the basic ones through the application of operators for negation, assertion, loop, potential choice, mandatory choice, weak sequencing, parallel execution and time constraint. The full set of operators as defined by UML 2.0 [OMG04] is somewhat more comprehensive, and it is beyond the scope of this paper to treat them all. We focus on the operators that we find most essential.
We only consider sequence diagrams that are considered syntacically correct in UML 2.0. Also, we do not handle extra global combined fragments. This means that for all operators except from seq and par we assume that the operand(s) consist only of complete messages, i.e. messages where both the transmission, the reception and the consumption event is within the same operand. Formally, for all operands $d_i$ of an operator different from seq and par, we require:

$$\forall\, m \in msg.d_i \quad : \quad \#\{\!\{\ e \in ev.d_i \mid k.e =! \ \wedge\ m.e = m\ \}\!\} = \#\{\!\{\ e \in ev.d_i \mid k.e =\sim\ \wedge\ m.e = m\ \}\!\} \tag{1}$$

$$\forall\, m \in msg.d_i \quad : \quad \#\{\!\{\ e \in ev.d_i \mid k.e =! \ \wedge\ m.e = m\ \}\!\} = \#\{\!\{\ e \in ev.d_i \mid k.e =? \ \wedge\ m.e = m\ \}\!\} \tag{2}$$

where $\{\!\{\ \}\!\}$ denotes a multi-set and $\#$ is overloaded to yield the number of elements in a multi-set. A multi-set is needed here as the same message (consisting of a signal, a transmitter and a receiver) may occur more than once in the same diagram.
All single-event diagrams are considered syntactically correct. For all diagrams consisting of more than one event, we require that a message is complete if both the transmitter and the receiver lifelines are present in the diagram:

$$\forall\, m \in msg.d : (\#ev.d > 1 \wedge tr.m \in ll.d \wedge re.m \in ll.d) \Rightarrow \#\{\!\{\ e \in ev.d \mid k.e =! \ \wedge\ m.e = m\ \}\!\} = \#\{\!\{\ e \in ev.d \mid k.e =\sim\ \wedge\ m.e = m\ \}\!\} \tag{3}$$

$$\forall\, m \in msg.d : (\#ev.d > 1 \wedge re.m \in ll.d) \Rightarrow \#\{\!\{\ e \in ev.d \mid k.e =\sim\ \wedge\ m.e = m\ \}\!\} = \#\{\!\{\ e \in ev.d \mid k.e =? \ \wedge\ m.e = m\ \}\!\} \tag{4}$$

### 3.3 Representing executions by traces

We are mainly interested in communication scenarios. The actual content of messages is not significant for the purpose of this paper. Hence, we do not give any semantic interpretation of messages as such. The same holds for events except that the timestamp tag is assigned a timestamp in the form of a real number. $\mathbb{R}$ denotes the set of all timestamps. Hence:[5]

$$[\![ \mathcal{E} ]\!] \stackrel{\mathsf{def}}{=} \{(k, m, t \mapsto r) \mid (k, m, t) \in \mathcal{E} \wedge r \in \mathbb{R}\} \tag{5}$$

We define the function

$$r.\_ \in [\![ \mathcal{E} ]\!] \to \mathbb{R}$$

to yield the timestamp of an event.
A trace $h$ is an element of $[\![ \mathcal{E} ]\!]^\omega$ that satisfies a number of well-formedness conditions. We use traces to represent executions. We require the events in $h$ to be ordered by time: the timestamp of the $i$th event is less than or equal to the timestamp of the $j$th event if $i < j$. Formally:

$$\forall\, i, j \in [1..\#h] : i < j \Rightarrow r.h[i] \leq r.h[j] \tag{6}$$

This means that two events may happen at the same time.
The same event takes place only once in the same execution. Hence, we also require:

$$\forall\, i, j \in [1..\#h] : i \neq j \Rightarrow h[i] \neq h[j] \tag{7}$$

We also need to make sure that time will eventually progress beyond any finite point in time. The following constraint states that for each lifeline $l$ represented by infinitely many events in the trace $h$, and for any possible timestamp $t$ there must exist an $l$-event in $h$ whose timestamp is greater than $t$:

$$\forall\, l \in \mathcal{L} : (\#e.l \circledS h = \infty \Rightarrow \forall\, t \in \mathbb{R} : \exists\, i \in \mathbb{N} : r.(e.l \circledS h)[i] > t) \tag{8}$$

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

$$e.l \stackrel{\mathsf{def}}{=} \{e \in [\![ \mathcal{E} ]\!] \mid (k.e =! \wedge tr.e = l) \vee (k.e \in \{\sim, ?\} \wedge re.e = l)\} \tag{9}$$

For any single message, transmission must happen before reception, which must happen before consumption. However, in a particular sequence diagram we may have only the transmitter or the receiver lifeline present. Thus we get the following well-formedness requirements on traces, stating that if at any point in the trace we have a transmission event, up to that point we must have had at least as many transmissions as receptions of that particular message, and similarly

[5] The functions $k, m, t, tr, re$ on $\mathcal{E}$ are lifted to $[\![ \mathcal{E} ]\!]$ in the obvious manner.

for reception events with respect to consumptions:

$$\forall\, i \in [1..\#h] : k.h[i] = ! \Rightarrow \tag{10}$$

$$\#(\{!\} \times \{m.h[i]\} \times U) \circledS\, h|_i > \#(\{\sim\} \times \{m.h[i]\} \times U) \circledS\, h|_i$$

$$\forall\, i \in [1..\#h] : k.h[i] = \sim \Rightarrow \tag{11}$$

$$\#(\{\sim\} \times \{m.h[i]\} \times U) \circledS\, h|_i > \#(\{?\} \times \{m.h[i]\} \times U) \circledS\, h|_i$$

where $U \stackrel{\mathsf{def}}{=} \{t \mapsto r \mid t \in \mathcal{T} \wedge r \in \mathbb{R}\}$.
$\mathcal{H}$ denotes the set of all well-formed traces.
We define three basic composition operators on trace sets, namely parallel execution, weak sequencing, and time constraint denoted by $\parallel$, $\gtrsim$, and $\wr$, respectively. Informally, $s_1 \parallel s_2$ is the set of all traces such that

- all events from one trace in $s_1$ and one trace in $s_2$ are included (and no other events),
- the ordering of events from each of the traces is preserved.

Formally:

$$\begin{aligned} s_1 \parallel s_2 \quad \stackrel{\mathsf{def}}{=} \quad & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2\}^\infty : \\ & \pi_2((\{1\} \times [\![\, \mathcal{E} \,]\!]) \circledT\, (p,h)) \in s_1 \wedge \\ & \pi_2((\{2\} \times [\![\, \mathcal{E} \,]\!]) \circledT\, (p,h)) \in s_2\} \end{aligned} \tag{12}$$

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. $\pi_2$ is a projection operator returning the second element of a pair.
For $s_1 \gtrsim s_2$ we have the constraint that events on one lifeline from one trace in $s_1$ should come before events from one trace in $s_2$ on the same lifeline:

$$\begin{aligned} s_1 \gtrsim s_2 \quad \stackrel{\mathsf{def}}{=} \quad & \{h \in \mathcal{H} \mid \exists\, h_1 \in s_1, h_2 \in s_2 : \forall\, l \in \mathcal{L} : \\ & e.l \circledS\, h = e.l \circledS\, h_1 \frown e.l \circledS\, h_2\} \end{aligned} \tag{13}$$

Time constraint is defined as

$$s \wr C \quad \stackrel{\mathsf{def}}{=} \quad \{h \in s \mid h \models C\} \tag{14}$$

where $h \models C$ holds if for all possible assignments of timestamps to timestamp tags done by $h$, there is an assigment of timestamps to the remaining timestamp tags in $C$ (possibly none) such that $C$ evaluates to true. For example, if

$$h = \langle (k_1, m_1, t_1 \mapsto r_1), (k_2, m_2, t_2 \mapsto r_2), (k_3, m_3, t_2 \mapsto r_3) \rangle, \qquad C = t_1 < t_2$$

then $h \models C$ if $r_1 < r_2$ and $r_1 < r_3$.

### 3.4 Interaction obligations

The semantics of sequence diagrams will eventually be defined as sets of interaction obligations. 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. The term obligation is used to explicitly convey that any implementation of a specification is obliged to fulfill each specified alternative. $\mathcal{O}$ denotes the set of all interaction obligations. Parallel execution, weak sequencing and time constraint are overloaded from sets of traces to interaction obligations as follows:

$$(p_1, n_1) \parallel (p_2, n_2) \quad \stackrel{\mathsf{def}}{=} \quad (p_1 \parallel p_2, (n_1 \parallel (p_2 \cup n_2)) \cup (n_2 \parallel p_1)) \tag{15}$$

$$(p_1, n_1) \gtrsim (p_2, n_2) \quad \stackrel{\mathsf{def}}{=} \quad (p_1 \gtrsim p_2, (n_1 \gtrsim (n_2 \cup p_2)) \cup (p_1 \gtrsim n_2)) \tag{16}$$

$$(p, n) \wr C \quad \stackrel{\mathsf{def}}{=} \quad (p \wr C, n \cup (p \wr \neg C)) \tag{17}$$

An obligation pair $(p, n)$ is contradictory if $p \cap n \neq \varnothing$.
The operators for parallel execution, weak sequencing and time constraint are also overloaded to sets of interaction obligations:

$$O_1 \parallel O_2 \quad \stackrel{\mathsf{def}}{=} \quad \{o_1 \parallel o_2 \mid o_1 \in O_1 \wedge o_2 \in O_2\} \tag{18}$$

$$O_1 \gtrsim O_2 \quad \stackrel{\mathsf{def}}{=} \quad \{o_1 \gtrsim o_2 \mid o_1 \in O_1 \wedge o_2 \in O_2\} \tag{19}$$

$$O \wr C \quad \stackrel{\mathsf{def}}{=} \quad \{o \wr C \mid o \in O\} \tag{20}$$

In order to define the semantics of infinite loops in sequence diagrams, we introduce the notion of "chain". Intuitively, an infinite loop corresponds to infinitely many weak sequencing steps. A chain of interaction obligations is an infinite sequence of obligations such that each element is a sequential composition of the previous obligation in the chain and some other appropriate obligation. For a set $O$ of interaction obligations, its chains is defined as:

$$\begin{aligned} \mathit{chains}(O) \quad \stackrel{\mathsf{def}}{=} \quad & \{\bar{o} \in \mathcal{O}^\infty \mid \\ & \bar{o}[1] \in O \wedge \\ & \forall\, j \in \mathbb{N} : \exists\, o \in O : \bar{o}[j+1] = \bar{o}[j] \gtrsim o\} \end{aligned} \tag{21}$$

From a chain $\bar{o}$ of interaction obligations, we obtain a chain of traces by selecting one positive trace from each obligation in the chain $\bar{o}$, and such that each trace in the chain is an extension (by means of weak sequencing) of the previous trace in the chain. For a chain $\bar{o}$ of interaction obligations, we define its positive chains of traces as:

$$\begin{aligned} \mathit{pos}(\bar{o}) \quad \stackrel{\mathsf{def}}{=} \quad & \{\bar{t} \in \mathcal{H}^\infty \mid \forall\, j \in \mathbb{N} : \\ & \bar{t}[j] \in \pi_1(\bar{o}[j]) \wedge \\ & \exists\, t \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \gtrsim \{t\}\ \} \end{aligned} \tag{22}$$

For a chain $\bar{o}$ of interaction obligations, we get a negative chain of traces by selecting the traces such that the first one is negative in some obligation $\bar{o}[i]$ and all the following traces belong to the negative trace sets of the corresponding obligations. By starting from some obligation $\bar{o}[i]$ and not just from $\bar{o}[1]$, we take into account that a negative trace may have been positive during a finite number of initial iterations. As for $pos(\bar{o})$, each trace in the chain is a weak sequencing extension of the previous trace in the chain. According to definition (16), once we have selected a negative trace, all extensions of this trace will also be negative. Hence, we get the following definition for the negative chains of traces:

$$\begin{aligned} negs(\bar{o}) \quad \stackrel{\mathsf{def}}{=} \quad & \{\bar{t} \in \mathcal{H}^{\infty} \mid \exists\, i \in \mathbb{N} : \forall\, j \in \mathbb{N} : \\ & \qquad \bar{t}[j] \in \pi_2(\bar{o}[j+i-1]) \wedge \\ & \qquad \exists\, t \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \gtrsim \{t\}\ \} \end{aligned} \tag{23}$$

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

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

constitutes a chain whose elements are ordered by $\sqsubseteq$. We use $\sqcup_l \bar{t}$ to denote the least upper bound of this chain of sequences (with respect to $\sqsubseteq$). Since sequences may be infinite such least upper bounds always exist.
For a chain of traces $\bar{t}$, we define its set of approximations $\sqcup\bar{t}$ as:

$$\sqcup\bar{t} \stackrel{\mathsf{def}}{=} \{h \in \mathcal{H} \mid \forall\, l \in \mathcal{L} : e.l \circledS h = \sqcup_l \bar{t}\} \tag{24}$$

For a chain of interaction obligations $\bar{o}$, we then define the obligation $\sqcup\bar{o}$ as:

$$\sqcup\bar{o} \stackrel{\mathsf{def}}{=} (\cup_{\bar{t} \in pos(\bar{o})} \sqcup \bar{t}, \cup_{\bar{t} \in negs(\bar{o})} \sqcup \bar{t}) \tag{25}$$

For a set of interaction obligations, we define a loop construct $\mu_n$, where $n$ denotes the number of times the loop is iterated. $\mu_n\, O$ is defined inductively as follows:

$$\mu_0\, O \quad \stackrel{\mathsf{def}}{=} \quad \{(\{\langle\rangle\}, \varnothing)\} \tag{26}$$

$$\mu_1\, O \quad \stackrel{\mathsf{def}}{=} \quad O \tag{27}$$

$$\mu_n\, O \quad \stackrel{\mathsf{def}}{=} \quad O \gtrsim \mu_{n-1}\, O \quad \text{for } 1 < n < \infty \tag{28}$$

$$\mu_\infty\, O \quad \stackrel{\mathsf{def}}{=} \quad \{\sqcup\bar{o} \mid \bar{o} \in chains(O)\} \tag{29}$$

Finally, to facilitate defining the common alternative choice operator $\mathsf{alt}$, we define an operator for inner union of sets of interaction obligations:

$$O_1 \uplus O_2 \quad \stackrel{\mathsf{def}}{=} \quad \{(p_1 \cup p_2, n_1 \cup n_2) \mid (p_1, n_1) \in O_1 \wedge (p_2, n_2) \in O_2\} \tag{30}$$

$\biguplus$ is the generalization of $\uplus$ to arbitrary many sets:

$$\biguplus_{i \in I} O_i \stackrel{\text{def}}{=} \{\biguplus_{i \in I} o_i \mid o_i \in O_i \text{ for all } i \in I\} \tag{31}$$

where $I$ is an index set and

$$\biguplus_{i \in I} (p_i, n_i) \stackrel{\text{def}}{=} (\bigcup_{i \in I} p_i \ , \bigcup_{i \in I} n_i) \tag{32}$$

### 3.5 Semantics of sequence diagrams

The semantics of sequence diagrams is defined by a function

$$[\![ \ \_ \ ]\!] \in \mathcal{D} \to \mathbb{P}(\mathcal{O})$$

that for any sequence diagram $d$ yields a set $[\![ \ d \ ]\!]$ of interaction obligations.
An event is represented by infinitely many unary positive traces – one for each possible assignment of timestamp to its timestamp tag:

$$[\![ \ (k, m, t) \ ]\!] \stackrel{\text{def}}{=} \{(\{\langle (k, m, t \mapsto r) \rangle \mid r \in \mathbb{R}\}, \varnothing)\} \quad \text{if } (k, m, t) \in \mathcal{E} \tag{33}$$

The neg construct defines negative traces:

$$[\![ \ \mathsf{neg}\ d \ ]\!] \stackrel{\text{def}}{=} \{(\{\langle\rangle\}, p \cup n) \mid (p, n) \in [\![ \ d \ ]\!]\} \tag{34}$$

Notice that a negative trace cannot be made positive by reapplying neg. Negative traces remain negative. Negation is an operation that characterizes traces absolutely and not relatively. The intuition is that the focus of the neg construct is on characterizing the positive traces in the operand as negative. Negative traces will always propagate as negative to the outermost level. The neg construct defines the empty trace as positive. This facilitates the embedding of negs in sequence diagrams also specifying positive behavior.
The assert construct makes all inconclusive traces negative. Otherwise the sets of positive and negative traces are left unchanged:

$$[\![ \ \mathsf{assert}\ d \ ]\!] \stackrel{\text{def}}{=} \{(p, n \cup (\mathcal{H} \setminus p)) \mid (p, n) \in [\![ \ d \ ]\!]\} \tag{35}$$

Note that contradictory obligation pairs remain contradictory.
Constructs of sequence diagrams are graphically either (almost) rectangular or a point (an event). In operands or diagrams, such free constructs are ordered according to their upper left corner. The ordering operator is the weak sequencing operator. In our formal syntax given in Section 3.2 all constructs are explicit. Weak sequencing is defined by the seq construct:

$$[\![ \ \mathsf{seq}\,[d] \ ]\!] \stackrel{\text{def}}{=} [\![ \ d \ ]\!] \tag{36}$$

$$[\![ \ \mathsf{seq}\,[D, d] \ ]\!] \stackrel{\text{def}}{=} [\![ \ \mathsf{seq}\,[D] \ ]\!] \gtrsim [\![ \ d \ ]\!]$$

for $d$ a syntactically correct sequence diagram and $D$ a non-empty list of such sequence diagrams.
The alt construct defines potential traces. The semantics is the union of the trace sets for both positive and negative:

$$[\![\ d_1 \ \mathsf{alt}\ d_2\ ]\!] \quad \stackrel{\mathsf{def}}{=} \quad [\![\ d_1\ ]\!] \uplus [\![\ d_2\ ]\!] \tag{37}$$

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

$$[\![\ d_1 \ \mathsf{xalt}\ d_2\ ]\!] \quad \stackrel{\mathsf{def}}{=} \quad [\![\ d_1\ ]\!] \cup [\![\ d_2\ ]\!] \tag{38}$$

Notice that the sets of negative traces are not combined as is the case with the alt. This is due to the fact that we want to allow behaviors that are positive in one interaction obligation to be negative in another interaction obligation. The intuition behind this is as follows: All positive behaviors in an interaction obligation serve the same overall purpose, e.g. different ways of making beef. Alternative ways of making beef can be introduced by the alt operator. Hence, a behavior cannot be present in both the positive and negative trace sets of an interaction obligation as this would lead to a contradictory specification. However, behaviors specified by different interaction obligations are meant to serve different purposes, e.g. make beef dish and make vegetarian dish. There is nothing wrong about stating that a behavior which is positive in one interaction obligation is negative in another. E.g. steak beef would definitely be positive in a beef context and negative in a vegetarian context. By insisting on separate negative sets of interaction obligations we achieve this wanted property.
The par construct represents a parallel merge. Any trace involving a negative trace will remain negative in the resulting interaction obligation:

$$[\![\ d_1 \ \mathsf{par}\ d_2\ ]\!] \quad \stackrel{\mathsf{def}}{=} \quad [\![\ d_1\ ]\!] \parallel [\![\ d_2\ ]\!] \tag{39}$$

The tc construct defines the effect of a time constraint. The positive traces of the operand that do not fulfill the constraint become negative in the result. The negative traces of the operand remain negative regardless of whether they fulfill the constraint:

$$[\![\ d\ \mathsf{tc}\ C\ ]\!] \quad \stackrel{\mathsf{def}}{=} \quad [\![\ d\ ]\!] \wr C \tag{40}$$

The semantics of the loop construct is the same as the inner union of the semantics for doing the contents of the loop for each possible number in the set $I$, where $I \subseteq \mathbb{N}_0 \cup \{\infty\}$:

$$[\![\ \mathsf{loop}\ I\ d\ ]\!] \quad \stackrel{\mathsf{def}}{=} \quad \biguplus_{i \in I} \mu_i\ [\![\ d\ ]\!] \tag{41}$$

We use inner union $\biguplus$ (as in alt) and not ordinary union (as in xalt) since we do not want to require an implementation to implement the loop for all possible values in the set $I$.

## 4 Two abstractions

An example to illustrate the importance of distinguishing between the message reception and the message consumption event when dealing with timed specifications goes as follows: A restaurant chain specifies in a sequence diagram (see Figure 3) that it should never take more than 10 minutes to prepare a beef dish. The specification is handed over to the local restaurant owner who takes these requirements as an input to the design process of her/his local restaurant. When testing the time it takes to prepare a beef the restaurant finds that it is in accordance with the timing requirements. However, when the restaurant chain inspector comes to verify that the timing policies of the chain are obeyed in the operational restaurant he finds that it takes much longer time than 10 minutes to prepare the beef. Thus, the inspector claims that the restaurant is not working according to the timing requirements while the restaurant owner claims they are working according to the requirements. Who is right? According to UML both are right as there is no notion of buffering of communication in UML. Whether the message arrival of "main dish please" to the kitchen shall be regarded as message reception or consumption is not defined in the semantics of UML, and hence, it is up to the users of the diagrams to interpret the meaning.

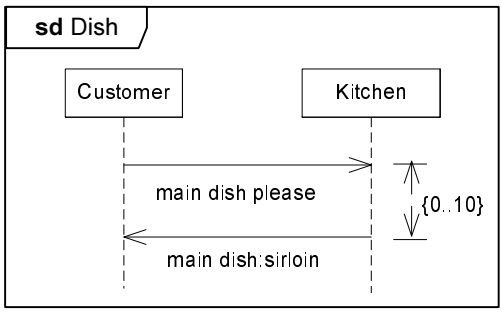


**Fig. 3.** Restaurant specification with time constraint

In this section we define two abstractions over the triple event semantics that match the two different views in the example above, namely the standard interpretation and the black-box interpretation.

### 4.1 Standard interpretation

The standard interpretation is meant to represent the traditional way of interpreting graphical sequence diagrams, namely that the input event of a message at a lifeline represents a consumption. We then only take send (!) and consume (?) events into consideration. Thus, we abstract away the fact that a message will arrive and be stored before it is consumed by the object. The standard interpretation sees graphical sequence diagrams like the diagram in Figure 3 as "standard diagrams".

The syntax and semantics of standard diagrams is defined in exactly the same manner as for general sequence diagrams in Section 3, with the following exceptions.
Standard diagrams do not contain any reception events, meaning that the syntactical well-formedness criteria (1) and (4) are no longer relevant. Criteria (2) is kept for standard diagrams, while criteria (3) is replaced by:

$$\forall\, m \in msg.d : (\#ev.d > 1 \wedge tr.m \in ll.d \wedge re.m \in ll.d) \Rightarrow \tag{42}$$

$$\#\{\!\{\ e \in ev.d \mid k.e =! \ \wedge\ m.e = m\ \}\!\} = \#\{\!\{\ e \in ev.d \mid k.e =? \ \wedge\ m.e = m\ \}\!\}$$

Also, the semantics of events is redefined as follows:

$$\begin{aligned}[\![\ (k, m, t)\ ]\!] \quad \stackrel{\mathsf{def}}{=} \quad & \{(\{h' \frown \langle (k, m, t \mapsto r)\rangle \frown h'' \in \mathcal{H} \mid \\ & \qquad h', h'' \in E(l, \sim)^{\,\omega} \wedge \#h' < \infty \wedge r \in \mathbb{R}\}, \\ & \varnothing)\}\end{aligned} \tag{43}$$

where $l = tr.m$ if $k =!$ and $l = re.m$ if $k =?$, and $E(l, \sim)$ is the set of all events $e \in [\![\ \mathcal{E}\ ]\!]$ such that $re.e = l$ and $k.e = \sim$.
This definition says essentially that in a standard diagram, reception events may happen anywhere on the relevant lifeline (as long as the well-formedness conditions (10) and (11) are obeyed) since they are considered irrelevant in this setting.
If we apply the standard interpretation to the diagram in Figure 3, every positive trace $h$ is such that

$$\begin{aligned}&\{e \in [\![\ \mathcal{E}\ ]\!] \mid k.e \neq \sim\} \circledS\, h = \\ &\quad \langle (!, m, t_1 \mapsto r_1), (?, m, t_3 \mapsto r_3), (!, n, t_4 \mapsto r_4), (?, n, t_6 \mapsto r_6)\rangle\end{aligned}$$

where $r_4 \leq r_3 + 10$, $m$ stands for "main dish please" and $n$ stands for "main dish:sirloin". The implicit reception of $m$ can happen at any point between the corresponding transmission and consumption events, and similarly for $n$ (and any other message).

### 4.2 Black-box interpretation

The black-box interpretation represents the view where the input event of a message at a lifeline represents a reception event. The black-box interpretation sees graphical sequence diagrams like the diagram in Figure 3 as "black-box diagrams".
As with standard diagrams, the syntax and semantics of black-box diagrams is defined in exactly the same manner as for general sequence diagrams in Section 3, with the following exceptions.
Black-box diagrams do not contain any consumption events, meaning that the syntactical well-formedness criteria (2) and (4) are no longer relevant. Criteria (1) and (3) are kept for black-box diagrams.

Also, the semantics of events is redefined as follows:

$$[\![\ (k, m, t)\ ]\!] \quad \stackrel{\mathsf{def}}{=} \quad \{(\{h' \frown \langle (k, m, t \mapsto r)\rangle \frown h'' \in \mathcal{H} \mid h', h'' \in E(l, ?)^{\omega} \wedge \#h' < \infty \wedge r \in \mathbb{R}\}, \varnothing)\} \tag{44}$$

where $l = tr.m$ if $k = !$ and $l = re.m$ if $k = \sim$, and $E(l, ?)$ is the set of all events $e \in [\![\ \mathcal{E}\ ]\!]$ such that $re.e = l$ and $k.e = ?$.
If we apply the black-box interpretation to the diagram in Figure 3, every positive trace $h$ is such that

$$\{e \in [\![\ \mathcal{E}\ ]\!] \mid k.e \neq ?\} \circledS\, h = \langle (!, m, t_1 \mapsto r_1), (\sim, m, t_2 \mapsto r_2), (!, n, t_4 \mapsto r_4), (\sim, n, t_5 \mapsto r_5)\rangle$$

where $r_4 \leq r_2 + 10$. Note that we do not impose any constraint on the implicit consumption events, except that the consumption cannot take place before its reception (if it takes place at all).

## 5 The general case

We have shown that input events are most naturally (standard) interpreted as consumption when they appear on lifelines that represent atomic processes and their concrete implementations should be derived from the lifelines. We have also shown that there are reasons, e.g. timing constraints, that sometimes make it necessary to consider the input event as representing the reception. Moreover, we have seen that timing constraints may also make good sense when applied to consumption events.

In fact we believe that notation for both reception and consumption events are necessary, but that most often for any given message a two-event notation will suffice. Sometimes the message will end in the reception and sometimes in the consumption, but seldom there is a need to make both the reception and the consumption explicit. There are, however, exceptions where all three events must be present to convey the exact meaning of the scenario. Hence, we will in the following introduce graphical notation in order to be able to explicitly state whether a message input event at a lifeline shall be interpreted as a reception event or a consumption event. That is, whether standard or black-box interpretation shall be applied.

Figure 4 shows the graphical notation to specify that a message input event at a lifeline shall be interpreted as a consumption event. Syntactically this notation is equal to the one applied for ordinary two-event sequence diagrams.

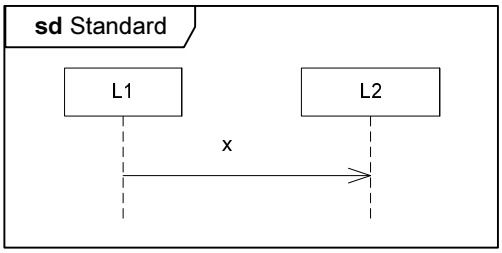


**Fig. 4.** Graphical syntax for specifying standard interpretation

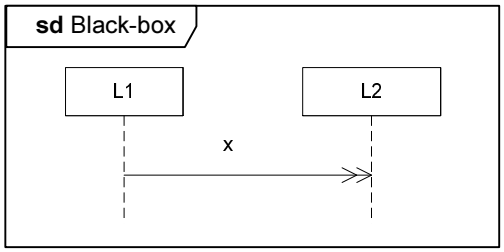


**Fig. 5.** Graphical syntax for specifying black-box interpretation

We express that a message input event at a lifeline shall be interpreted as a reception event and thus be given black-box interpretation by the double arrowhead as shown in Figure 5. We will in the following give some examples of the full approach describing reception events as well as consumption events explicitly.

Let us return to the dinner example where we may assume that the cook is not really one single person, but actually a chef and his apprentice. We may decompose the cook lifeline into new sequence diagrams where the chef and apprentice constitute the internal lifelines. We have shown this in Figure 6 for the preparation of beef shown originally in Figure 2. Let us assume that the apprentice wants to go and get the meat before heating the stove. His priorities may be so because heating the stove is more of a burden, or because the refrigerator is closer at hand. For our purposes we would like to describe a scenario that highlights that the apprentice fetches the meat before heating the stove even though he received the order to turn on the heat first.

In Figure 6 we have shown some explicit reception events, but we have chosen not to show explicitly the corresponding consumptions. This is because our needs were to describe the relationship between the receptions and the actions (outputs) of the apprentice.

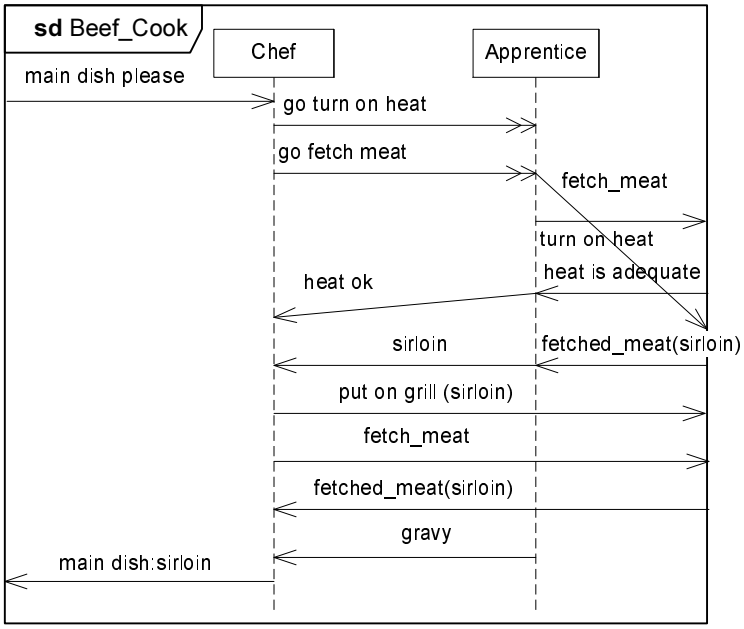


**Fig. 6.** Prioritizing to fetch the meat

The consumptions were considered less important. The disadvantage of this is that we cannot from Figure 6 deduce whether the apprentice actually fetched the meat because he received the order "go fetch meat" or the order "go turn on heat". The reader should appreciate that the "fetch_meat" message crosses the other messages only due to the need to graphically let the formal gates match the events on the decomposed Cook lifeline shown in Figure 2. Semantically there is no ordering between gates. For a formal treatment of gates, see Appendix A.

If we want to give an even more detailed account of the apprentice's options, we may introduce both reception and consumption events. We have done so in Figure 7.

In Figure 7 we see that the chef instructs the apprentice to go turn on heat and to go and fetch meat. The apprentice makes independent decisions for the order of consumption. Here he has decided to consume the order to go fetch meat before consuming go turn on heat. Now we can easily see that the apprentice reacts adequately to the consumptions. It is of course rather risky for the apprentice not to react immediately to the chef's order to turn on the heat, but we may remedy this by timetagging the message receptions of "go turn on heat" and "go fetch meat". Then we specify that the scenario is only valid if these receptions are sufficiently close together in time by a formula including these time tags.

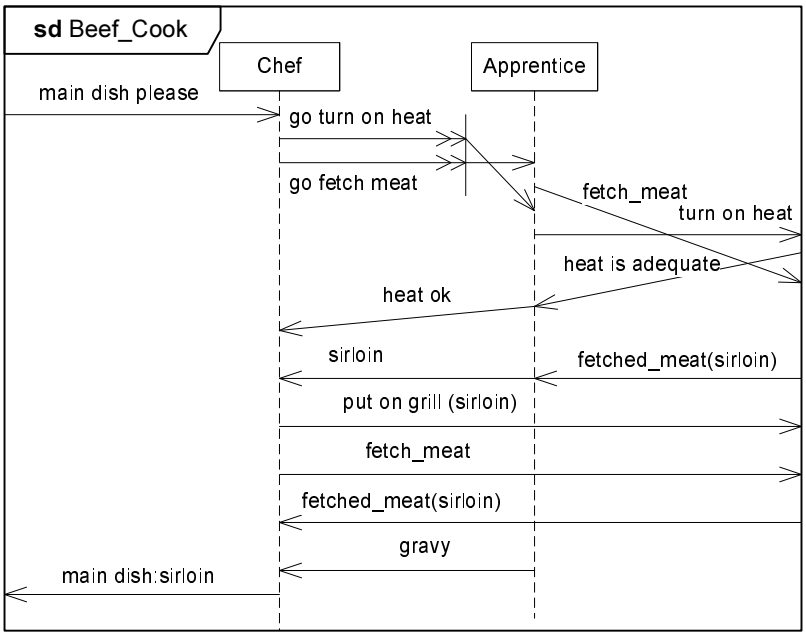


**Fig. 7.** The whole truth

As the examples in this section demonstrate, we have cases where we need to explicitly describe both reception and consumption events in the same diagram, but seldom for the same message. This means that general diagrams may contain standard, black-box as well as three-event arrows. The semantics of such diagrams is given by the definitions in Section 3 with two exceptions:

- The semantics of a consumption event of a standard arrow should be as for consumption events in the standard case (see Section 4.1).
- The semantics of a receive event of a black-box arrow should be as for receive events in the black-box case (see Section 4.2).

## 6 Refinement

Refinement means to add information to a specification such that the specification becomes closer to an implementation. The set of potential traces will be narrowed and situations that we have not yet considered will be supplemented. We define formally two forms of refinement - glass-box refinement which takes the full semantics of the diagram into account, and black-box refinement which only considers changes that are externally visible.
Negative traces must always remain negative in a refinement, while positive traces may remain positive or become negative if the trace has been cancelled out. Inconclusive traces may go anywhere.

### 6.1 Definition of glass-box refinement

An interaction obligation $(p_2, n_2)$ is a refinement of an interaction obligation $(p_1, n_1)$, written $(p_1, n_1) \rightsquigarrow_r (p_2, n_2)$, iff

$$n_1 \subseteq n_2 \quad \wedge \quad p_1 \subseteq p_2 \cup n_2 \tag{45}$$

A set of interaction obligations $O'$ is a glass-box refinement of a set $O$, written $O \rightsquigarrow_g O'$, iff

$$\forall\, o \in O : \exists\, o' \in O' : o \rightsquigarrow_r o' \tag{46}$$

A sequence diagram $d'$ is then a glass-box refinement of a sequence diagram $d$, written $d \rightsquigarrow_g d'$, iff

$$[\![\ d\ ]\!] \rightsquigarrow_g [\![\ d'\ ]\!] \tag{47}$$

The refinement semantics supports the classical notion of compositional refinement providing a firm foundation for compositional analysis, verification and testing. In Appendix D we prove that refinement as defined above is reflexive and transitive. Reflexivity means that a sequence diagram is viewed as a refinement of itself. Transitivity is important, as it means that a sequence diagram resulting from successive refinement steps will still be a valid refinement of the original specification.
Also, in Appendix E we prove that refinement is monotonic with respect to the composition operators presented in Section 3.5, except from assert. For assert, we have monotonicity in the special case of narrowing defined below. Monotonicity is important, as it ensures compositionality in the sense that the different operands of a specification may be refined separately.

### 6.2 Supplementing and narrowing

Supplementing and narrowing are special cases of the general notion of refinement. Supplementing categorizes inconclusive behavior as either positive or negative. An interaction obligation $(p_2, n_2)$ supplements an interaction obligation $(p_1, n_1)$, written $(p_1, n_1) \rightsquigarrow_s (p_2, n_2)$, iff

$$(n_1 \subset n_2 \quad \wedge \quad p_1 \subseteq p_2) \qquad \vee \qquad (n_1 \subseteq n_2 \quad \wedge \quad p_1 \subset p_2) \tag{48}$$

Narrowing reduces the allowed behavior to match the problem better. An interaction obligation $(p_2, n_2)$ narrows an interaction obligation $(p_1, n_1)$, written $(p_1, n_1) \rightsquigarrow_n (p_2, n_2)$, iff

$$p_2 \subset p_1 \quad \wedge \quad n_2 = n_1 \cup (p_1 \setminus p_2) \tag{49}$$

### 6.3 Example of glass-box refinement

We want to refine the Beef_Cook diagram presented in Figure 7. In a glass-box refinement we are interested in the complete traces described by the diagram, and a selection and/or a supplement of these traces.

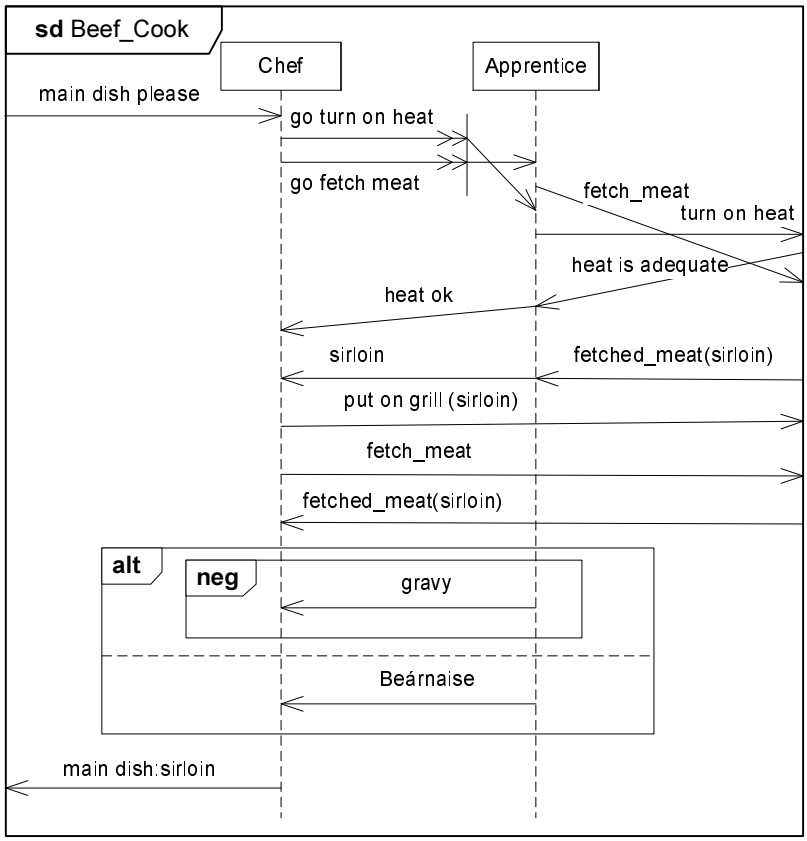


**Fig. 8.** Glass-box refinement of Beef_Cook

Figure 8 is a glass-box refinement of Figure 7. In this diagram we state that we no longer want gravy, but Beárnaise sauce instead. Defining gravy as negative is a narrowing, as it means to reduce the set of positive traces of the original specification. The traces with Beárnaise sauce was earlier considered inconclusive (i.e. neither positive nor negative), but are now defined as positive. This is an example of supplementing. In addition, the diagram in Figure 8 permits using no sauce at all. This is because the neg fragment also introduces the empty trace ($\langle\rangle$) as positive. We summarize these changes in Figure 9.

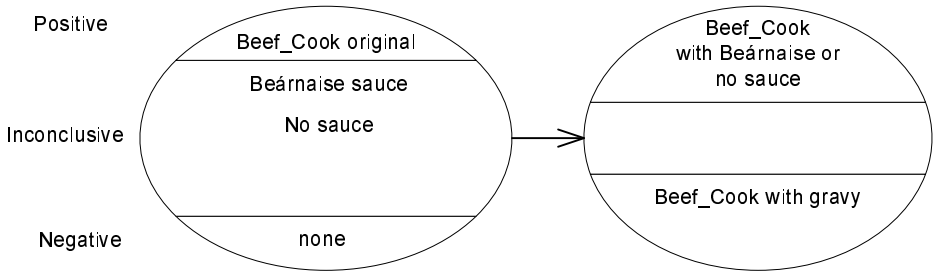


**Fig. 9.** Summary glass-box refinement

### 6.4 Definition of black-box refinement

Black-box refinement may be understood as refinement restricted to the externally visible behavior. We define the function

$$ext \in \mathcal{H} \times \mathbb{P}(\mathcal{L}) \to \mathcal{H}$$

to yield the trace obtained from the trace given as first argument by filtering away those events that are internal with respect to the set of lifelines given as second argument, i.e.:

$$ext(h, l) \stackrel{\text{def}}{=} \{e \in [\![\ \mathcal{E}\ ]\!] \mid (tr.e \notin l \vee re.e \notin l) \wedge k.e \neq ?\} \circledS\, h \tag{50}$$

The *ext* operator is overloaded to sets of traces, to pairs of sets of traces, and sets of pairs of sets of traces in the standard pointwise manner:

$$ext(s, l) \stackrel{\text{def}}{=} \{ext(h, l) \mid h \in s\} \tag{51}$$

$$ext((p, n), l) \stackrel{\text{def}}{=} (ext(p, l), ext(n, l)) \tag{52}$$

$$ext(O, l) \stackrel{\text{def}}{=} \{ext((p, n), l) \mid (p, n) \in O\} \tag{53}$$

A sequence diagram $d'$ is a black-box refinement of a sequence diagram $d$, written $d \rightsquigarrow_b d'$, iff

$$ext([\![\ d\ ]\!], ll.d) \rightsquigarrow_g ext([\![\ d'\ ]\!], ll.d') \tag{54}$$

Notice that the *ext* operator also filters away all consumption events regardless of lifeline, as was the case with black-box interpretation of sequence diagrams. Thus, black-box refinement is mainly relevant in the context of black-box interpretation (even though it may also be applied to standard diagrams).

### 6.5 Example of black-box refinement

It is obvious from the definition of black-box refinement that any glass-box refinement is also a black-box refinement. What would be a black-box refinement

in our Beef_Cook context, but not a glass-box refinement? If we in a refinement of the specification in Figure 7 had just replaced the gravy with Bearnaise, this change would not affect the externally visible behavior of Beef_Cook as it is defined, and would therefore be a legal black-box refinement. However, it would not be a glass-box refinement since the traces involving gravy have been lost (they are now inconclusive), and this violates the definition.

### 6.6 Detailing

When we increase the granularity of sequence diagrams we call this a detailing of the specification. The granularity can be altered in two different ways: either by decomposing the lifelines such that their inner parts and their internal behavior are displayed in the diagram or by changing the data-structure of messages such that they convey more detailed information.

Black-box refinement is sufficiently general to formalize lifeline decompositions that are not externally visible. However, many lifeline decompositions are externally visible. As an example of a lifeline decomposition that is externally visible, consider the decomposition of Beef_Cook in Figure 6. The messages that originally (in Figure 2) had the Cook as sender/receiver, now have the chef or the apprentice as sender/receiver.

To allow for this, we extend the definition of black-box refinement with the notion of a lifeline substitution. The resulting refinement relation is called lifeline decomposition. A lifeline substitution is a partial function of type $\mathcal{L} \to \mathcal{L}$. $\mathcal{LS}$ denotes the set of all such substitutions. We define the function

$$\mathit{subst} \in \mathcal{D} \times \mathcal{LS} \to \mathcal{D}$$

such that $\mathit{subst}(d, ls)$ yields the sequence diagram obtained from $d$ by substituting every lifeline $l$ in $d$ for which $ls$ is defined with the lifeline $ls(l)$.

We then define that a sequence diagram $d'$ is a lifeline decomposition of a sequence diagram $d$ with respect to a lifeline substitution $ls$, written $d \rightsquigarrow_l^{ls} d'$, iff

$$d \rightsquigarrow_b \mathit{subst}(d', ls) \tag{55}$$

Changing the data-structure of messages may be understood as black-box refinement modulo a translation of the externally visible behavior. This translation is specified by a sequence diagram $t$, and we refer to this as an interface refinement.

We define that a sequence diagram $d'$ is an interface refinement of a sequence diagram $d$ with respect to a sequence diagram $t$, written $d \rightsquigarrow_i^t d'$, iff

$$d \rightsquigarrow_b \mathsf{seq}\,[t, d'] \tag{56}$$

Detailing may then be understood as the transitive and reflexive closure of lifeline decomposition and interface refinement.

### 6.7 Refinement through time constraints

Having given examples of refinement in terms of pure event manipulation and trace selection, we go on to present an example where time constraints represent the refinement constructs.

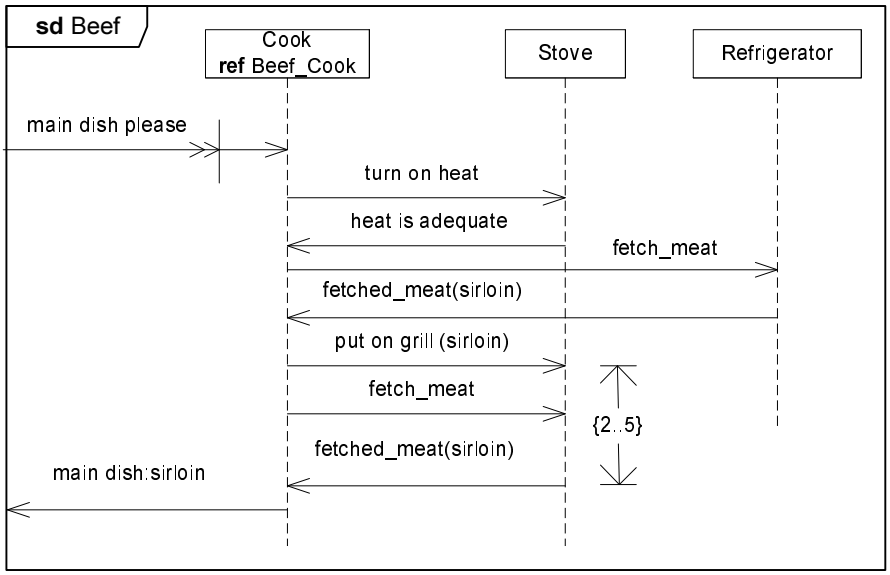


**Fig. 10.** Imposing constraints on timing

We will now introduce two time refinements as indicated in Figures 10 and 11. First we would like to make sure that beefs are neither burned nor raw when fetched from the stove. To make sure that this constraint holds we will put the time constraint on the consumption event of the "put on grill" message. This is because it is the time that the beef is actually present on the stove that matters with respect to how much it is grilled, not the time the beef lies on a plate beside the stove waiting for free space on the stove. All behaviors that do not meet this time constraint are considered negative according to definition (40) of time constraint semantics. Traces that originally were positive are because of the new time constraint now defined as negative. Thus, this step constitutes a glass-box refinement according to definition (47). In fact, it is a narrowing as defined by definition (49). Since the consumption events and transmit events locally define the object behavior, it is only the behavior of the stove that is affected by this refinement step, and not the environment. Using double arrowhead on the "put on grill" message we would not be able to express the intended refinement because it is necessary to talk about message consumption. On the other hand, comparing Figure 10 with the original diagram in Figure 2, we have replaced a standard arrow with a three-event arrow. This is a valid refinement, as it means to make explicit one of the implicit reception events that are already present in

the semantics of Figure 2. This change was made in order to make Figure 10 a valid lifeline decomposition of the Kitchen lifeline in Figure 11, which we will describe next.
We would now like to limit the overall time it takes to prepare a beef. This represents a customer requirement on the kitchen as illustrated in Figure 11. However, the customer does not care about the details of beef preparation, just that it is prepared in time. As seen from Figure 11 this can be interpreted as a time constraint on the reception event. In the same manner as with the glass-box refinement above, the introduction of the time constraint is a narrowing that "moves" traces from the set of positive traces to the set of negative traces. We are not concerned about where the beef spends its time in the kitchen during the preparation process, just that it is prepared in time.

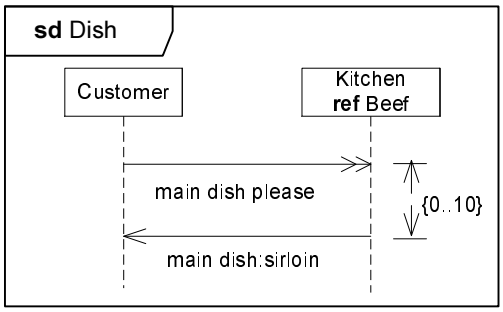


**Fig. 11.** Customer requirement on the beef preparation time

## 7 Conclusions and related work

We have presented Timed STAIRS, a formal approach to the step-wise, incremental development of timed sequence diagrams. It is based on trace semantics. Traces are sequences of events. Events are representations of sending, receiving and consuming messages.
Three event semantics of sequence diagrams has been considered before. In [EMR97] the event ordering imposed by the MSCs is used to determine the physical architecture needed for implementing the specified behavior such as a FIFO buffer between each of the processes. In Timed STAIRS we implicitly assume that every message has one associated input buffer unless something else is explicitly specified in the diagrams. Thus, we do not deduce the communication architecture from the sequence diagrams but instead make it an option for the designer to explicitly specify the wanted architecture in the diagrams. The main rationale for introducing the three event semantics in Timed STAIRS is to be able to distinguish between reception and consumption of messages in order to specify time-constraints on black-box behavior as well as message consumption. Hence, the purpose of the three event semantics is quite different from [EMR97] where time and black-box behavior is not considered.

To consider not only positive traces, but also negative ones, has been suggested before. In [Hau95] the proposed methodology stated that specifying negative scenarios could be even more practical and powerful than only specifying the possible or mandatory ones. It was made clear that the MSC-92 standard [ITU93] was not sufficient to express the intention behind the scenarios and that the MSC documents had to be supplemented with informal statements about the intended interpretation of the set of traces expressed by the different MSCs.

The algebraic semantics of MSC-92 [ITU94] gave rise to a canonical logical expression restricted to the strict sequencing operator and a choice operator. When the MSC standard evolved with more advanced structuring mechanisms, the formal semantics as given in [ITU98] and [Ren98] was based on sets of traces, but it was still expressed in algebraic terms. The MSC approach to sequence diagram semantics is an interleaving semantics based on a fully compositional paradigm. The set of traces denoting the semantics of a message sequence chart can be calculated from its constituent parts based on definitions of the semantics of the structuring concepts as operators. This is very much the approach that we base our semantics on as we calculate our semantics of an interaction fragment from the semantics of its internal fragments. The notion of negative traces, and the explicit distinction between mandatory and potential behavior is beyond the MSC language and its semantics. The Eindhoven school of MSC researchers led by Sjouke Mauw concentrated mainly on establishing the formal properties of the logical systems used for defining the semantics, and also how this could be applied to make tools.

The need for describing also the intention behind the scenarios motivated the so-called "two-layer" approaches. In [CPRO95] they showed how MSC could be combined with languages for temporal logics such as CTL letting the scenarios constitute the atoms for the higher level of modal descriptions. With this one could describe that certain scenarios should appear or should never appear.

Damm and Harel brought this further through their augmented MSC language LSC (Live Sequence Charts) [DH99]. This may also be characterized as a two-layer approach as it takes the basic message sequence charts as starting point and add modal characteristics upon those. The modal expressiveness is strong in LSC since charts, locations, messages and conditions are orthogonally characterized as either mandatory or provisional. Since LSC also includes a notion of subchart, the combinatory complexity can be quite high. The "inline expressions" of MSC-96 (corresponding to combined fragments in UML 2.0) and MSC documents as in MSC-2000 [Hau01] (corresponds to classifier in UML 2.0) are, however, not included in LSC. Mandatory charts are called universal. Their interpretation is that provided their initial condition holds, these charts must happen. Mandatory as in LSC should not be confused with mandatory as in Timed STAIRS, since the latter only specifies traces that must be present in an implementation while the first specifies all allowed traces. Hence, mandatory as in Timed STAIRS does not distinguish between universal or existential interpretation, but rather gives a restriction on what behaviors that must be kept during a refinement. Provisional charts are called existential and they may

happen if their initial condition holds. Through mandatory charts it is of course indirectly also possible to define scenarios that are forbidden or negative. Their semantics is said to be a conservative extension of the original MSC semantics, but their construction of the semantics is based on a two-stage procedure. The first stage defines a symbolic transition system from an LSC and from that a set of executions accepted by the LSC is produced. These executions represent traces where each basic element is a snapshot of a corresponding system.
The motivation behind LSC is explicitly to relate sequence diagrams to other system descriptions, typically defined with state machines. Harel has also been involved in the development of a tool-supported methodology that uses LSC as a way to prescribe systems as well as verifying the correspondence between manually described LSCs and State Machines [HM03].
Our approach is similar to LSC since it is basically interleaving. Timed STAIRS is essentially one-stage as the modal distinction between the positive and negative traces in principle is present in every fragment. The final modality results directly from the semantic compositions. With respect to language, we consider almost only what is UML 2.0, while LSC is a language extension of its own. LSC could in the future become a particular UML profile. Furthermore, our focus is on refinement of sequence diagrams as a means for system development and system validation. This means that in our approach the distinction between mandatory and provisional is captured through the interaction obligations.
The work by Krüger [Krü00] addresses similar concerns as the ones introduced in this article and covered by the LSC-approach of Harel. Just as with LSC MSCs can be given interpretations as existential or universal. The exact and negative interpretations are also introduced. Krüger also proposes notions of refinement for MSCs. Binding references, interface refinement, property refinement and structural refinement are refinement relations between MSCs at different level of abstraction. Narrowing as described in Timed STAIRS corresponds closely to property refinement in [Krü00] and detailing corresponds to interface refinement and structural refinement. However, Krüger does not distinguish between intended non-determinism and non-determinism as a result of underspecification in the refinement relation.
Although this paper presents Timed STAIRS in the setting of UML 2.0 sequence diagrams, the underlying principles apply just as well to MSC given that the MSC language is extended with an `xalt` construct similar to the one proposed above for UML 2.0. Timed STAIRS may also be adapted to support LSC. Timed STAIRS is complementary to software development processes based on use-cases, and classical object-oriented approaches such as the Unified Process [JBR99]. Timed STAIRS provides formal foundation for the basic incremental steps of such processes.

### 7.1 Acknowledgements

The research on which this paper reports has partly been carried out within the context of the IKT-2010 project SARDAS (15295/431) funded by the Research

Council of Norway. We thank Mass Soldal Lund, Atle Refsdal, Ina Schieferdecker and Fredrik Seehusen for helpful feedback.

## References

[BS01] M. Broy and K. Stølen. *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*. Springer, 2001.

[CPRO95] P. Combes, S. Pickin, B. Renard, and F. Olsen. MSCs to express service requirements as properties on an SDL model: Application to service interaction detection. In *7th SDL Forum (SDL'95)*, pages 243–256. North-Holland, 1995.

[DH99] W. Damm and D. Harel. LSCs: Breathing life into message sequence charts. In *Formal Methods for Open Object-Based Distributed Systems (FMOODS'99)*, pages 293–311. Kluwer, 1999.

[EMR97] A. Engels, S. Mauw, and M.A. Reniers. A hierarchy of communication models for message sequence charts. In *Formal Description Tecniques and Protocol Specification, Testing and Verification*, pages 75–90. Chapman & Hall, 1997.

[Hau95] Ø. Haugen. Using MSC-92 effectively. In *7th SDL Forum (SDL'95)*, pages 37–49. North-Holland, 1995.

[Hau01] Ø. Haugen. MSC-2000 interaction diagrams for the new millennium. *Computer Networks*, 35:721–732, 2001.

[HM03] D. Harel and R. Marelly. Specifying and executing behavioral requirements: The play-in/play-out approach. *Software and System Modeling*, 2:82–107, 2003.

[HS03] Ø. Haugen and K. Stølen. STAIRS – Steps to analyze interactions with refinement semantics. In *Sixth International Conference on UML (UML'2003)*, number 2863 in Lecture Notes in Computer Science, pages 388–402. Springer, 2003.

[ITU93] International Telecommunication Union. *Recommendation Z.120 — Message Sequence Chart (MSC)*, 1993.

[ITU94] International Telecommunication Union. *Recommendation Z.120 Annex B: Algebraic Semantics of Message Sequence Charts*, 1994.

[ITU98] International Telecommunication Union. *Recommendation Z.120 Annex B: Formal Semantics of Message Sequence Charts*, 1998.

[ITU99] International Telecommunication Union. *Recommendation Z.120 — Message Sequence Chart (MSC)*, 1999.

[JBR99] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999.

[Krü00] I. Krüger. *Distributed System Design with Message Sequence Charts*. PhD thesis, Technische Universität München, 2000.

[Lam93] L. Lamport. How to write a proof. *American Mathematical Monthly*, 102(7):600–608, August–September 1993.

[OMG00] Object Management Group. *Unified Modeling Language, Version 1.4*, 2000.

[OMG04] Object Management Group. *UML 2.0 Superstructure Specification*, document: ptc/04-10-02 edition, 2004.

[Ren98] M. A. Reniers. *Message Sequence Chart: Syntax and Semantics*. PhD thesis, Eindhoven University of Technology, 1998.

# A Extending STAIRS to handle gates

In the definition of a sequence diagram in Section 3 we do not take the notion of "gates" into consideration. This is just to keep the presentation simple. In the following we extend our approach to also handle gates.

## A.1 Syntax

As before, a message is a triple $(s, tr, re)$ of a signal $s$, a transmitter $tr$, and a receiver $re$, but contrary to earlier we allow gates as "receivers" and "transmitters". Hence, the transmitters and receivers are now lifelines or gates, and not just lifelines as previously. A gate occurring in the position of a receiver is an output gate, and a gate occurring in the position of a transmitter is an input gate. However, the transmitter and the receiver of a message cannot both be gates, since no message should go directly from an input gate to an output gate. $\mathcal{L}$ and $\mathcal{G}$ denote the set of all lifelines and gates, respectively. $\mathcal{L}$ and $\mathcal{G}$ are assumed to be disjoint.
Events are defined in exactly the same way as before, with the exception that we impose two constraints to make sure that events take place at lifelines only, i.e., not at gates. Formally, an event is a triple of kind, message and timestamp tag

$$(k, (s, tr, re), t) \in \mathcal{K} \times \mathcal{M} \times \mathcal{T}$$

such that

- $k = ! \Rightarrow tr \in \mathcal{L}$
- $k = \sim \vee k = ? \Rightarrow re \in \mathcal{L}$

The set of syntactically correct sequence diagrams $\mathcal{D}$ is defined inductively in the same way as before with the following constraints on gates:

- The operands of alt, xalt and par cannot communicate directly via gates. Hence, the input ports of the first operand are disjoint from the output ports of the second operand, and the other way around.
- The operands of seq have disjoint sets of input gates and disjoint sets of output gates.
- If two operands of seq share a gate[6] then the corresponding messages contain the same signal.

In addition we introduce an operator for gate substitution. If $d \in \mathcal{D}$ is a sequence diagram, $gates(d)$ its set of gates, $g \in gates(d)$ and $h$ is a gate that does not occur in $d$ then $d[^g_h] \in \mathcal{D}$ is a sequence diagram with gates:

$$(gates(d) \setminus \{g\}) \cup \{h\}$$

[6] A gate is shared if it is an input gate of one operand and an output gate of another operand. Thus, a shared gate is a way to represent the same as a connector in the graphical syntax of UML 2.0.

### A.2 Semantics

For any sequence diagram $d$, let the function $gm$ be a set of maplets such that the following holds:

- $d \in \mathcal{E} \wedge tr.d, re.d \in \mathcal{L} \Rightarrow gm(d) = \varnothing$
- $d \in \mathcal{E} \wedge tr.d \in \mathcal{G} \Rightarrow gm(d) = \{tr.d \mapsto re.d\}$
- $d \in \mathcal{E} \wedge re.d \in \mathcal{G} \Rightarrow gm(d) = \{re.d \mapsto tr.d\}$
- $d = \mathsf{neg}\ d' \Rightarrow gm(d) = gm(d')$
- $d = \mathsf{loop}\ d' \Rightarrow gm(d) = gm(d')$
- $d = \mathsf{assert}\ d' \Rightarrow gm(d) = gm(d')$
- $d = d'\ \mathsf{tc}\ C \Rightarrow gm(d) = gm(d')$
- $d = d_1\ \mathsf{alt}\ d_2 \Rightarrow gm(d) = gm(d_1) \cup gm(d_2)$
- $d = d_1\ \mathsf{xalt}\ d_2 \Rightarrow gm(d) = gm(d_1) \cup gm(d_2)$
- $d = d_1\ \mathsf{par}\ d_2 \Rightarrow gm(d) = gm(d_1) \cup gm(d_2)$
- $d = \mathsf{seq}\,[d_1, \ldots, d_n] \Rightarrow$
  $gm(d) = \{(g \mapsto l) \in gm(d_i) \mid i \in [1 \ldots n] \wedge$
  $\neg\, \exists\, l', i' : i \neq i' \wedge (g \mapsto l') \in gm(d_{i'})\}$
- $d = d'[^{g}_{h}] \Rightarrow \exists\, l \in L : gm(d) = (gm(d') \setminus \{g \mapsto l\}) \cup \{h \mapsto l\}$

We may think of $gm(d)$ as a function providing a matching lifeline for each gate in its domain. In the case of $\mathsf{seq}$, the definition states that if there is a shared gate (i.e. a gate that exists in two operands), this gate should *not* be taken as a gate of the total diagram (see example below).
For any interaction obligation $o$ and substitution $gm(d)$, we define $o \cdot gm(d)$ to denote the interaction obligation obtained from $o$ by replacing any occurrence of any gate $g$ for which $gm(d)$ is defined in any of $o$'s traces by $(gm(d))(g)$.
The semantics of a sequence diagram can then be defined as before with the exception that

- sequential composition is updated to

$$[\![\ \mathsf{seq}\,[d]\ ]\!] \quad \stackrel{\mathsf{def}}{=} \quad [\![\ d\ ]\!] \tag{57}$$

$$[\![\ \mathsf{seq}\,[D, d]\ ]\!] \quad \stackrel{\mathsf{def}}{=} \quad [\![\ \mathsf{seq}\,[D]\ ]\!] \cdot gm(d) \gtrsim [\![\ d\ ]\!] \cdot gm(\mathsf{seq}\,[D])$$

  with $O \cdot gm(d)$ defined as

$$O \cdot gm(d) \quad \stackrel{\mathsf{def}}{=} \quad \{o \cdot gm(d) \mid o \in O\} \tag{58}$$

- substitution is defined as

$$[\![\ d[^{g}_{h}]\ ]\!] \quad \stackrel{\mathsf{def}}{=} \quad [\![\ d\ ]\!][^{g}_{h}] \tag{59}$$

$$O[^{g}_{h}] \quad \stackrel{\mathsf{def}}{=} \quad \{o \cdot \{g \mapsto h\} \mid o \in O\} \tag{60}$$

### A.3 Example

In the following, we demonstrate how the formal definitions work on a concrete example with gates. The definitions handle three-event, timed sequence diagrams, but in order to simplify the presentation and focus on how gates are handled, in this example we use the two-event, untimed version.

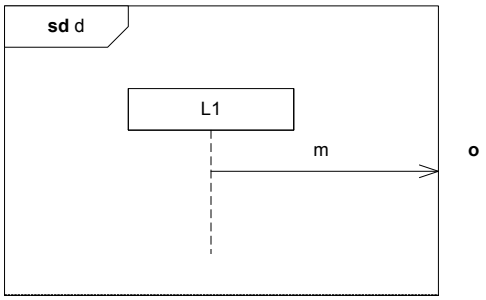


**Fig. 12.** Sequence diagram with one output gate

The sequence diagram $d$ in Figure 12 has one output gate, $o$. Its semantics and gate matching function are defined as:

$$[\![\ d\ ]\!] = \{(\{\langle(!, (m, L_1, o))\rangle\}, \varnothing)\}$$
$$gm(d) = \{o \mapsto L_1\}$$

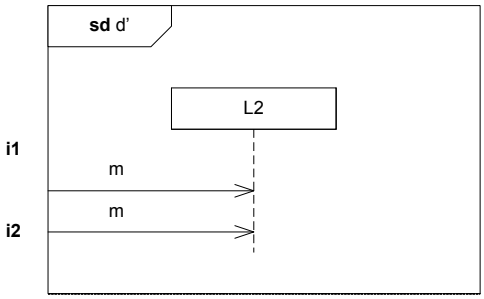


**Fig. 13.** Sequence diagram with two input gates

The sequence diagram $d'$ in Figure 13 has two input gates, $i_1$ and $i_2$. Its semantics and gate matching function are defined as:

$$[\![\ d'\ ]\!] = \{(\{\langle(?, (m, i_1, L_2)), (?, (m, i_2, L_2))\rangle\}, \varnothing)\}$$
$$gm(d') = \{i_1 \mapsto L_2, i_2 \mapsto L_2\}$$

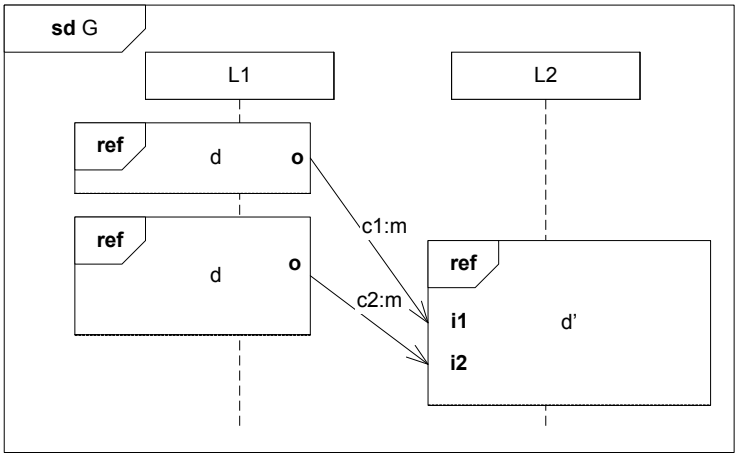


**Fig. 14.** Composed sequence diagram

Two instances of $d$ may be composed with $d'$ as follows:

$$G = \mathsf{seq}\,[\, d[^{o}_{c_1}], d[^{o}_{c_2}], d'[^{i_1}_{c_1}][^{i_2}_{c_2}]\,]$$

This is illustrated in Figure 14.
The semantics of $G$ may be calculated as follows:

$$[\![\ d[^{o}_{c_1}]\ ]\!] = \{(\{\langle(!,(m,L_1,c_1))\rangle\},\varnothing)\}$$
$$gm(d[^{o}_{c_1}]) = \{c_1 \mapsto L_1\}$$

$$[\![\ d[^{o}_{c_2}]\ ]\!] = \{(\{\langle(!,(m,L_1,c_2))\rangle\},\varnothing)\}$$
$$gm(d[^{o}_{c_2}]) = \{c_2 \mapsto L_1\}$$

$$[\![\ \mathsf{seq}\,[\, d[^{o}_{c_1}], d[^{o}_{c_2}]\,]\ ]\!] = \{(\{\langle(!,(m,L_1,c_1)),(!,(m,L_1,c_2))\rangle\},\varnothing)\}$$
$$gm(\mathsf{seq}\,[\, d[^{o}_{c_1}], d[^{o}_{c_2}]\,]) = \{c_1 \mapsto L_1, c_2 \mapsto L_1\}$$

$$[\![\ d'[^{i_1}_{c_1}][^{i_2}_{c_2}]\ ]\!] = \{(\{\langle(?,(m,c_1,L_2)),(?,(m,c_2,L_2))\rangle\},\varnothing)\}$$
$$gm(d'[^{i_1}_{c_1}][^{i_2}_{c_2}]) = \{c_1 \mapsto L_2, c_2 \mapsto L_2\}$$

And finally:

$$\begin{aligned}
[\![\ G\ ]\!] &= \{(\{\langle(!,(m,L_1,L_2)),(!,(m,L_1,L_2))\rangle\},\varnothing)\} \gtrsim \\
&\quad\ \{(\{\langle(?,(m,L_1,L_2)),(?,(m,L_1,L_2))\rangle\},\varnothing)\} \\
&= \{(\{\langle(!,(m,L_1,L_2)),(!,(m,L_1,L_2)),(?,(m,L_1,L_2)),(?,(m,L_1,L_2))\rangle\ , \\
&\qquad \langle(!,(m,L_1,L_2)),(?,(m,L_1,L_2)),(!,(m,L_1,L_2)),(?,(m,L_1,L_2))\rangle\},\varnothing)\}
\end{aligned}$$

The gate matching function of $G$ is the empty set, since both of the actual gates ($c_1$ and $c_2$) have a mapping for both $\mathsf{seq}\,[\, d[^{o}_{c_1}], d[^{o}_{c_2}]\,]$ and $d'[^{i_1}_{c_1}][^{i_2}_{c_2}]$:

$$gm(G) = \varnothing$$

This means that $G$ has no gates, as can be seen from Figure 14.

## B Trace completeness

In section 3.2 we formulated a number of syntactical well-formedness criteria for the sequence diagrams considered in this paper. Criteria (3) and (4) required that for all diagrams consisting of more than one event, a message should be complete if both the transmitter and the receiver lifelines are present in the diagram. In this section we formulate a corresponding semantic notion of trace completeness. We also demonstrate how trace completeness follows for all syntactically correct sequence diagrams, a fact that will be used when proving associativity of seq and par in Section C.
In order to increase the readability of the following definitions and later proofs, we use the notation $\#!m \in h$ to denote the number of send-events with respect to the message $m$ in $h$, and similarly for receive and consume events. Formally:

$$\# * m \in h \stackrel{\mathsf{def}}{=} \#\{\!\{\ h[i] \mid i \in [1..\#h] \wedge k.h[i] = * \wedge m.h[i] = m\ \}\!\} \tag{61}$$

where $*$ is one of !, $\sim$, or ?.
We now introduce the notion of trace completeness. For a trace $h$ to be complete, we require that

1. if $h$ contains both send and receive events of a message $m$, it contains equally many. Formally:

$$(\#!m \in h) > 0 \wedge (\# \sim m \in h) > 0 \Rightarrow (\#!m \in h) = (\# \sim m \in h) \tag{62}$$

2. if $h$ contains more than one event, it contains equally many receive and consume events for all messages $m$. Formally:

$$\#h > 1 \Rightarrow (\# \sim m \in h) = (\#?m \in h) \tag{63}$$

We now define $traces(d)$ to yield all positive and negative traces in the interaction obligations in $[\![\ d\ ]\!]$:

$$traces(d) = \{h \in \mathcal{H} \mid \exists\, s \in tracesets(d) : h \in s\} \tag{64}$$

where $tracesets(d)$ denotes all positive and negative trace-sets in the interaction obligations in $[\![\ d\ ]\!]$, formally defined by:

$$tracesets(d) = \{s \in \mathbb{P}(\mathcal{H}) \mid \exists\, s' \in \mathbb{P}(\mathcal{H}) : (s, s') \in [\![\ d\ ]\!] \vee (s', s) \in [\![\ d\ ]\!]\} \tag{65}$$

The following two observations relate trace completeness to the syntactial well-formedness criteria from Section 3.2.

**Observation 1** Let $d$ be a syntactically correct sequence diagram. From the syntactical well-formedness criteria (1)–(4) and the definition of $[\![\ d\ ]\!]$, it follows that all traces in $traces(d)$ are complete as defined by definitions (62) and (63).

**Observation 2** Let $d_1$ and $d_2$ be syntactically correct sequence diagrams such that also $d_1$ $\mathsf{par}$ $d_2$ and $\mathsf{seq}$ $[d_1, d_2]$ are syntactically correct. From the syntactical well-formedness criteria (1)–(4) and the definition of $[\![\ d\ ]\!]$, it follows that for all traces $h_1 \in traces(d_1)$ and $h_2 \in traces(d_2)$, their combination is complete as defined by definitions (62) and (63), i.e. for all messages $m$:

1. $((\#!m \in h_1) + (\#!m \in h_2)) > 0 \wedge ((\# \sim m \in h_1) + (\# \sim m \in h_2)) > 0$
   $\Rightarrow ((\#!m \in h_1) + (\#!m \in h_2)) = ((\# \sim m \in h_1) + (\# \sim m \in h_2))$
2. $(\#h_1 + \#h_2) > 1$
   $\Rightarrow ((\# \sim m \in h_1) + (\# \sim m \in h_2)) = ((\#?m \in h_1) + (\#?m \in h_2))$

## C Associativity, commutativity and distributivity

In this section we present several lemmas and theorems of associativity, commutativity and distributivity for various operators defined and used in this paper. In particular, we prove that alt, xalt and par are associative and commutative, and that seq is associative.
First, in Section C.1 we prove some helpful lemmas regarding well-formedness of traces. Section C.2 provides the necessary lemmas with respect to trace sets, while Sections C.3 and C.4 contain lemmas on interaction obligations and sets of interaction obligations, respectively. Finally, Section C.5 provides the associativity and commutativity theorems for the sequence diagram operators.
Many of the proofs in these and the following sections are structured in a way similar to that of [Lam93], where the hierarchical structure is used to demonstrate how each proof step is justified.

### C.1 Lemmas on well-formedness

First, we define $h_1 \lhd h_2$ to denote the fact that $h_1$ is a subtrace (possibly non-continuous) of $h_2$. Formally:

$$h_1 \lhd h_2 \stackrel{\mathsf{def}}{=} \exists\, p \in \{1,2\}^\infty : \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p, h_2)) = h_1 \tag{66}$$

**Lemma 1.**

ASSUME: 1. $h_1 \lhd h_2$
2. $h_1$ violates criteria (6), i.e.
$\neg(\forall\, i, j \in [1..\#h_1] : i < j \Rightarrow r.h_1[i] \leq r.h_1[j])$
PROVE: $h_2$ violates criteria (6), i.e.
$\neg(\forall\, i, j \in [1..\#h_2] : i < j \Rightarrow r.h_2[i] \leq r.h_2[j])$

$\langle 1 \rangle 1$. $\exists\, i, j \in [1..\#h_1] : i < j \wedge r.h_1[i] > r.h_1[j]$
PROOF: Assumption 2 and $\neg$-rules.
$\langle 1 \rangle 2$. $\exists\, i', j' \in [1..\#h_2] : i' < j' \wedge r.h_2[i'] > r.h_2[j']$
PROOF: $\langle 1 \rangle 1$ and assumption 1.
$\langle 1 \rangle 3$. $\neg(\forall\, i', j' \in [1..\#h_2] : i' < j' \Rightarrow r.h_2[i'] \leq r.h_2[j'])$
PROOF: $\langle 1 \rangle 2$ and $\neg$-rules.
$\langle 1 \rangle 4$. Q.E.D.

□

**Lemma 2.**

ASSUME: 1. $h_1 \lhd h_2$
2. $h_1$ violates criteria (7), i.e.
$\neg(\forall\, i, j \in [1..\#h_1] : i \neq j \Rightarrow h_1[i] \neq h_1[j])$
PROVE: $h_2$ violates criteria (7), i.e.
$\neg(\forall\, i, j \in [1..\#h_2] : i \neq j \Rightarrow h_2[i] \neq h_2[j])$

⟨1⟩1. $\exists\, i, j \in [1..\#h_1] : i \neq j \wedge h_1[i] = h_1[j]$
PROOF: Assumption 2 and ¬-rules.
⟨1⟩2. $\exists\, i', j' \in [1..\#h_2] : i' \neq j' \wedge h_2[i'] = h_2[j']$
PROOF: ⟨1⟩1 and assumption 1.
⟨1⟩3. $\neg(\forall\, i', j' \in [1..\#h_2] : i' \neq j' \Rightarrow h_2[i'] \neq h_2[j'])$
PROOF: ⟨1⟩2 and ¬-rules.
⟨1⟩4. Q.E.D.

□

### C.2 Lemmas on trace sets

**Lemma 3.** *Commutativity of parallel execution*

PROVE: $s_1 \parallel s_2 = s_2 \parallel s_1$

PROOF:

$$
\begin{array}{lcll}
s_1 \parallel s_2 & = & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2\}^\infty : & \\
 & & \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \wedge & \\
 & & \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_2\} & \text{by definition (12)} \\
 & = & \{h \in \mathcal{H} \mid \exists\, p \in \{2,1\}^\infty : & \\
 & & \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \wedge & \\
 & & \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_2\} & \text{as the symbols in } p \text{ are arbitrary} \\
 & = & \{h \in \mathcal{H} \mid \exists\, p \in \{2,1\}^\infty : & \\
 & & \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_2 \wedge & \\
 & & \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1\} & \text{by the commutativity of } \wedge \\
 & = & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2\}^\infty : & \\
 & & \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_2 \wedge & \\
 & & \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1\} & \text{since } \{1,2\}^\infty = \{2,1\}^\infty \\
 & = & s_2 \parallel s_1 & \text{by definition (12)}
\end{array}
$$

□

**Lemma 4.** *Associativity of parallel execution for non-empty trace-sets*

ASSUME: *1.* $(s_1 \parallel s_2) \neq \varnothing$
*2.* $(s_2 \parallel s_3) \neq \varnothing$

PROVE: $(s_1 \parallel s_2) \parallel s_3 = s_1 \parallel (s_2 \parallel s_3)$

PROOF:
The two sides of the equation reduce to the same formula.
Left side:

$$
\begin{array}{lcll}
(s_1 \parallel s_2) \parallel s_3 & & & \\
 & = & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2\}^\infty : & \\
 & & \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in (s_1 \parallel s_2) \wedge & \\
 & & \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_3\} & \text{by definition (12)} \\
 & = & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2,3\}^\infty : & \\
 & & \quad \pi_2((\{1,2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in (s_1 \parallel s_2) \wedge & \\
 & & \quad \pi_2((\{3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_3\} & \text{as the symbols in } p \text{ are arbitrary}
\end{array}
$$

$$
\begin{aligned}
= \quad & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2,3\}^\infty : \\
& \quad \pi_2((\{1,2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in \\
& \quad\quad \{h' \in \mathcal{H} \mid \exists\, p' \in \{1,2\}^\infty : \\
& \quad\quad\quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p',h')) \in s_1 \wedge \\
& \quad\quad\quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p',h')) \in s_2 \quad \} \wedge \\
& \quad \pi_2((\{3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_3\} && \text{by definition (12)} \\
= \quad & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2,3\}^\infty, p' \in \{1,2\}^\infty : \\
& \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p', \pi_2((\{1,2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)))) \in s_1 \wedge \\
& \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p', \pi_2((\{1,2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)))) \in s_2 \wedge && \text{by definition (12)} \\
& \quad \pi_2((\{3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_3\} && \text{and assumption 1}
\end{aligned}
$$

$$
\begin{aligned}
= \quad & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2,3\}^\infty : \\
& \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \wedge \\
& \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_2 \wedge \\
& \quad \pi_2((\{3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_3\} && \text{constructing } p \text{ with } p' \text{ as a subsequence}
\end{aligned}
$$

Right side:

$s_1 \parallel (s_2 \parallel s_3)$

$$
\begin{aligned}
= \quad & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2\}^\infty : \\
& \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \wedge \\
& \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in (s_2 \parallel s_3)\} && \text{by definition (12)} \\
= \quad & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2,3\}^\infty : \\
& \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \wedge \\
& \quad \pi_2((\{2,3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in (s_2 \parallel s_3)\} && \text{as the symbols in } p \text{ are arbitrary} \\
= \quad & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2,3\}^\infty : \\
& \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \wedge \\
& \quad \pi_2((\{2,3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in \\
& \quad\quad \{h' \in \mathcal{H} \mid \exists\, p' \in \{2,3\}^\infty : \\
& \quad\quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p',h')) \in s_2 \wedge \\
& \quad\quad \pi_2((\{3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p',h')) \in s_3\}\} && \text{by definition (12)}
\end{aligned}
$$

$$
\begin{aligned}
= \quad & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2,3\}^\infty, p' \in \{2,3\}^\infty : \\
& \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \wedge \\
& \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p', \pi_2((\{2,3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)))) \in s_2 \wedge && \text{by definition (12)} \\
& \quad \pi_2((\{3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p', \pi_2((\{2,3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)))) \in s_3\} && \text{and assumption 2}
\end{aligned}
$$

$$
\begin{array}{lll}
= & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2,3\}^\infty : & \\
& \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \wedge & \\
& \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_2 \wedge & \\
& \quad \pi_2((\{3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_3\} & \text{constructing } p \text{ with } p' \text{ as a subsequence}
\end{array}
$$

□

**Lemma 5.** *(To be used when proving associativity of parallel execution in lemma 7)*

Let $d_1$, $d_2$ and $d_3$ be syntactically well-formed sequence diagrams such that also $d_1$ par $d_2$, $d_2$ par $d_3$ and $(d_1 \text{ par } d_2)$ par $d_3$ are syntactically well-formed.

ASSUME: 1. $s_1 \in \mathit{tracesets}(d_1)$
2. $s_2 \in \mathit{tracesets}(d_2)$
3. $s_3 \in \mathit{tracesets}(d_3)$
PROVE: $s_2 \parallel s_3 = \varnothing \Rightarrow (s_1 \parallel s_2) \parallel s_3 = \varnothing$

⟨1⟩1. ASSUME: $s_2 \parallel s_3 = \varnothing$
PROVE: $(s_1 \parallel s_2) \parallel s_3 = \varnothing$
⟨2⟩1. CASE: $s_1 = \varnothing$, $s_2 = \varnothing$, $s_3 = \varnothing$ or $s_1 \parallel s_2 = \varnothing$
PROOF: $(s_1 \parallel s_2) \parallel s_3 = \varnothing$ by definition (12) of $\parallel$.
⟨2⟩2. CASE: $s_1 \neq \varnothing$, $s_2 \neq \varnothing$, $s_3 \neq \varnothing$ and $s_1 \parallel s_2 \neq \varnothing$
⟨3⟩1. $h \notin \mathcal{H}$ for arbitrary
$h \in \{h \in [\![\ \mathcal{E}\ ]\!]^\omega \mid \exists\, p \in \{12,3\}^\infty :$
$\pi_2((\{12\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in (s_1 \parallel s_2) \wedge$
$\pi_2((\{3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_3\}$
⟨4⟩1. Choose $h_{12} \in s_1 \parallel s_2$, $h_3 \in s_3$ and $p \in \{12,3\}^\infty$ such that
$\pi_2((\{12\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) = h_{12}$ and
$\pi_2((\{3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) = h_3$
PROOF: ⟨2⟩2 and ⟨3⟩1.
⟨4⟩2. Choose $h_1 \in s_1$, $h_2 \in s_2$ and $p' \in \{1,2\}^\infty$ such that
$\pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p',h_{12})) = h_1$ and
$\pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p',h_{12})) = h_2$
PROOF: ⟨4⟩1 and definition (12) of $\parallel$.
⟨4⟩3. $\{h \in \mathcal{H} \mid \exists\, p \in \{2,3\}^\infty :$
$\pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_2 \wedge$
$\pi_2((\{3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_3\} = \varnothing$
PROOF: ⟨1⟩1 and definition (12) of $\parallel$.
⟨4⟩4. Choose $h_{23} \lhd h$ such that
$h_{23} \in \{h \in [\![\ \mathcal{E}\ ]\!]^\omega \mid \exists\, p'' \in \{2,3\}^\infty :$
$\pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p'',h_{23})) = h_2 \wedge$
$\pi_2((\{3\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p'',h_{23})) = h_3\}$
⟨5⟩1. $h_2 \lhd h$
⟨6⟩1. $h_2 \lhd h_{12}$
PROOF: ⟨4⟩2 and definition (66) of $\lhd$.

⟨6⟩2. $h_{12} \lhd h$
PROOF: ⟨4⟩1 and definition (66) of $\lhd$.
⟨6⟩3. Q.E.D.
PROOF: ⟨6⟩1, ⟨6⟩2 and definition (66) of $\lhd$.
⟨5⟩2. $h_3 \lhd h$
PROOF: ⟨4⟩1 and definition (66) of $\lhd$.
⟨5⟩3. Q.E.D.
PROOF: ⟨5⟩1, ⟨5⟩2, ⟨4⟩4 and definition (66) of $\lhd$.
⟨4⟩5. $h_{23} \notin \mathcal{H}$
PROOF: ⟨4⟩1, ⟨4⟩2, ⟨4⟩3 and ⟨4⟩4.
⟨4⟩6. $h \notin \mathcal{H}$
⟨5⟩1. CASE: $h_{23}$ violates criteria (6), i.e.
$\neg(\forall\, i, j \in [1..\#h_{23}] : i < j \Rightarrow r.h_{23}[i] \leq r.h_{23}[j])$
PROOF: $h$ violates criteria (6) by ⟨4⟩4 and lemma 1 on well-formedness.
⟨5⟩2. CASE: $h_{23}$ violates criteria (7), i.e.
$\neg(\forall\, i, j, \in [1..\#h_{23}] : i \neq j \Rightarrow h_{23}[i] \neq h_{23}[j])$
PROOF: $h$ violates criteria (7) by ⟨4⟩4 and lemma 2 on well-formedness.
⟨5⟩3. CASE: $h_{23}$ violates criteria (8), i.e.
$\neg(\forall\, l \in \mathcal{L} : \#e.l \circledS h_{23} = \infty$
$\Rightarrow \forall\, t \in \mathbb{R} : \exists\, i \in \mathbb{N} : r.(e.l \circledS h_{23})[i] > t)$
⟨6⟩1. $\forall\, l \in \mathcal{L} : \#e.l \circledS h_2 = \infty$
$\Rightarrow \forall\, t \in \mathbb{R} : \exists\, i \in \mathbb{N} : r.(e.l \circledS h_2)[i] > t$
PROOF: ⟨4⟩2 and assumption 2.
⟨6⟩2. $\forall\, l \in \mathcal{L} : \#e.l \circledS h_3 = \infty$
$\Rightarrow \forall\, t \in \mathbb{R} : \exists\, i \in \mathbb{N} : r.(e.l \circledS h_3)[i] > t$
PROOF: ⟨4⟩1 and assumption 3.
⟨6⟩3. $\forall\, l \in \mathcal{L} : \#e.l \circledS h_{23} = \infty$
$\Rightarrow \forall\, t \in \mathbb{R} : \exists\, i \in \mathbb{N} : r.(e.l \circledS h_{23})[i] > t$
PROOF: ⟨6⟩1, ⟨6⟩2 and ⟨4⟩4.
⟨6⟩4. Q.E.D.
PROOF: Case impossible by ⟨6⟩3.
⟨5⟩4. CASE: $h_{23}$ violates criteria (10), i.e.
$\neg(\forall\, i \in [1..\#h_{23}] : k.h_{23}[i] =! \Rightarrow$
$\#(\{!\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i >$
$\#(\{\sim\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i)$
⟨6⟩1. Choose $i \in [1..\#h_{23}]$ such that $k.h_{23}[i] =!$ and
$\#(\{!\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i \leq$
$\#(\{\sim\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i$
PROOF: ⟨5⟩4 and $\neg$-rules.
⟨6⟩2. LET: $m = m.h_{23}[i]$
PROOF: ⟨6⟩1.
⟨6⟩3. $(\#!m \in h_2) + (\#!m \in h_3) > 0$
PROOF: ⟨4⟩4, ⟨6⟩1 and ⟨6⟩2.
⟨6⟩4. $(\# \sim m \in h_2) + (\# \sim m \in h_3) > 0$
PROOF: ⟨4⟩4, ⟨6⟩1 and ⟨6⟩2.

⟨6⟩5. $(\#!m \in h_2) + (\#!m \in h_3) = (\# \sim m \in h_2) + (\# \sim m \in h_3)$
PROOF: ⟨6⟩3, ⟨6⟩4, ⟨4⟩1, ⟨4⟩2, assumptions 2 and 3 and observation 2, part 2.
⟨6⟩6. $(\#!m \in h) = (\# \sim m \in h)$
PROOF: ⟨6⟩3, ⟨6⟩4, ⟨4⟩1, ⟨4⟩2 and observation 1.
⟨6⟩7. $(\#!m \in h_1) = (\# \sim m \in h_1)$
PROOF: ⟨6⟩5, ⟨6⟩6, ⟨4⟩1 and ⟨4⟩2.
⟨6⟩8. $\exists\, i' \in [1..\#h] : k.h[i'] =! \wedge$
$\#(\{!\} \times \{m.h[i']\} \times U) \circledS h|_{i'} \leq \#(\{\sim\} \times \{m.h[i']\} \times U) \circledS h|_{i'}$
PROOF: ⟨6⟩1, ⟨6⟩7 and ⟨4⟩4.
⟨6⟩9. Q.E.D.
PROOF: $h$ violates criteria (10) by ⟨6⟩8.
⟨5⟩5. CASE: $h_{23}$ violates criteria (11), i.e.
$\neg(\forall\, i \in [1..\#h] : k.h_{23}[i] =\sim \Rightarrow$
$\#(\{\sim\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i >$
$\#(\{?\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i)$
⟨6⟩1. Choose $i \in [1..\#h_{23}]$ such that $k.h_{23}[i] =\sim$ and
$\#(\{\sim\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i \leq$
$\#(\{?\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i$
PROOF: ⟨5⟩5 and ¬-rules.
⟨6⟩2. LET: $m = m.h_{23}[i]$
PROOF: ⟨6⟩1.
⟨6⟩3. $(\# \sim m \in h_2) + (\# \sim m \in h_3) > 0$
PROOF: ⟨4⟩4, ⟨6⟩1 and ⟨6⟩2.
⟨6⟩4. $(\#?m \in h_2) + (\#?m \in h_3) > 0$
PROOF: ⟨4⟩4, ⟨6⟩1 and ⟨6⟩2.
⟨6⟩5. $(\# \sim m \in h_2) + (\# \sim m \in h_3) = (\#?m \in h_2) + (\#?m \in h_3)$
PROOF: ⟨6⟩3, ⟨6⟩4, ⟨4⟩1, ⟨4⟩2, assumptions 2 and 3 and observation 2, part 1.
⟨6⟩6. $(\# \sim m \in h) = (\#?m \in h)$
PROOF: ⟨6⟩3, ⟨6⟩4, ⟨4⟩1, ⟨4⟩2 and observation 1.
⟨6⟩7. $(\# \sim m \in h_1) = (\#?m \in h_1)$
PROOF: ⟨6⟩5, ⟨6⟩6, ⟨4⟩1 and ⟨4⟩2.
⟨6⟩8. $\exists\, i' \in [1..\#h] : k.h[i'] =\sim \wedge$
$\#(\{\sim\} \times \{m.h[i']\} \times U) \circledS h|_{i'} \leq \#(\{?\} \times \{m.h[i']\} \times U) \circledS h|_{i'}$
PROOF: ⟨6⟩1, ⟨6⟩7 and ⟨4⟩4.
⟨6⟩9. Q.E.D.
PROOF: $h$ violates criteria (11) by ⟨6⟩8.
⟨5⟩6. Q.E.D.
PROOF: The cases are exhaustive by ⟨4⟩5 and definition of $\mathcal{H}$.
⟨4⟩7. Q.E.D.
PROOF: ⟨4⟩6.
⟨3⟩2. $\{h \in \mathcal{H} \mid \exists\, p \in \{12, 3\}^\infty :$
$\pi_2((\{12\} \times [\![\, \mathcal{E} \,]\!]) \circledT (p, h)) \in (s_1 \parallel s_2) \wedge$
$\pi_2((\{3\} \times [\![\, \mathcal{E} \,]\!]) \circledT (p, h)) \in s_3\} \quad = \varnothing$

PROOF: ⟨3⟩1.
⟨3⟩3. Q.E.D.
PROOF: $(s_1 \parallel s_2) \parallel s_3 = \varnothing$ by ⟨3⟩2 and definition (12) of $\parallel$.
⟨2⟩3. Q.E.D.
PROOF: The cases are exhaustive.
⟨1⟩2. Q.E.D.
PROOF: ⇒-rule.

□

**Lemma 6.** *(To be used when proving associativity of parallel execution in lemma 7)*

Let $d_1$, $d_2$ and $d_3$ be syntactically well-formed sequence diagrams such that also $d_1$ par $d_2$, $d_2$ par $d_3$ and $d_1$ par $(d_2$ par $d_3)$ are syntactically well-formed.

ASSUME: 1. $s_1 \in \mathit{tracesets}(d_1)$
2. $s_2 \in \mathit{tracesets}(d_2)$
3. $s_3 \in \mathit{tracesets}(d_3)$
PROVE: $s_1 \parallel s_2 = \varnothing \Rightarrow s_1 \parallel (s_2 \parallel s_3) = \varnothing$

⟨1⟩1. ASSUME: $s_1 \parallel s_2 = \varnothing$
PROVE: $s_1 \parallel (s_2 \parallel s_3) = \varnothing$
⟨2⟩1. $s_1 \parallel (s_2 \parallel s_3) = (s_2 \parallel s_3) \parallel s_1$
PROOF: Lemma 3 (commutativity of $\parallel$).
⟨2⟩2. $(s_2 \parallel s_3) \parallel s_1 = (s_3 \parallel s_2) \parallel s_1$
PROOF: Lemma 3 (commutativity of $\parallel$).
⟨2⟩3. $s_2 \parallel s_1 = \varnothing$
PROOF: ⟨1⟩1 and lemma 3 (commutativity of $\parallel$).
⟨2⟩4. $(s_3 \parallel s_2) \parallel s_1 = \varnothing$
PROOF: ⟨2⟩3 and lemma 5.
⟨2⟩5. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩2 and ⟨2⟩4.
⟨1⟩2. Q.E.D.
PROOF: ⇒-rule.

□

**Lemma 7.** *Associativity of parallel execution*

Let $d_1$, $d_2$ and $d_3$ be syntactically well-formed sequence diagrams such that also $d_1$ par $d_2$, $d_2$ par $d_3$, $(d_1$ par $d_2)$ par $d_3$ and $d_1$ par $(d_2$ par $d_3)$ are syntactically well-formed.

ASSUME: 1. $s_1 \in \mathit{tracesets}(d_1)$
2. $s_2 \in \mathit{tracesets}(d_2)$
3. $s_3 \in \mathit{tracesets}(d_3)$
PROVE: $(s_1 \parallel s_2) \parallel s_3 = s_1 \parallel (s_2 \parallel s_3)$

⟨1⟩1. CASE: $s_1 \parallel s_2 = \varnothing$
  ⟨2⟩1. $(s_1 \parallel s_2) \parallel s_3 = \varnothing$
    PROOF: Definition (12) of $\parallel$.
  ⟨2⟩2. $s_1 \parallel (s_2 \parallel s_3) = \varnothing$
    PROOF: ⟨1⟩1 and lemma 6.
  ⟨2⟩3. Q.E.D.
⟨1⟩2. CASE: $s_2 \parallel s_3 = \varnothing$
  ⟨2⟩1. $s_1 \parallel (s_2 \parallel s_3) = \varnothing$
    PROOF: Definition (12) of $\parallel$.
  ⟨2⟩2. $(s_1 \parallel s_2) \parallel s_3 = \varnothing$
    PROOF: ⟨1⟩2 and lemma 5.
  ⟨2⟩3. Q.E.D.
⟨1⟩3. CASE: $s_1 \parallel s_2 \neq \varnothing \wedge s_2 \parallel s_3 \neq \varnothing$
  PROOF: $(s_1 \parallel s_2) \parallel s_3 = s_1 \parallel (s_2 \parallel s_3)$ by lemma 4.
⟨1⟩4. Q.E.D.
  PROOF: The cases are exhaustive.

□

**Lemma 8.** *Associativity of weak sequencing for non-empty trace-sets*

ASSUME: 1. $s_1 \succsim s_2 \neq \varnothing$
2. $s_2 \succsim s_3 \neq \varnothing$

PROVE: $(s_1 \succsim s_2) \succsim s_3 = s_1 \succsim (s_2 \succsim s_3)$

PROOF:
The two sides of the equation reduce to the same formula.
Left side:

$$
\begin{array}{lll}
(s_1 \succsim s_2) \succsim s_3 & & \\
= & \{h \in \mathcal{H} \mid \exists\, h_{12} \in (s_1 \succsim s_2), h_3 \in s_3 : \forall\, l \in \mathcal{L} : & \\
& \quad e.l \circledS h = e.l \circledS h_{12} \frown e.l \circledS h_3\} & \text{by definition (13)} \\
= & \{h \in \mathcal{H} \mid & \\
& \quad \exists\, h_{12} \in \{h \in \mathcal{H} \mid \exists\, h_1 \in s_1, h_2 \in s_2 : \forall\, l' \in \mathcal{L} : & \\
& \qquad e.l' \circledS h = e.l' \circledS h_1 \frown e.l' \circledS h_2\} \wedge & \\
& \quad \exists\, h_3 \in s_3 : \forall\, l \in \mathcal{L} : & \\
& \qquad e.l \circledS h = e.l \circledS h_{12} \frown e.l \circledS h_3\} & \text{by definition (13)} \\
= & \{h \in \mathcal{H} \mid & \\
& \quad \exists\, h_{12} \in \mathcal{H}, h_1 \in s_1, h_2 \in s_2 : & \\
& \qquad (\forall\, l' \in \mathcal{L} : e.l' \circledS h_{12} = e.l' \circledS h_1 \frown e.l' \circledS h_2) \ \wedge & \\
& \qquad \exists\, h_3 \in s_3 : \forall\, l \in \mathcal{L} : & \text{by definition (13)} \\
& \qquad\quad e.l \circledS h = e.l \circledS h_{12} \frown e.l \circledS h_3\} & \text{and assumption 1} \\
= & \{h \in \mathcal{H} \mid \exists\, h_1 \in s_1, h_2 \in s_2, h_3 \in s_3 : \forall\, l \in \mathcal{L} : & \text{using } e.l \circledS h_{12} = e.l \circledS h_1 \frown e.l \circledS h_2 \\
& \quad e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2 \frown e.l \circledS h_3\} & \text{for all } l \in \mathcal{L}
\end{array}
$$

Right side:

$s_1 \succsim (s_2 \succsim s_3)$

$= \{h \in \mathcal{H} \mid \exists\, h_1 \in s_1, h_{23} \in (s_2 \succsim s_3) : \forall\, l \in \mathcal{L} :$
$\quad e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_{23}\}$ by definition (13)

$= \{h \in \mathcal{H} \mid \exists\, h_1 \in s_1,$
$\quad h_{23} \in \{h \in \mathcal{H} \mid \exists\, h_2 \in s_2, h_3 \in s_3 : \forall\, l' \in \mathcal{L} :$
$\quad\quad e.l' \circledS \mathcal{H} = e.l' \circledS h_2 \frown e.l' \circledS h_3\} :$
$\quad \forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_{23}\}$ by definition (13)

$= \{h \in \mathcal{H} \mid$
$\quad \exists\, h_1 \in s_1, h_{23} \in \mathcal{H}, h_2 \in s_2, h_3 \in s_3 :$
$\quad\quad (\forall\, l' \in \mathcal{L} : e.l' \circledS h_{23} = e.l' \circledS h_2 \frown e.l' \circledS h_3) \ \wedge$ by definition (13)
$\quad\quad \forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_{23}\}$ and assumption 2

$= \{h \in \mathcal{H} \mid \exists\, h_1 \in s_1, h_2 \in s_2, h_3 \in s_3 : \forall\, l \in \mathcal{L} :$ using $e.l \circledS h_{23} = e.l \circledS h_2 \frown e.l \circledS h_3$
$\quad e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2 \frown e.l \circledS h_3\}$ for all $l \in \mathcal{L}$

□

**Lemma 9.** *(To be used when proving associativity of weak sequencing in lemma 11)*

Let $d_1$, $d_2$ and $d_3$ be syntactically well-formed sequence diagrams such that also $\mathsf{seq}\ [d_1, d_2]$, $\mathsf{seq}\ [d_2, d_3]$ and $\mathsf{seq}\ [\mathsf{seq}\ [d_1, d_2], d_3]$ are syntactically well-formed.

ASSUME: 1. $s_1 \in \mathit{tracesets}(d_1)$
2. $s_2 \in \mathit{tracesets}(d_2)$
3. $s_3 \in \mathit{tracesets}(d_3)$

PROVE: $s_2 \succsim s_3 = \varnothing \Rightarrow (s_1 \succsim s_2) \succsim s_3 = \varnothing$

⟨1⟩1. ASSUME: $s_2 \succsim s_3 = \varnothing$
PROVE: $(s_1 \succsim s_2) \succsim s_3 = \varnothing$
⟨2⟩1. CASE: $s_1 = \varnothing$, $s_2 = \varnothing$, $s_3 = \varnothing$ or $s_1 \succsim s_2 = \varnothing$
PROOF: $(s_1 \succsim s_2) \succsim s_3 = \varnothing$ by definition (13) of $\succsim$.
⟨2⟩2. CASE: $s_1 \neq \varnothing$, $s_2 \neq \varnothing$, $s_3 \neq \varnothing$ and $s_1 \succsim s_2 \neq \varnothing$
⟨3⟩1. $h \notin \mathcal{H}$ for arbitrary
$h \in \{h \in [\![\ \mathcal{E}\ ]\!]^\omega \mid \exists\, h_{12} \in s_1 \succsim s_2, h_3 \in s_3 : \forall\, l \in \mathcal{L} :$
$e.l \circledS h = e.l \circledS h_{12} \frown e.l \circledS h_3\}$
⟨4⟩1. Choose $h_{12} \in s_1 \succsim s_2$ and $h_3 \in s_3$ such that
$\forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_{12} \frown e.l \circledS h_3$
PROOF: ⟨2⟩2 and ⟨3⟩1.
⟨4⟩2. Choose $h_1 \in s_1$ and $h_2 \in s_2$ such that
$\forall\, l \in \mathcal{L} : e.l \circledS h_{12} = e.l \circledS h_1 \frown e.l \circledS h_2$
PROOF: ⟨4⟩1 and definition (13) of $\succsim$.

⟨4⟩3. $\{h \in \mathcal{H} \mid \exists\, h_2 \in s_2, h_3 \in s_3 : \forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_2 \frown e.l \circledS h_3\}$
$= \varnothing$
PROOF: ⟨1⟩1 and definition (13) of $\succsim$.
⟨4⟩4. $\forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2 \frown e.l \circledS h_3$
PROOF: ⟨4⟩1 and ⟨4⟩2.
⟨4⟩5. Choose $h_{23} \lhd h$ such that $\forall\, l \in \mathcal{L} : e.l \circledS h_{23} = e.l \circledS h_2 \frown e.l \circledS h_3$
PROOF: ⟨4⟩4, definition (66) of $\lhd$ and definitions of $\circledS$ and $\frown$.
⟨4⟩6. $h_{23} \notin \mathcal{H}$
PROOF: ⟨4⟩1, ⟨4⟩2, ⟨4⟩3 and ⟨4⟩5.
⟨4⟩7. $h \notin \mathcal{H}$
⟨5⟩1. CASE: $h_{23}$ violates criteria (6), i.e.
$\neg(\forall\, i, j \in [1..\#h_{23}] : i < j \Rightarrow r.h_{23}[i] \leq r.h_{23}[j])$
PROOF: $h$ violates criteria (6) by ⟨4⟩5 and lemma 1 on well-formedness.
⟨5⟩2. CASE: $h_{23}$ violates criteria (7), i.e.
$\neg(\forall\, i, j, \in [1..\#h_{23}] : i \neq j \Rightarrow h_{23}[i] \neq h_{23}[j])$
PROOF: $h$ violates critiera (7) by ⟨4⟩5 and lemma 2 on well-formedness.
⟨5⟩3. CASE: $h_{23}$ violates criteria (8), i.e.
$\neg(\forall\, l \in \mathcal{L} : \#e.l \circledS h_{23} = \infty$
$\Rightarrow \forall\, t \in \mathbb{R} : \exists\, i \in \mathbb{N} : r.(e.l \circledS h_{23})[i] > t)$
⟨6⟩1. $\forall\, l \in \mathcal{L} : \#e.l \circledS h_2 = \infty$
$\Rightarrow \forall\, t \in \mathbb{R} : \exists\, i \in \mathbb{N} : r.(e.l \circledS h_2)[i] > t$
PROOF: ⟨4⟩2 and assumption 2.
⟨6⟩2. $\forall\, l \in \mathcal{L} : \#e.l \circledS h_3 = \infty$
$\Rightarrow \forall\, t \in \mathbb{R} : \exists\, i \in \mathbb{N} : r.(e.l \circledS h_3)[i] > t$
PROOF: ⟨4⟩1 and assumption 3.
⟨6⟩3. $\forall\, l \in \mathcal{L} : \#e.l \circledS h_{23} = \infty$
$\Rightarrow \forall\, t \in \mathbb{R} : \exists\, i \in \mathbb{N} : r.(e.l \circledS h_{23})[i] > t$
PROOF: ⟨6⟩1, ⟨6⟩2 and ⟨4⟩5.
⟨6⟩4. Q.E.D.
PROOF: Case impossible by ⟨6⟩3.
⟨5⟩4. CASE: $h_{23}$ violates criteria (10), i.e.
$\neg(\forall\, i \in [1..\#h_{23}] : k.h_{23}[i] =! \Rightarrow$
$\#(\{!\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i >$
$\#(\{\sim\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i)$
⟨6⟩1. Choose $i \in [1..\#h_{23}]$ such that $k.h_{23}[i] =!$ and
$\#(\{!\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i \leq$
$\#(\{\sim\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i$
PROOF: ⟨5⟩4 and ¬-rules.
⟨6⟩2. LET: $m = m.h_{23}[i]$
PROOF: ⟨6⟩1.
⟨6⟩3. $(\#!m \in h_2) + (\#!m \in h_3) > 0$
PROOF: ⟨4⟩5, ⟨6⟩1 and ⟨6⟩2.
⟨6⟩4. $(\# \sim m \in h_2) + (\# \sim m \in h_3) > 0$
PROOF: ⟨4⟩5, ⟨6⟩1 and ⟨6⟩2.
⟨6⟩5. $(\#!m \in h_2) + (\#!m \in h_3) = (\# \sim m \in h_2) + (\# \sim m \in h_3)$

PROOF: ⟨6⟩3, ⟨6⟩4, ⟨4⟩1, ⟨4⟩2, assumptions 2 and 3 and observation 2, part 2.

⟨6⟩6. $(\#!m \in h) = (\# \sim m \in h)$

PROOF: ⟨6⟩3, ⟨6⟩4, ⟨4⟩1, ⟨4⟩2 and observation 1.

⟨6⟩7. $(\#!m \in h_1) = (\# \sim m \in h_1)$

PROOF: ⟨6⟩5, ⟨6⟩6, ⟨4⟩1 and ⟨4⟩2.

⟨6⟩8. $\exists\, i' \in [1..\#h] : k.h[i'] =! \wedge$
$\#(\{!\} \times \{m.h[i']\} \times U) \circledS h|_{i'} \leq \#(\{\sim\} \times \{m.h[i']\} \times U) \circledS h|_{i'}$

PROOF: ⟨6⟩1, ⟨6⟩7 and ⟨4⟩5.

⟨6⟩9. Q.E.D.

PROOF: $h$ violates criteria (10) by ⟨6⟩8.

⟨5⟩5. CASE: $h_{23}$ violates criteria (11), i.e.
$\neg(\forall\, i \in [1..\#h] : k.h_{23}[i] =\sim\Rightarrow$
$\#(\{\sim\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i >$
$\#(\{?\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i)$

⟨6⟩1. Choose $i \in [1..\#h_{23}]$ such that $k.h_{23}[i] =\sim$ and
$\#(\{\sim\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i \leq$
$\#(\{?\} \times \{m.h_{23}[i]\} \times U) \circledS h_{23}|_i$

PROOF: ⟨5⟩5 and ¬-rules.

⟨6⟩2. LET: $m = m.h_{23}[i]$

PROOF: ⟨6⟩1.

⟨6⟩3. $(\# \sim m \in h_2) + (\# \sim m \in h_3) > 0$

PROOF: ⟨4⟩5, ⟨6⟩1 and ⟨6⟩2.

⟨6⟩4. $(\#?m \in h_2) + (\#?m \in h_3) > 0$

PROOF: ⟨4⟩5, ⟨6⟩1 and ⟨6⟩2.

⟨6⟩5. $(\# \sim m \in h_2) + (\# \sim m \in h_3) = (\#?m \in h_2) + (\#?m \in h_3)$

PROOF: ⟨6⟩3, ⟨6⟩4, ⟨4⟩1, ⟨4⟩2, assumptions 2 and 3 and observation 2, part 1.

⟨6⟩6. $(\# \sim m \in h) = (\#?m \in h)$

PROOF: ⟨6⟩3, ⟨6⟩4, ⟨4⟩1, ⟨4⟩2 and observation 1.

⟨6⟩7. $(\# \sim m \in h_1) = (\#?m \in h_1)$

PROOF: ⟨6⟩5, ⟨6⟩6, ⟨4⟩1 and ⟨4⟩2.

⟨6⟩8. $\exists\, i' \in [1..\#h] : k.h[i'] =\sim \wedge$
$\#(\{\sim\} \times \{m.h[i']\} \times U) \circledS h|_{i'} \leq \#(\{?\} \times \{m.h[i']\} \times U) \circledS h|_{i'}$

PROOF: ⟨6⟩1, ⟨6⟩7 and ⟨4⟩5.

⟨6⟩9. Q.E.D.

PROOF: $h$ violates criteria (11) by ⟨6⟩8.

⟨5⟩6. Q.E.D.

PROOF: The cases are exhaustive by ⟨4⟩6 and definition of $\mathcal{H}$.

⟨4⟩8. Q.E.D.

PROOF: ⟨4⟩7.

⟨3⟩2. $\{h \in \mathcal{H} \mid \exists\, h_{12} \in s_1 \gtrsim s_2, h_3 \in s_3 : \forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_{12} \frown e.l \circledS h_3\}$
$= \varnothing$

PROOF: ⟨3⟩1.

⟨3⟩3. Q.E.D.

PROOF: $(s_1 \succsim s_2) \succsim s_3 = \varnothing$ by $\langle 3 \rangle 2$ and definition (13) of $\succsim$.
$\langle 2 \rangle 3$. Q.E.D.
PROOF: The cases are exhaustive.
$\langle 1 \rangle 2$. Q.E.D.
PROOF: $\Rightarrow$-rule.

□

**Lemma 10.** *(To be used when proving associativity of weak sequencing in lemma 11)*

*Let $d_1$, $d_2$ and $d_3$ be syntactically well-formed sequence diagrams such that also* $\mathsf{seq}\ [d_1, d_2]$, $\mathsf{seq}\ [d_2, d_3]$ *and* $\mathsf{seq}\ [d_1, \mathsf{seq}\ [d_2, d_3]]$ *are syntactically well-formed.*

ASSUME: *1.* $s_1 \in \mathit{tracesets}(d_1)$
*2.* $s_2 \in \mathit{tracesets}(d_2)$
*3.* $s_3 \in \mathit{tracesets}(d_3)$
PROVE: $s_1 \succsim s_2 = \varnothing \Rightarrow s_1 \succsim (s_2 \succsim s_3) = \varnothing$
PROOF:
Symmetrical to the proof of lemma 9.

□

**Lemma 11.** *Associativity of weak sequencing*

Let $d_1$, $d_2$ and $d_3$ be syntactically well-formed sequence diagrams such that also $\mathsf{seq}\ [d_1, d_2]$, $\mathsf{seq}\ [d_2, d_3], \mathsf{seq}\ [\mathsf{seq}\ [d_1, d_2], d_3]$ and $\mathsf{seq}\ [d_1, [\mathsf{seq}\ [d_2, d_3]]$ are syntactically well-formed.

ASSUME: 1. $s_1 \in \mathit{tracesets}(d_1)$
2. $s_2 \in \mathit{tracesets}(d_2)$
3. $s_3 \in \mathit{tracesets}(d_3)$
PROVE: $(s_1 \succsim s_2) \succsim s_3 = s_1 \succsim (s_2 \succsim s_3)$
$\langle 1 \rangle 1$. CASE: $s_1 \succsim s_2 = \varnothing$
$\langle 2 \rangle 1$. $(s_1 \succsim s_2) \succsim s_3 = \varnothing$
PROOF: Definition (13) of $\succsim$.
$\langle 2 \rangle 2$. $s_1 \succsim (s_2 \succsim s_3) = \varnothing$
PROOF: $\langle 1 \rangle 1$ and lemma 10.
$\langle 2 \rangle 3$. Q.E.D.
$\langle 1 \rangle 2$. CASE: $s_2 \succsim s_3 = \varnothing$
$\langle 2 \rangle 1$. $s_1 \succsim (s_2 \succsim s_3) = \varnothing$
PROOF: Definition (13) of $\succsim$.
$\langle 2 \rangle 2$. $(s_1 \succsim s_2) \succsim s_3 = \varnothing$
PROOF: $\langle 1 \rangle 2$ and lemma 9.
$\langle 2 \rangle 3$. Q.E.D.
$\langle 1 \rangle 3$. CASE: $s_1 \succsim s_2 \neq \varnothing \wedge s_2 \succsim s_3 \neq \varnothing$
PROOF: $(s_1 \succsim s_2) \succsim s_3 = s_1 \succsim (s_2 \succsim s_3)$ by lemma 8.

⟨1⟩4. Q.E.D.
PROOF: The cases are exhaustive.

□

**Lemma 12.** *Left distributivity of parallel execution* $\|$ *over union* $\cup$
PROVE: $s_1 \parallel (s_2 \cup s_3) = (s_1 \parallel s_2) \cup (s_1 \parallel s_3)$

PROOF:

$$
\begin{array}{lll}
& s_1 \parallel (s_2 \cup s_3) & \\
= & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2\}^\infty : & \\
& \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \wedge & \\
& \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_2 \cup s_3\} & \text{by definition (12)} \\
= & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2\}^\infty : & \\
& \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \wedge & \\
& \quad (\pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_2 \vee & \\
& \quad\ \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_3)\} & \text{by definition of } \cup \\
= & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2\}^\infty : & \\
& \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \wedge & \\
& \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_2\} & \\
& \qquad \cup & \\
& \{h \in \mathcal{H} \mid \exists\, p \in \{1,2\}^\infty : & \\
& \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \wedge & \\
& \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_3\} & \\
= & (s_1 \parallel s_2) \cup (s_1 \parallel s_3) & \text{by definition (12)}
\end{array}
$$

□

**Lemma 13.** *Right distributivity of parallel execution* $\|$ *over union* $\cup$
PROVE: $(s_1 \cup s_2) \parallel s_3 = (s_1 \parallel s_3) \cup (s_2 \parallel s_3)$

PROOF:

$$
\begin{array}{lll}
& (s_1 \cup s_2) \parallel s_3 & \\
= & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2\}^\infty : & \\
& \quad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \cup s_2 \wedge & \\
& \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_3\} & \text{by definition (12)} \\
= & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2\}^\infty : & \\
& \quad (\pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_1 \vee & \\
& \quad\ \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_2) \wedge & \\
& \quad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledcirc (p,h)) \in s_3\} & \text{by definition of } \cup
\end{array}
$$

$$
\begin{array}{ll}
= & \{h \in \mathcal{H} \mid \exists\, p \in \{1,2\}^\infty : \\
& \qquad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledT (p,h)) \in s_1 \wedge \\
& \qquad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledT (p,h)) \in s_3\} \\
& \qquad\quad \cup \\
& \{h \in \mathcal{H} \mid \exists\, p \in \{1,2\}^\infty : \\
& \qquad \pi_2((\{1\} \times [\![\ \mathcal{E}\ ]\!]) \circledT (p,h)) \in s_2 \wedge \\
& \qquad \pi_2((\{2\} \times [\![\ \mathcal{E}\ ]\!]) \circledT (p,h)) \in s_3\} \\
= & (s_1 \parallel s_3) \cup (s_2 \parallel s_3) \qquad \text{by definition (12)}
\end{array}
$$

□

**Lemma 14.** *Left distributivity of weak sequencing $\gtrsim$ over union $\cup$*
PROVE: $s_1 \gtrsim (s_2 \cup s_3) = (s_1 \gtrsim s_2) \cup (s_1 \gtrsim s_3)$

PROOF:

$$
\begin{array}{ll}
& s_1 \gtrsim (s_2 \cup s_3) \\
= & \{h \in \mathcal{H} \mid \exists\, h_1 \in s_1, h_2 \in (s_2 \cup s_3) : \\
& \qquad \forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2\} \quad \text{by definition (13)} \\
= & \{h \in \mathcal{H} \mid \exists\, h_1 \in s_1, h_2 \in s_2 : \\
& \qquad \forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2\} \\
& \qquad\quad \cup \\
& \{h \in \mathcal{H} \mid \exists\, h_1 \in s_1, h_2 \in s_3 : \\
& \qquad \forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2\} \\
= & (s_1 \gtrsim s_2) \cup (s_1 \gtrsim s_3) \qquad \text{by definition (13)}
\end{array}
$$

□

**Lemma 15.** *Right distributivity of weak sequencing $\gtrsim$ over union $\cup$*
PROVE: $(s_1 \cup s_2) \gtrsim s_3 = (s_1 \gtrsim s_3) \cup (s_2 \gtrsim s_3)$

PROOF:

$$
\begin{array}{ll}
& (s_1 \cup s_2) \gtrsim s_3 \\
= & \{h \in \mathcal{H} \mid \exists\, h_1 \in (s_1 \cup s_2), h_2 \in s_3 : \\
& \qquad \forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2\} \quad \text{by definition (13)} \\
= & \{h \in \mathcal{H} \mid \exists\, h_1 \in s_1, h_2 \in s_3 : \\
& \qquad \forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2\} \\
& \qquad\quad \cup \\
& \{h \in \mathcal{H} \mid \exists\, h_1 \in s_2, h_2 \in s_3 : \\
& \qquad \forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2\} \\
= & (s_1 \gtrsim s_3) \cup (s_2 \gtrsim s_3) \qquad \text{by definition (13)}
\end{array}
$$

□

**Lemma 16.** *Distributivity of time constraint* $\wr$ *over union* $\cup$
PROVE: $(s_1 \cup s_2) \wr C = (s_1 \wr C) \cup (s_2 \wr C)$

PROOF:

$$
\begin{array}{lll}
(s_1 \cup s_2) \wr C & & \\
= & \{h \in (s_1 \cup s_2) \mid h \models C\} & \text{by definition (14)} \\
= & \{h \in s_1 \mid h \models C\} & \\
& \quad \cup & \\
& \{h \in s_2 \mid h \models C\} & \\
= & (s_1 \wr C) \cup (s_2 \wr C) & \text{by definition (14)}
\end{array}
$$

□

### C.3 Lemmas on interaction obligations

**Lemma 17.** *Commutativity of parallel execution*
PROVE: $(p_1, n_1) \parallel (p_2, n_2) = (p_2, n_2) \parallel (p_1, n_1)$

PROOF:

$(p_1, n_1) \parallel (p_2, n_2)$

$= (p_1 \parallel p_2, (n_1 \parallel (p_2 \cup n_2)) \cup (n_2 \parallel p_1))$ by definition (15)

$= (p_1 \parallel p_2, n_1 \parallel p_2 \cup n_1 \parallel n_2 \cup n_2 \parallel p_1)$ by lemma 12 (distributivity of $\parallel$ over $\cup$)

$= (p_2 \parallel p_1, n_1 \parallel p_2 \cup n_2 \parallel n_1 \cup n_2 \parallel p_1)$ by lemma 3 (commutativity of $\parallel$)

$= (p_2 \parallel p_1, n_2 \parallel p_1 \cup n_2 \parallel n_1 \cup n_1 \parallel p_2)$ by commutativity of $\cup$

$= (p_2 \parallel p_1, (n_2 \parallel (p_1 \cup n_1)) \cup (n_1 \parallel p_2))$ by lemma 12 (distributivity of $\parallel$ over $\cup$)

$= (p_2, n_2) \parallel (p_1, n_1)$ by definition (15)

□

**Lemma 18.** *Associativity of parallel execution*
Let $d_1$, $d_2$ and $d_3$ be syntactically well-formed sequence diagrams such that also $d_1$ par $d_2$, $d_2$ par $d_3$, $(d_1$ par $d_2)$ par $d_3$ and $d_1$ par $(d_2$ par $d_3)$ are syntactically well-formed.

PROVE: $\forall (p_1, n_1) \in [\![\ d_1\ ]\!], (p_2, n_2) \in [\![\ d_2\ ]\!], (p_3, n_3) \in [\![\ d_3\ ]\!]:$
$((p_1, n_1) \parallel (p_2, n_2)) \parallel (p_3, n_3) = (p_1, n_1) \parallel ((p_2, n_2) \parallel (p_3, n_3))$

PROOF:
The two sides of the equation reduce to the same formula.
Left side:

$((p_1, n_1) \parallel (p_2, n_2)) \parallel (p_3, n_3)$

$= (p_1 \parallel p_2, (n_1 \parallel (p_2 \cup n_2)) \cup (n_2 \parallel p_1)) \parallel (p_3, n_3)$ by definition (15)

$= ((p_1 \parallel p_2) \parallel p_3,$
$\quad (((n_1 \parallel (p_2 \cup n_2)) \cup (n_2 \parallel p_1)) \parallel (p_3 \cup n_3)) \cup$
$\quad (n_3 \parallel (p_1 \parallel p_2)))$ by definition (15)

$= ((p_1 \parallel p_2) \parallel p_3,$
$\quad ((n_1 \parallel p_2 \cup n_1 \parallel n_2 \cup n_2 \parallel p_1) \parallel (p_3 \cup n_3)) \cup$ by lemma 12
$\quad (n_3 \parallel (p_1 \parallel p_2)))$ (distributivity of $\parallel$ over $\cup$)

$= ((p_1 \parallel p_2) \parallel p_3,$
$\quad ((n_1 \parallel p_2 \cup n_1 \parallel n_2 \cup n_2 \parallel p_1) \parallel p_3) \cup$
$\quad ((n_1 \parallel p_2 \cup n_1 \parallel n_2 \cup n_2 \parallel p_1) \parallel n_3) \cup$ by lemma 12
$\quad (n_3 \parallel (p_1 \parallel p_2)))$ (distributivity of $\parallel$ over $\cup$)

$= ((p_1 \parallel p_2) \parallel p_3,$
$\quad (n_1 \parallel p_2) \parallel p_3 \cup (n_1 \parallel n_2) \parallel p_3 \cup (n_2 \parallel p_1) \parallel p_3 \cup$
$\quad (n_1 \parallel p_2) \parallel n_3 \cup (n_1 \parallel n_2) \parallel n_3 \cup (n_2 \parallel p_1) \parallel n_3 \cup$ by lemma 12
$\quad n_3 \parallel (p_1 \parallel p_2))$ (distributivity of $\parallel$ over $\cup$)

Right side:

$(p_1, n_1) \parallel ((p_2, n_2) \parallel (p_3, n_3))$

$= (p_1, n_1) \parallel (p_2 \parallel p_3, (n_2 \parallel (p_3 \cup n_3)) \cup (n_3 \parallel p_2))$ by definition (15)

$= (p_1 \parallel (p_2 \parallel p_3),$
$(n_1 \parallel ((p_2 \parallel p_3) \cup ((n_2 \parallel (p_3 \cup n_3)) \cup (n_3 \parallel p_2)))) \cup$
$(((n_2 \parallel (p_3 \cup n_3)) \cup (n_3 \parallel p_2)) \parallel p_1))$ by definition (15)

$= (p_1 \parallel (p_2 \parallel p_3),$
$(n_1 \parallel ((p_2 \parallel p_3) \cup (n_2 \parallel p_3 \cup n_2 \parallel n_3 \cup n_3 \parallel p_2))) \cup$ by lemma 12
$((n_2 \parallel p_3 \cup n_2 \parallel n_3 \cup n_3 \parallel p_2) \parallel p_1))$ (distributivity of $\parallel$ over $\cup$)

$= (p_1 \parallel (p_2 \parallel p_3),$
$n_1 \parallel (p_2 \parallel p_3) \cup n_1 \parallel (n_2 \parallel p_3) \cup n_1 \parallel (n_2 \parallel n_3) \cup$
$n_1 \parallel (n_3 \parallel p_2) \cup (n_2 \parallel p_3) \parallel p_1 \cup (n_2 \parallel n_3) \parallel p_1 \cup$ by lemma 12
$(n_3 \parallel p_2) \parallel p_1)$ (distributivity of $\parallel$ over $\cup$)

$= (p_1 \parallel (p_2 \parallel p_3),$
$n_1 \parallel (p_2 \parallel p_3) \cup n_1 \parallel (n_2 \parallel p_3) \cup (n_2 \parallel p_3) \parallel p_1) \cup$
$n_1 \parallel (n_3 \parallel p_2) \cup n_1 \parallel (n_2 \parallel n_3) \cup (n_2 \parallel n_3) \parallel p_1 \cup$
$(n_3 \parallel p_2) \parallel p_1)$ by commutativity of $\cup$

$= ((p_1 \parallel p_2) \parallel p_3,$
$(n_1 \parallel p_2) \parallel p_3 \cup (n_1 \parallel n_2) \parallel p_3 \cup (n_2 \parallel p_1) \parallel p_3 \cup$
$(n_1 \parallel p_2) \parallel n_3 \cup (n_1 \parallel n_2) \parallel n_3 \cup (n_2 \parallel p_1) \parallel n_3 \cup$ by lemmas 3 and 7
$n_3 \parallel (p_1 \parallel p_2))$ (commutativity and associativity of $\parallel$)

□

**Lemma 19.** *Associativity of weak sequencing*
Let $d_1$, $d_2$ and $d_3$ be syntactically well-formed sequence diagrams such that also $\mathsf{seq}\ [d_1, d_2]$, $\mathsf{seq}\ [d_2, d_3]$, $\mathsf{seq}\ [\mathsf{seq}\ [d_1, d_2], d_3]$ and $\mathsf{seq}\ [d_1, \mathsf{seq}\ [d_2, d_3]]$ are syntactically well-formed.

PROVE: $\forall (p_1, n_1) \in [\![\ d_1\ ]\!], (p_2, n_2) \in [\![\ d_2\ ]\!], (p_3, n_3) \in [\![\ d_3\ ]\!]:$
$((p_1, n_1) \succsim (p_2, n_2)) \succsim (p_3, n_3) = (p_1, n_1) \succsim ((p_2, n_2) \succsim (p_3, n_3))$

PROOF:
The two sides of the equation reduce to the same formula.
Left side:

$((p_1, n_1) \succsim (p_2, n_2)) \succsim (p_3, n_3)$

$= (p_1 \succsim p_2, (n_1 \succsim (n_2 \cup p_2)) \cup (p_1 \succsim n_2)) \succsim (p_3, n_3)$ by definition (16)

$= ((p_1 \succsim p_2) \succsim p_3,$
$(((n_1 \succsim (n_2 \cup p_2)) \cup (p_1 \succsim n_2)) \succsim (n_3 \cup p_3)) \cup$
$((p_1 \succsim p_2) \succsim n_3))$ by definition (16)

$$
\begin{array}{lll}
= & ((p_1 \succsim p_2) \succsim p_3, & \\
& \quad ((n_1 \succsim n_2 \cup n_1 \succsim p_2 \cup p_1 \succsim n_2) \succsim (n_3 \cup p_3)) \cup & \text{by lemma 14} \\
& \quad ((p_1 \succsim p_2) \succsim n_3)) & \text{(distributivity of } \succsim \text{ over } \cup\text{)} \\
= & ((p_1 \succsim p_2) \succsim p_3, & \\
& \quad ((n_1 \succsim n_2 \cup n_1 \succsim p_2 \cup p_1 \succsim n_2) \succsim n_3) \cup & \\
& \quad ((n_1 \succsim n_2 \cup n_1 \succsim p_2 \cup p_1 \succsim n_2) \succsim p_3) \cup & \text{by lemma 14} \\
& \quad ((p_1 \succsim p_2) \succsim n_3)) & \text{(distributivity of } \succsim \text{ over } \cup\text{)} \\
= & ((p_1 \succsim p_2) \succsim p_3, & \\
& \quad (n_1 \succsim n_2) \succsim n_3 \cup (n_1 \succsim p_2) \succsim n_3 \cup (p_1 \succsim n_2) \succsim n_3 \cup & \\
& \quad (n_1 \succsim n_2) \succsim p_3 \cup (n_1 \succsim p_2) \succsim p_3 \cup (p_1 \succsim n_2) \succsim p_3 \cup & \text{by lemma 15} \\
& \quad (p_1 \succsim p_2) \succsim n_3) & \text{(distributivity of } \succsim \text{ over } \cup\text{)}
\end{array}
$$

Right side:

$$
\begin{array}{lll}
(p_1, n_1) \succsim ((p_2, n_2) \succsim (p_3, n_3)) & & \\
= & (p_1, n_1) \succsim (p_2 \succsim p_3, (n_2 \succsim (n_3 \cup p_3)) \cup (p_2 \succsim n_3)) & \text{by definition (16)} \\
= & (p_1 \succsim (p_2 \succsim p_3), & \\
& \quad (n_1 \succsim (((n_2 \succsim (n_3 \cup p_3)) \cup (p_2 \succsim n_3)) \cup (p_2 \succsim p_3))) \cup & \\
& \quad (p_1 \succsim ((n_2 \succsim (n_3 \cup p_3)) \cup (p_2 \succsim n_3)))) & \text{by definition (16)} \\
= & (p_1 \succsim (p_2 \succsim p_3), & \\
& \quad (n_1 \succsim (n_2 \succsim n_3 \cup n_2 \succsim p_3 \cup p_2 \succsim n_3 \cup p_2 \succsim p_3)) \cup & \text{by lemma 14} \\
& \quad (p_1 \succsim (n_2 \succsim n_3 \cup n_2 \succsim p_3 \cup p_2 \succsim n_3))) & \text{(distributivity of } \succsim \text{ over } \cup\text{)} \\
= & (p_1 \succsim (p_2 \succsim p_3), & \\
& \quad n_1 \succsim (n_2 \succsim n_3) \cup n_1 \succsim (n_2 \succsim p_3) \cup n_1 \succsim (p_2 \succsim n_3) \cup & \\
& \quad n_1 \succsim (p_2 \succsim p_3) \cup p_1 \succsim (n_2 \succsim n_3) \cup p_1 \succsim (n_2 \succsim p_3) \cup & \text{by lemma 14} \\
& \quad p_1 \succsim (p_2 \succsim n_3)) & \text{(distributivity of } \succsim \text{ over } \cup\text{)} \\
= & (p_1 \succsim (p_2 \succsim p_3), & \\
& \quad n_1 \succsim (n_2 \succsim n_3) \cup n_1 \succsim (p_2 \succsim n_3) \cup p_1 \succsim (n_2 \succsim n_3) \cup & \\
& \quad n_1 \succsim (n_2 \succsim p_3) \cup n_1 \succsim (p_2 \succsim p_3) \cup p_1 \succsim (n_2 \succsim p_3) \cup & \\
& \quad p_1 \succsim (p_2 \succsim n_3)) & \text{by commutativity of } \cup \\
= & ((p_1 \succsim p_2) \succsim p_3, & \\
& \quad (n_1 \succsim n_2) \succsim n_3 \cup (n_1 \succsim p_2) \succsim n_3 \cup (p_1 \succsim n_2) \succsim n_3 \cup & \\
& \quad (n_1 \succsim n_2) \succsim p_3 \cup (n_1 \succsim p_2) \succsim p_3 \cup (p_1 \succsim n_2) \succsim p_3 \cup & \\
& \quad (p_1 \succsim p_2) \succsim n_3) & \text{by lemma 11 (associativity of } \succsim\text{)}
\end{array}
$$

□

### C.4 Lemmas on sets of interaction obligations

**Lemma 20.** *Commutativity of inner union*
PROVE: $O_1 \uplus O_2 = O_2 \uplus O_1$

PROOF:

$O_1 \uplus O_2$

$= \{(p_1 \cup p_2, n_1 \cup n_2) \mid (p_1, n_1) \in O_1 \wedge (p_2, n_2) \in O_2\}$ by definition (30)

$= \{(p_2 \cup p_1, n_2 \cup n_1) \mid (p_2, n_2) \in O_1 \wedge (p_1, n_1) \in O_2\}$ by renaming

$= \{(p_1 \cup p_2, n_1 \cup n_2) \mid (p_1, n_1) \in O_2 \wedge (p_2, n_2) \in O_1\}$ by commutativity of $\cup$ and $\wedge$

$= O_2 \uplus O_1$ by definition (30)

□

**Lemma 21.** *Associativity of inner union*
PROVE: $(O_1 \uplus O_2) \uplus O_3 = O_1 \uplus (O_2 \uplus O_3)$

PROOF:
The two sides of the equation reduce to the same formula.
Left side:

$(O_1 \uplus O_2) \uplus O_3$

$= \{(p_1 \cup p_2, n_1 \cup n_2) \mid (p_1, n_1) \in O_1 \uplus O_2 \wedge (p_2, n_2) \in O_3\}$ by definition (30)

$= \{(p_{12} \cup p_3, n_{12} \cup n_3) \mid (p_{12}, n_{12}) \in O_1 \uplus O_2 \wedge (p_3, n_3) \in O_3\}$ by renaming

$= \{(p_{12} \cup p_3, n_{12} \cup n_3) \mid$
$(p_{12}, n_{12}) \in \{(p_1 \cup p_2, n_1 \cup n_2) \mid (p_1, n_1) \in O_1 \wedge (p_2, n_2) \in O_2\} \wedge$
$(p_3, n_3) \in O_3\}$ by definition (30)

$= \{((p_1 \cup p_2) \cup p_3, (n_1 \cup n_2) \cup n_3) \mid$
$(p_1, n_1) \in O_1 \wedge (p_2, n_2) \in O_2 \wedge (p_3, n_3) \in O_3\}$

$= \{(p_1 \cup p_2 \cup p_3, n_1 \cup n_2 \cup n_3) \mid$ by commutativity
$(p_1, n_1) \in O_1 \wedge (p_2, n_2) \in O_2 \wedge (p_3, n_3) \in O_3\}$ of $\cup$

Right side:

$O_1 \uplus (O_2 \uplus O_3)$

$= \{(p_1 \cup p_2, n_1 \cup n_2) \mid (p_1, n_1) \in O_1 \wedge (p_2, n_2) \in O_2 \uplus O_3\}$ by definition (30)

$= \{(p_1 \cup p_{23}, n_1 \cup n_{23}) \mid (p_1, n_1) \in O_1 \wedge (p_{23}, n_{23}) \in O_2 \uplus O_3\}$ by renaming

$= \{(p_1 \cup p_{23}, n_1 \cup n_{23}) \mid (p_1, n_1) \in O_1 \wedge$
$(p_{23}, n_{23}) \in \{(p_1 \cup p_2, n_1 \cup n_2) \mid (p_1, n_1) \in O_2 \wedge (p_2, n_2) \in O_3\}\}$ by definition (30)

$= \{(p_1 \cup p_{23}, n_1 \cup n_{23}) \mid (p_1, n_1) \in O_1 \wedge$
$(p_{23}, n_{23}) \in \{(p_2 \cup p_3, n_2 \cup n_3) \mid (p_2, n_2) \in O_2 \wedge (p_3, n_3) \in O_3\}\}$ by renaming

$= \{(p_1 \cup (p_2 \cup p_3), n_1 \cup (n_2 \cup n_3)) \mid$
$(p_1, n_1) \in O_1 \wedge (p_2, n_2) \in O_2 \wedge (p_3, n_3) \in O_3\}$

$=$ $\{(p_1 \cup p_2 \cup p_3, n_1 \cup n_2 \cup n_3) \mid$ by commutativity
$(p_1, n_1) \in O_1 \wedge (p_2, n_2) \in O_2 \wedge (p_3, n_3) \in O_3\}$ of $\cup$

□

**Lemma 22.** *Commutativity of parallel execution*
PROVE: $O_1 \parallel O_2 = O_2 \parallel O_1$

PROOF:
$O_1 \parallel O_2$
$=$ $\{o_1 \parallel o_2 \mid o_1 \in O_1 \wedge o_2 \in O_2\}$ by definition (18)
$=$ $\{o_2 \parallel o_1 \mid o_2 \in O_1 \wedge o_1 \in O_2\}$ by renaming
$=$ $\{o_1 \parallel o_2 \mid o_1 \in O_2 \wedge o_2 \in O_1\}$ by commutativity of $\parallel$ (lemma 17) and $\wedge$
$=$ $O_2 \parallel O_1$ by definition (18)

□

**Lemma 23.** *Associativity of parallel execution*
Let $d_1$, $d_2$ and $d_3$ be syntactically well-formed sequence diagrams such that also $d_1$ par $d_2$, $d_2$ par $d_3$, $(d_1$ par $d_2)$ par $d_3$ and $d_1$ par $(d_2$ par $d_3)$ are syntactically well-formed.

ASSUME: 1. $[\![ d_1 ]\!] = O_1$
2. $[\![ d_2 ]\!] = O_2$
3. $[\![ d_3 ]\!] = O_3$
PROVE: $(O_1 \parallel O_2) \parallel O_3 = O_1 \parallel (O_2 \parallel O_3)$

PROOF:
The two sides of the equation reduce to the same formula.
Left side:
$(O_1 \parallel O_2) \parallel O_3$
$=$ $\{o_1 \parallel o_2 \mid o_1 \in O_1 \parallel O_2 \wedge o_2 \in O_3\}$ by definition (18)
$=$ $\{o_{12} \parallel o_3 \mid o_{12} \in O_1 \parallel O_2 \wedge o_3 \in O_3\}$ by renaming
$=$ $\{o_{12} \parallel o_3 \mid$
$o_{12} \in \{o_1 \parallel o_2 \mid o_1 \in O_1 \wedge o_2 \in O_2\} \wedge$
$o_3 \in O_3\}$ by definition (18)
$=$ $\{(o_1 \parallel o_2) \parallel o_3 \mid$
$o_1 \in O_1 \wedge o_2 \in O_2 \wedge o_3 \in O_3\}$
Right side:
$O_1 \parallel (O_2 \parallel O_3)$
$=$ $\{o_1 \parallel o_2 \mid o_1 \in O_1 \wedge o_2 \in O_2 \parallel O_3\}$ by definition (18)
$=$ $\{o_1 \parallel o_{23} \mid o_1 \in O_1 \wedge o_{23} \in O_2 \parallel O_3\}$ by renaming

$$
\begin{array}{lll}
= & \{o_1 \parallel o_{23} \mid o_1 \in O_1 \wedge & \\
& \quad o_{23} \in \{o_1 \parallel o_2 \mid o_1 \in O_2 \wedge o_2 \in O_3\}\} & \text{by definition (18)} \\
= & \{o_1 \parallel o_{23} \mid o_1 \in O_1 \wedge & \\
& \quad o_{23} \in \{o_2 \parallel o_3 \mid o_2 \in O_2 \wedge o_3 \in O_3\}\} & \text{by renaming} \\
= & \{o_1 \parallel (o_2 \parallel o_3) \mid & \\
& \quad o_1 \in O_1 \wedge o_2 \in O_2 \wedge o_3 \in O_3\} & \\
= & \{(o_1 \parallel o_2) \parallel o_3 \mid & \\
& \quad o_1 \in O_1 \wedge o_2 \in O_2 \wedge o_3 \in d_3\} & \text{by lemma 18 (associativity of } \parallel\text{)}
\end{array}
$$

□

**Lemma 24.** *Associativity of weak sequencing*
Let $d_1$, $d_2$ and $d_3$ be syntactically well-formed sequence diagrams such that also $\mathsf{seq}\ [d_1, d_2]$, $\mathsf{seq}\ [d_2, d_3]$, $\mathsf{seq}\ [\mathsf{seq}\ [d_1, d_2], d_3]$ and $\mathsf{seq}\ [d_1, \mathsf{seq}\ [d_2, d_3]]$ are syntactically well-formed.

ASSUME: 1. $[\![\ d_1\ ]\!] = O_1$
2. $[\![\ d_2\ ]\!] = O_2$
3. $[\![\ d_3\ ]\!] = O_3$

PROVE: $(O_1 \succsim O_2) \succsim O_3 = O_1 \succsim (O_2 \succsim O_3)$

PROOF:
The two sides of the equation reduce to the same formula.
Left side:

$$
\begin{array}{lll}
(O_1 \succsim O_2) \succsim O_3 & & \\
= & \{o_1 \succsim o_2 \mid o_1 \in O_1 \succsim O_2 \wedge o_2 \in O_3\} & \text{by definition (19)} \\
= & \{o_{12} \succsim o_3 \mid o_{12} \in O_1 \succsim O_2 \wedge o_3 \in O_3\} & \text{by renaming} \\
= & \{o_{12} \succsim o_3 \mid & \\
& \quad o_{12} \in \{o_1 \succsim o_2 \mid o_1 \in O_1 \wedge o_2 \in O_2\} \wedge & \\
& \quad o_3 \in O_3\} & \text{by definition (19)} \\
= & \{(o_1 \succsim o_2) \succsim o_3 \mid & \\
& \quad o_1 \in O_1 \wedge o_2 \in O_2 \wedge o_3 \in O_3\} &
\end{array}
$$

Right side:

$$
\begin{array}{lll}
O_1 \succsim (O_2 \succsim O_3) & & \\
= & \{o_1 \succsim o_2 \mid o_1 \in O_1 \wedge o_2 \in O_2 \succsim O_3\} & \text{by definition (19)} \\
= & \{o_1 \succsim o_{23} \mid o_1 \in O_1 \wedge o_{23} \in O_2 \succsim O_3\} & \text{by renaming} \\
= & \{o_1 \succsim o_{23} \mid o_1 \in O_1 \wedge & \\
& \quad o_{23} \in \{o_1 \succsim o_2 \mid o_1 \in O_2 \wedge o_2 \in O_3\}\} & \text{by definition (19)} \\
= & \{o_1 \succsim o_{23} \mid o_1 \in O_1 \wedge & \\
& \quad o_{23} \in \{o_2 \succsim o_3 \mid o_2 \in O_2 \wedge o_3 \in O_3\}\} & \text{by renaming}
\end{array}
$$

$$
\begin{aligned}
&= \{o_1 \succsim (o_2 \succsim o_3) \mid \\
&\qquad o_1 \in O_1 \wedge o_2 \in O_2 \wedge o_3 \in O_3\} \\
&= \{(o_1 \succsim o_2) \succsim o_3 \mid \\
&\qquad o_1 \in O_1 \wedge o_2 \in O_2 \wedge o_3 \in O_3\} \quad \text{by lemma 19 (associativity of } \succsim \text{)}
\end{aligned}
$$

□

### C.5 Theorems on sequence diagram operators

**Theorem 1.** *Commutativity of potential alternative*
PROVE: $d_1 \,\mathsf{alt}\, d_2 = d_2 \,\mathsf{alt}\, d_1$

PROOF:

$$
\begin{aligned}
&[\![\ d_1 \,\mathsf{alt}\, d_2\ ]\!] \\
&\quad = [\![\ d_1\ ]\!] \uplus [\![\ d_2\ ]\!] && \text{by definition (37)} \\
&\quad = [\![\ d_2\ ]\!] \uplus [\![\ d_1\ ]\!] && \text{by commutativity of } \uplus \text{ (lemma 20)} \\
&\quad = [\![\ d_2 \,\mathsf{alt}\, d_1\ ]\!] && \text{by definition (37)}
\end{aligned}
$$

□

**Theorem 2.** *Associativity of potential alternative*
PROVE: $(d_1 \,\mathsf{alt}\, d_2) \,\mathsf{alt}\, d_3 = d_1 \,\mathsf{alt}\, (d_2 \,\mathsf{alt}\, d_3)$

PROOF:

$$
\begin{aligned}
&[\![\ (d_1 \,\mathsf{alt}\, d_2) \,\mathsf{alt}\, d_3\ ]\!] \\
&\quad = [\![\ d_1 \,\mathsf{alt}\, d_2\ ]\!] \uplus [\![\ d_3\ ]\!] && \text{by definition (37)} \\
&\quad = ([\![\ d_1\ ]\!] \uplus [\![\ d_2\ ]\!]) \uplus [\![\ d_3\ ]\!] && \text{by definition (37)} \\
&\quad = [\![\ d_1\ ]\!] \uplus ([\![\ d_2\ ]\!] \uplus [\![\ d_3\ ]\!]) && \text{by lemma 21 (associativity of } \uplus\text{)} \\
&\quad = [\![\ d_1\ ]\!] \uplus [\![\ d_2 \,\mathsf{alt}\, d_3\ ]\!] && \text{by definition (37)} \\
&\quad = [\![\ d_1 \,\mathsf{alt}\, (d_2 \,\mathsf{alt}\, d_3)\ ]\!] && \text{by definition (37)}
\end{aligned}
$$

□

**Theorem 3.** *Commutativity of mandatory alternative*
PROVE: $d_1 \,\mathsf{xalt}\, d_2 = d_2 \,\mathsf{xalt}\, d_1$

PROOF:

$$
\begin{aligned}
&[\![\ d_1 \,\mathsf{xalt}\, d_2\ ]\!] \\
&\quad = [\![\ d_1\ ]\!] \cup [\![\ d_2\ ]\!] && \text{by definition (38)} \\
&\quad = [\![\ d_2\ ]\!] \cup [\![\ d_1\ ]\!] && \text{by commutativity of } \cup \\
&\quad = [\![\ d_2 \,\mathsf{xalt}\, d_1\ ]\!] && \text{by definition (38)}
\end{aligned}
$$

□

**Theorem 4.** *Associativity of mandatory alternative*
PROVE: $(d_1 \,\mathsf{xalt}\, d_2) \,\mathsf{xalt}\, d_3 = d_1 \,\mathsf{xalt}\, (d_2 \,\mathsf{xalt}\, d_3)$

PROOF:

$$
\begin{aligned}
&[\![\ (d_1 \,\mathsf{xalt}\, d_2) \,\mathsf{xalt}\, d_3\ ]\!] \\
&\quad = [\![\ d_1 \,\mathsf{xalt}\, d_2\ ]\!] \cup [\![\ d_3\ ]\!] && \text{by definition (38)} \\
&\quad = ([\![\ d_1\ ]\!] \cup [\![\ d_2\ ]\!]) \cup [\![\ d_3\ ]\!] && \text{by definition (38)} \\
&\quad = [\![\ d_1\ ]\!] \cup ([\![\ d_2\ ]\!] \cup [\![\ d_3\ ]\!]) && \text{by associativity of } \cup \\
&\quad = [\![\ d_1\ ]\!] \cup [\![\ d_2 \,\mathsf{xalt}\, d_3\ ]\!] && \text{by definition (38)}
\end{aligned}
$$

$= \quad [\![\ d_1 \,\mathsf{xalt}\, (d_2 \,\mathsf{xalt}\, d_3)\ ]\!]$ by definition (38)

□

**Theorem 5.** *Commutativity of parallel execution*
PROVE: $d_1 \,\mathsf{par}\, d_2 = d_2 \,\mathsf{par}\, d_1$

PROOF:
$[\![\ d_1 \,\mathsf{par}\, d_2\ ]\!]$
$= \quad [\![\ d_1\ ]\!] \parallel [\![\ d_2\ ]\!]$ by definition (39)
$= \quad [\![\ d_2\ ]\!] \parallel [\![\ d_1\ ]\!]$ by commutativity of $\parallel$ (lemma 22)
$= \quad [\![\ d_2 \,\mathsf{par}\, d_1\ ]\!]$ by definition (39)

□

**Theorem 6.** *Associativity of parallel execution*
PROVE: $(d_1 \,\mathsf{par}\, d_2) \,\mathsf{par}\, d_3 = d_1 \,\mathsf{par}\, (d_2 \,\mathsf{par}\, d_3)$

PROOF:
$[\![\ (d_1 \,\mathsf{par}\, d_2) \,\mathsf{par}\, d_3\ ]\!]$
$= \quad [\![\ d_1 \,\mathsf{par}\, d_2\ ]\!] \parallel [\![\ d_3\ ]\!]$ by definition (39)
$= \quad ([\![\ d_1\ ]\!] \parallel [\![\ d_2\ ]\!]) \parallel [\![\ d_3\ ]\!]$ by definition (39)
$= \quad [\![\ d_1\ ]\!] \parallel ([\![\ d_2\ ]\!] \parallel [\![\ d_3\ ]\!])$ by lemma 23 (associativity of $\parallel$)
$= \quad [\![\ d_1\ ]\!] \parallel [\![\ d_2 \,\mathsf{par}\, d_3\ ]\!]$ by definition (39)
$= \quad [\![\ d_1 \,\mathsf{par}\, (d_2 \,\mathsf{par}\, d_3)\ ]\!]$ by definition (39)

□

**Theorem 7.** *Associativity of weak sequencing*
PROVE: $\mathsf{seq}\,[\mathsf{seq}\,[d_1, d_2], d_3] = \mathsf{seq}\,[d_1, \mathsf{seq}\,[d_2, d_3]]$

PROOF:
$[\![\ \mathsf{seq}\,[\mathsf{seq}\,[d_1, d_2], d_3]\ ]\!]$
$= \quad [\![\ \mathsf{seq}\,[\mathsf{seq}\,[d_1, d_2]]\ ]\!] \succsim [\![\ d_3\ ]\!]$ by definition (36)
$= \quad [\![\ \mathsf{seq}\,[d_1, d_2]\ ]\!] \succsim [\![\ d_3\ ]\!]$ by definition (36)
$= \quad ([\![\ \mathsf{seq}\,[d_1]\ ]\!] \succsim [\![\ d_2\ ]\!]) \succsim [\![\ d_3\ ]\!]$ by definition (36)
$= \quad ([\![\ d_1\ ]\!] \succsim [\![\ d_2\ ]\!]) \succsim [\![\ d_3\ ]\!]$ by definition (36)
$= \quad [\![\ d_1\ ]\!] \succsim ([\![\ d_2\ ]\!] \succsim [\![\ d_3\ ]\!])$ by lemma 24 (associativity of $\succsim$)
$= \quad [\![\ d_1\ ]\!] \succsim ([\![\ \mathsf{seq}\,[d_2]\ ]\!] \succsim [\![\ d_3\ ]\!])$ by definition (36)
$= \quad [\![\ d_1\ ]\!] \succsim [\![\ \mathsf{seq}\,[d_2, d_3]\ ]\!]$ by definition (36)
$= \quad [\![\ \mathsf{seq}\,[d_1]\ ]\!] \succsim [\![\ \mathsf{seq}\,[d_2, d_3]\ ]\!]$ by definition (36)
$= \quad [\![\ \mathsf{seq}\,[d_1, \mathsf{seq}\,[d_2, d_3]]\ ]\!]$ by definition (36)

□

## D Reflexivity and transitivity

In this section we prove that refinement as defined in this paper is reflexive and transitive.

### D.1 Reflexivity

**Lemma 25.** *Reflexivity of* $\rightsquigarrow_r$
ASSUME: $o = (p, n)$
PROVE: $o \rightsquigarrow_r o$

⟨1⟩1. Requirement 1: $n \subseteq n$
  PROOF: Trivial.
⟨1⟩2. Requirement 2: $p \subseteq p \cup n$
  PROOF: Trivial.
⟨1⟩3. Q.E.D.
  PROOF: Definition (45) of $\rightsquigarrow_r$.

□

**Theorem 8.** *Reflexivity of the refinement operator* $\rightsquigarrow_g$
PROVE: $d \rightsquigarrow_g d$,
  i.e. $[\![ d ]\!] \rightsquigarrow_g [\![ d ]\!]$ by definition (47),
  i.e. $\forall o \in [\![ d ]\!] : \exists o' \in [\![ d ]\!] : o \rightsquigarrow_r o'$ by definition (46).

⟨1⟩1. $\exists o' \in [\![ d ]\!] : o \rightsquigarrow_r o'$ for arbitrary $o \in [\![ d ]\!]$
  ⟨2⟩1. Choose $o' = o$
    PROOF: $o \in [\![ d ]\!]$ by ⟨1⟩1.
  ⟨2⟩2. $o \rightsquigarrow_r o$
    PROOF: Lemma 25 (reflexivity of $\rightsquigarrow_r$).
  ⟨2⟩3. Q.E.D.
⟨1⟩2. Q.E.D.
  PROOF: ∀-rule.

□

### D.2 Transitivity

**Lemma 26.** *Transitivity of* $\rightsquigarrow_r$
ASSUME: 1. $(p, n) \rightsquigarrow_r (p', n')$
  2. $(p', n') \rightsquigarrow_r (p'', n'')$
PROVE: $(p, n) \rightsquigarrow_r (p'', n'')$

⟨1⟩1. $n \subseteq n''$
  ⟨2⟩1. $n \subseteq n'$
    PROOF: Assumption 1 and definition (45) of $\rightsquigarrow_r$.
  ⟨2⟩2. $n' \subseteq n''$
    PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.
  ⟨2⟩3. Q.E.D.

PROOF: $\langle 2\rangle 1$, $\langle 2\rangle 2$, and transitivity of $\subseteq$.

$\langle 1\rangle 2$. $p \subseteq p'' \cup n''$

$\langle 2\rangle 1$. $p \subseteq p' \cup n'$

PROOF: Assumption 1 and definition (45) of $\rightsquigarrow_r$.

$\langle 2\rangle 2$. $p' \subseteq p'' \cup n''$

PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.

$\langle 2\rangle 3$. $n' \subseteq n''$

PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.

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

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

$p \subseteq p' \cup n' \wedge p' \subseteq p'' \cup n'' \wedge n' \subseteq n''$

$\Downarrow$

$p \subseteq p'' \cup n' \cup n''$

$\Downarrow$

$p \subseteq p'' \cup n'' \cup n''$

$\Downarrow$

$p \subseteq p'' \cup n''$

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

PROOF: By definition (45) of $\rightsquigarrow_r$.

□

**Theorem 9.** *Transitivity of the refinement operator* $\rightsquigarrow_g$

ASSUME: 1. $d \rightsquigarrow_g d'$

2. $d' \rightsquigarrow_g d''$

PROVE: $d \rightsquigarrow_g d''$

$\langle 1\rangle 1$. $[\![ d ]\!] \rightsquigarrow_g [\![ d'' ]\!]$

$\langle 2\rangle 1$. $\forall o \in [\![ d ]\!] : \exists o'' \in [\![ d'' ]\!] : o \rightsquigarrow_r o''$

$\langle 3\rangle 1$. $\forall (p, n) \in [\![ d ]\!] : \exists (p'', n'') \in [\![ d'' ]\!] : (p, n) \rightsquigarrow_r (p'', n'')$

$\langle 4\rangle 1$. $\exists (p'', n'') \in [\![ d'' ]\!] : (p, n) \rightsquigarrow_r (p'', n'')$ for arbitrary $(p, n) \in [\![ d ]\!]$

$\langle 5\rangle 1$. Choose $(p', n') \in [\![ d' ]\!]$ and $(p'', n'') \in [\![ d'' ]\!]$ such that

1. $(p, n) \rightsquigarrow_r (p', n')$
2. $(p', n') \rightsquigarrow_r (p'', n'')$

PROOF: Assumptions 1 and 2, and definitions (47) and (46) of $\rightsquigarrow_g$.

$\langle 5\rangle 2$. $(p, n) \rightsquigarrow_r (p'', n'')$

PROOF: $\langle 5\rangle 1$ and lemma 26 (transitivity of $\rightsquigarrow_r$).

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

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

PROOF: $\forall$-rule.

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

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

PROOF: By definition (46) of $\rightsquigarrow_g$.

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

PROOF: By definition (47) of $\rightsquigarrow_g$.

□

# E Monotonicity

In this section we prove that the refinement operator $\rightsquigarrow_g$ is monotonic with respect to the composition operators neg, alt, xalt, seq, loop, par and tc, meaning that refining one operand will give a refinement of the whole composition.
In general, we do not have monotonicity with respect to assert. However, we prove that in the special case of narrowing, written $\rightsquigarrow_{g,n}$, we do also have monotonicity with respect to assert.
First, in Section E.1 we prove monotonicity of $\rightsquigarrow_r$ with respect to weak sequencing, parallel execution and time constraint on trace sets. Then, in Section E.2 we prove monotonicity of $\rightsquigarrow_g$ with respect to weak sequencing, parallel execution, time constraint, inner union and looping on sets of interaction obligations. Finally, in Section E.3 we prove the monotonicity theorems for $\rightsquigarrow_g$ with respect to the sequence diagram operators.

## E.1 Monotonicity of $\rightsquigarrow_r$

**Lemma 27.** *(To be used when proving monotonicity with respect to $\succsim$)*
ASSUME: 1. $s_1 \subseteq s_1'$
2. $s_2 \subseteq s_2'$
PROVE: $s_1 \succsim s_2 \subseteq s_1' \succsim s_2'$

$\langle 1 \rangle 1$. CASE: $s_1 \succsim s_2 = \varnothing$
PROOF: Trivial, as $\varnothing \subseteq A$ for all sets $A$.
$\langle 1 \rangle 2$. CASE: $s_1 \succsim s_2 \neq \varnothing$
$\langle 2 \rangle 1$. Choose arbitrary $h \in s_1 \succsim s_2$
PROOF: Non-empty by case assumption.
$\langle 2 \rangle 2$. $h \in s_1' \succsim s_2'$
$\langle 3 \rangle 1$. Choose $h_1 \in s_1$ and $h_2 \in s_2$ such that $\forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2$
PROOF: $\langle 2 \rangle 1$ and definition (13) of $\succsim$.
$\langle 3 \rangle 2$. $h_1 \in s_1'$
PROOF: $\langle 3 \rangle 1$ and assumption 1.
$\langle 3 \rangle 3$. $h_2 \in s_2'$
PROOF: $\langle 3 \rangle 1$ and assumption 2.
$\langle 3 \rangle 4$. $h \in s_1' \succsim s_2'$
PROOF: $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 3 \rangle 3$ and definition (13) of $\succsim$.
$\langle 3 \rangle 5$. Q.E.D.
$\langle 2 \rangle 3$. Q.E.D.
PROOF: $\langle 2 \rangle 1$, $\langle 2 \rangle 2$ and definition of $\subseteq$.
$\langle 1 \rangle 3$. Q.E.D.
PROOF: The cases are exhaustive.

□

**Lemma 28.** *(To be used when proving monotonicity with respect to $\|$)*
ASSUME: 1. $s_1 \subseteq s_1'$
2. $s_2 \subseteq s_2'$
PROVE: $s_1 \parallel s_2 \subseteq s_1' \parallel s_2'$

$\langle 1 \rangle 1$. CASE: $s_1 \succsim s_2 = \varnothing$
PROOF: Trivial, as $\varnothing \subseteq A$ for all sets $A$.
$\langle 1 \rangle 2$. CASE: $s_1 \parallel s_2 \neq \varnothing$
$\langle 2 \rangle 1$. Choose arbitrary $h \in s_1 \parallel s_2$
PROOF: Non-empty by case assumption.
$\langle 2 \rangle 2$. $h \in s_1' \parallel s_2'$
$\langle 3 \rangle 1$. Choose $p \in \{1, 2\}^\infty$ such that
$\pi_2((\{1\} \times [\![ \mathcal{E} ]\!]) \circledcirc (p, h)) \in s_1$ and
$\pi_2((\{2\} \times [\![ \mathcal{E} ]\!]) \circledcirc (p, h)) \in s_2$
PROOF: $\langle 2 \rangle 1$ and definition (12) of $\|$.
$\langle 3 \rangle 2$. $\pi_2((\{1\} \times [\![ \mathcal{E} ]\!]) \circledcirc (p, h)) \in s_1'$
PROOF: $\langle 3 \rangle 1$ and assumption 1.
$\langle 3 \rangle 3$. $\pi_2((\{2\} \times [\![ \mathcal{E} ]\!]) \circledcirc (p, h)) \in s_2'$
PROOF: $\langle 3 \rangle 1$ and assumption 2.
$\langle 3 \rangle 4$. $h \in s_1' \parallel s_2'$
PROOF: $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 3 \rangle 3$ and definition (12) of $\|$.
$\langle 3 \rangle 5$. Q.E.D.
$\langle 2 \rangle 3$. Q.E.D.
PROOF: $\langle 2 \rangle 1$, $\langle 2 \rangle 2$ and definition of $\subseteq$.
$\langle 1 \rangle 3$. Q.E.D.
PROOF: The cases are exhaustive.

□

**Lemma 29.** *(To be used when proving monotonicity with respect to $\wr$)*
ASSUME: $s \subseteq s'$
PROVE: $s \wr C \subseteq s' \wr C$

$\langle 1 \rangle 1$. CASE: $s \wr C = \varnothing$
PROOF: Trivial, as $\varnothing \subseteq A$ for all sets $A$.
$\langle 1 \rangle 2$. CASE: $s \wr C \neq \varnothing$
$\langle 2 \rangle 1$. Choose arbitrary $h \in s \wr C$
PROOF: Non-empty by case assumption.
$\langle 2 \rangle 2$. $h \in s' \wr C$
$\langle 3 \rangle 1$. $h \in \{h \in s \mid h \models C\}$
PROOF: $\langle 2 \rangle 1$ and definition (14) of $\wr$.
$\langle 3 \rangle 2$. $h \in \{h \in s' \mid h \models C\}$
PROOF: $\langle 3 \rangle 1$ and the assumption.
$\langle 3 \rangle 3$. Q.E.D.
PROOF: $\langle 3 \rangle 2$ and definition (14) of $\wr$.
$\langle 3 \rangle 4$. Q.E.D.
$\langle 2 \rangle 3$. Q.E.D.
PROOF: $\langle 2 \rangle 1$, $\langle 2 \rangle 2$ and definition of $\subseteq$.

⟨1⟩3. Q.E.D.
PROOF: The cases are exhaustive.

□

**Lemma 30.** *Monotonicity of $\rightsquigarrow_r$ with respect to $\succsim$*
ASSUME: 1. $(p, n) = (p_1, n_1) \succsim (p_2, n_2)$,
i.e. $(p, n) = (p_1 \succsim p_2, (n_1 \succsim (n_2 \cup p_2)) \cup (p_1 \succsim n_2))$ by definition (16).
2. $(p_1, n_1) \rightsquigarrow_r (p_1', n_1')$
3. $(p_2, n_2) \rightsquigarrow_r (p_2', n_2')$
4. $(p', n') = (p_1', n_1') \succsim (p_2', n_2')$,
i.e. $(p', n') = (p_1' \succsim p_2', (n_1' \succsim (n_2' \cup p_2')) \cup (p_1' \succsim n_2'))$ by definition (16).
PROVE: $(p, n) \rightsquigarrow_r (p', n')$

⟨1⟩1. Requirement 1: $n \subseteq n'$,
i.e. $(n_1 \succsim (n_2 \cup p_2)) \cup (p_1 \succsim n_2) \subseteq (n_1' \succsim (n_2' \cup p_2')) \cup (p_1' \succsim n_2')$
⟨2⟩1. $n_1 \succsim n_2 \subseteq n_1' \succsim n_2'$
⟨3⟩1. $n_1 \subseteq n_1'$
PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩2. $n_2 \subseteq n_2'$
PROOF: Assumption 3 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩2 and lemma 27.
⟨2⟩2. $n_1 \succsim p_2 \subseteq n_1' \succsim (n_2' \cup p_2')$
⟨3⟩1. $n_1 \subseteq n_1'$
PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩2. $p_2 \subseteq p_2' \cup n_2'$
PROOF: Assumption 3 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩3. $n_1 \succsim p_2 \subseteq n_1' \succsim (p_2' \cup n_2')$
PROOF: ⟨3⟩1, ⟨3⟩2 and lemma 27.
⟨3⟩4. Q.E.D.
PROOF: ⟨3⟩3 and associativity of $\cup$.
⟨2⟩3. $p_1 \succsim n_2 \subseteq (p_1' \succsim n_2') \cup (n_1' \succsim n_2')$
⟨3⟩1. $p_1 \subseteq p_1' \cup n_1'$
PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩2. $n_2 \subseteq n_2'$
PROOF: Assumption 3 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩3. $p_1 \succsim n_2 \subseteq (p_1' \cup n_1') \succsim n_2'$
PROOF: ⟨3⟩1, ⟨3⟩2 and lemma 27.
⟨3⟩4. Q.E.D.
PROOF: ⟨3⟩3 and lemma 15 (distributivity of $\succsim$ over $\cup$).
⟨2⟩4. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩2 and ⟨2⟩3 and lemma 14 (distributivity of $\succsim$ over $\cup$).
⟨1⟩2. Requirement 2: $p \subseteq p' \cup n'$,
i.e. $p_1 \succsim p_2 \subseteq (p_1' \succsim p_2') \cup (n_1' \succsim (n_2' \cup p_2')) \cup (p_1' \succsim n_2')$

⟨2⟩1. $p_1 \subseteq p'_1 \cup n'_1$
PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.
⟨2⟩2. $p_2 \subseteq p'_2 \cup n'_2$
PROOF: Assumption 3 and definition (45) of $\rightsquigarrow_r$.
⟨2⟩3. $p_1 \succsim p_2 \subseteq (p'_1 \cup n'_1) \succsim (p'_2 \cup n'_2)$
PROOF: ⟨2⟩1, ⟨2⟩2 and lemma 27.
⟨2⟩4. Q.E.D.
PROOF: ⟨2⟩3 and lemmas 14 and 15 (distributivity of $\succsim$ over $\cup$).
⟨1⟩3. Q.E.D.
PROOF: Assumptions 1 and 4 and definition (45) of $\rightsquigarrow_r$.

□

**Lemma 31.** *Monotonicity of $\rightsquigarrow_r$ with respect to $\|$*
ASSUME: 1. $(p, n) = (p_1, n_1) \parallel (p_2, n_2)$,
i.e. $(p, n) = (p_1 \parallel p_2, (n_1 \parallel (p_2 \cup n_2)) \cup (n_2 \parallel p_1))$ by definition (15).
2. $(p_1, n_1) \rightsquigarrow_r (p'_1, n'_1)$
3. $(p_2, n_2) \rightsquigarrow_r (p'_2, n'_2)$
4. $(p', n') = (p'_1, n'_1) \parallel (p'_2, n'_2)$,
i.e. $(p', n') = (p'_1 \parallel p'_2, (n'_1 \succsim (p'_2 \cup n'_2)) \cup (n'_2 \parallel p'_1))$ by definition (15).
PROVE: $(p, n) \rightsquigarrow_r (p', n')$

⟨1⟩1. Requirement 1: $n \subseteq n'$,
i.e. $(n_1 \parallel (p_2 \cup n_2)) \cup (n_2 \parallel p_1) \subseteq (n'_1 \parallel (p'_2 \cup n'_2)) \cup (n'_2 \parallel p'_1)$
⟨2⟩1. $n_1 \parallel n_2 \subseteq n'_1 \parallel n'_2$
⟨3⟩1. $n_1 \subseteq n'_1$
PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩2. $n_2 \subseteq n'_2$
PROOF: Assumption 3 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩2 and lemma 28.
⟨2⟩2. $n_1 \parallel p_2 \subseteq n'_1 \parallel (p'_2 \cup n'_2)$
⟨3⟩1. $n_1 \subseteq n'_1$
PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩2. $p_2 \subseteq p'_2 \cup n'_2$
PROOF: Assumption 3 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩2 and lemma 27.
⟨2⟩3. $n_2 \parallel p_1 \subseteq (n'_2 \parallel p'_1) \cup (n'_2 \parallel n'_1)$
⟨3⟩1. $n_2 \subseteq n'_2$
PROOF: Assumption 3 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩2. $p_1 \subseteq p'_1 \cup n'_1$
PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩3. $n_2 \parallel p_1 \subseteq n'_2 \parallel (p'_1 \cup n'_1)$
PROOF: ⟨3⟩2, ⟨3⟩1 and lemma 28.
⟨3⟩4. Q.E.D.

PROOF: ⟨3⟩3 and lemma 15 (distributivity of $\gtrsim$ over $\cup$).

⟨2⟩4. Q.E.D.

PROOF: ⟨2⟩1, ⟨2⟩2 and ⟨2⟩3, lemma 3 (commutativity of $\|$) and lemma 12 (distributivity of $\|$ over $\cup$).

⟨1⟩2. Requirement 2: $p \subseteq p' \cup n'$,
i.e. $p_1 \parallel p_2 \subseteq (p'_1 \parallel p'_2) \cup (n'_1 \parallel (p'_2 \cup n'_2)) \cup (n'_2 \parallel p'_1)$

⟨2⟩1. $p_1 \subseteq p'_1 \cup n'_1$

PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.

⟨2⟩2. $p_2 \subseteq p'_2 \cup n'_2$

PROOF: Assumption 3 and definition (45) of $\rightsquigarrow_r$.

⟨2⟩3. $p_1 \parallel p_2 \subseteq (p'_1 \cup n'_1) \gtrsim (p'_2 \parallel n'_2)$

PROOF: ⟨2⟩1, ⟨2⟩2 and lemma 28.

⟨2⟩4. Q.E.D.

PROOF: ⟨2⟩3, lemma 3 (commutativity of $\|$) and lemmas 12 and 13 (distributivity of $\|$ over $\cup$).

⟨1⟩3. Q.E.D.

PROOF: Assumptions 1 and 4 and definition (45) of $\rightsquigarrow_r$.

□

**Lemma 32.** *Monotonicity of $\rightsquigarrow_r$ with respect to $\wr$*

ASSUME: 1. $(p, n) = (p_1, n_1) \wr C$,
i.e. $(p, n) = (p_1 \wr C, n_1 \cup (p_1 \wr \neg C))$ by definition (17).
2. $(p_1, n_1) \rightsquigarrow_r (p'_1, n'_1)$
3. $(p', n') = (p'_1, n'_1) \wr C$,
i.e. $(p', n') = (p'_1 \wr C, n'_1 \cup (p'_1 \wr \neg C))$ by definition (17).

PROVE: $(p, n) \rightsquigarrow_r (p', n')$

⟨1⟩1. Requirement 1: $n \subseteq n'$,
i.e. $n_1 \cup (p_1 \wr \neg C) \subseteq n'_1 \cup (p'_1 \wr \neg C)$

⟨2⟩1. $n_1 \subseteq n'_1$

PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.

⟨2⟩2. $p_1 \wr \neg C \subseteq n'_1 \cup (p'_1 \wr \neg C)$

⟨3⟩1. $p_1 \subseteq p'_1 \cup n'_1$

PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.

⟨3⟩2. $p_1 \wr \neg C \subseteq (p'_1 \cup n'_1) \wr \neg C$

PROOF: ⟨3⟩1 and lemma 29.

⟨3⟩3. $p_1 \wr \neg C \subseteq (p'_1 \wr \neg C) \cup (n'_1 \wr \neg C)$

PROOF: ⟨3⟩2 and lemma 16 (distributivity of $\wr$ over$\cup$).

⟨3⟩4. $n'_1 \wr \neg C \subseteq n'_1$

PROOF: Definition (14) of $\wr$.

⟨3⟩5. Q.E.D.

PROOF: ⟨3⟩3, ⟨3⟩4 and associativity of $\cup$.

⟨2⟩3. Q.E.D.

PROOF: ⟨2⟩1 and ⟨2⟩2.

⟨1⟩2. Requirement 2: $p \subseteq p' \cup n'$,
i.e. $p_1 \wr C \subseteq (p'_1 \wr C) \cup (n'_1 \cup (p'_1 \wr \neg C))$

⟨2⟩1. $p_1 \subseteq p_1' \cup n_1'$
PROOF: Assumption 2 and definition (45) of $\rightsquigarrow_r$.
⟨2⟩2. $p_1 \wr C \subseteq (p_1' \cup n_1') \wr C$
PROOF: ⟨2⟩1 and lemma 29.
⟨2⟩3. $p_1 \wr C \subseteq (p_1' \wr C) \cup (n_1' \wr C)$
PROOF: ⟨2⟩3 and lemma 16 (distributivity of $\wr$ over $\cup$.
⟨2⟩4. $n_1' \wr C \subseteq n_1'$
PROOF: Definition (14) of $\wr$.
⟨2⟩5. Q.E.D.
PROOF: ⟨2⟩3 and ⟨2⟩4.
⟨1⟩3. Q.E.D.
PROOF: Assumptions 1 and 3 and definition (45) of $\rightsquigarrow_r$.

□

### E.2 Monotonicity of $\rightsquigarrow_g$ with respect to operators on sets of interaction obligations

**Lemma 33.** *Monotonicity of $\rightsquigarrow_g$ with respect to $\succsim$ on sets of interaction obligations*

ASSUME: 1. $O = O_1 \succsim O_2$,
i.e. $O = \{o_1 \succsim o_2 \mid o_1 \in O_1 \wedge o_2 \in O_2\}$ by definition (19).
2. $O_1 \rightsquigarrow_g O_1'$,
i.e. $\forall\, o_1 \in O_1 : \exists\, o_1' \in O_1' : o_1 \rightsquigarrow_r o_1'$ by definition (46).
3. $O_2 \rightsquigarrow_g O_2'$,
i.e. $\forall\, o_2 \in O_2 : \exists\, o_2' \in O_2' : o_2 \rightsquigarrow_r o_2'$ by definition (46).
4. $O' = O_1' \succsim O_2'$,
i.e. $O' = \{o_1 \succsim o_2 \mid o_1 \in O_1' \wedge o_2 \in O_2'\}$ by definition (19).

PROVE: $O \rightsquigarrow_g O'$,
i.e. $O_1 \succsim O_2 \rightsquigarrow_g O_1' \succsim O_2'$,
i.e. $\forall\, o \in O_1 \succsim O_2 : \exists\, o' \in O_1' \succsim O_2' : o \rightsquigarrow_r o'$ by definition (46).

⟨1⟩1. $\exists\, o' \in O_1' \succsim O_2' : o \rightsquigarrow_r o'$ for arbitrary $o \in O_1 \succsim O_2$
⟨2⟩1. Choose $o_1 \in O_1$ and $o_2 \in O_2$ such that $o = o_1 \succsim o_2$
PROOF: Assumption 1.
⟨2⟩2. Choose $o_1' \in O_1'$ such that $o_1 \rightsquigarrow_r o_1'$
PROOF: Assumption 2.
⟨2⟩3. Choose $o_2' \in O_2'$ such that $o_2 \rightsquigarrow_r o_2'$
PROOF: Assumption 3.
⟨2⟩4. $o' = o_1' \succsim o_2' \in O_1' \succsim O_2'$
PROOF: Assumption 4.
⟨2⟩5. $o \rightsquigarrow_r o'$, i.e. $o_1 \succsim o_2 \rightsquigarrow_r o_1' \succsim o_2'$
PROOF: Lemma 30 with $(p, n) = o$, $(p', n') = o'$, $(p_1, n_1) = o_1$, $(p_2, n_2) = o_2$, $(p_1', n_1') = o_1'$ and $(p_2', n_2') = o_2'$.
⟨2⟩6. Q.E.D.
⟨1⟩2. Q.E.D.
PROOF: ∀-rule.

□

**Lemma 34.** *Monotonicity of $\rightsquigarrow_g$ with respect to $\parallel$ on sets of interaction obligations*

ASSUME: 1. $O = O_1 \parallel O_2$,
i.e. $O = \{o_1 \parallel o_2 \mid o_1 \in O_1 \wedge o_2 \in O_2\}$ by definition (18).
2. $O_1 \rightsquigarrow_g O_1'$,
i.e. $\forall\, o_1 \in O_1 : \exists\, o_1' \in O_1' : o_1 \rightsquigarrow_r o_1'$ by definition (46).
3. $O_2 \rightsquigarrow_g O_2'$,
i.e. $\forall\, o_2 \in O_2 : \exists\, o_2' \in O_2' : o_2 \rightsquigarrow_r o_2'$ by definition (46).
4. $O' = O_1' \parallel O_2'$,
i.e. $O' = \{o_1 \parallel o_2 \mid o_1 \in O_1' \wedge o_2 \in O_2'\}$ by definition (18).

PROVE: $O \rightsquigarrow_g O'$,
i.e. $O_1 \parallel O_2 \rightsquigarrow_g O_1' \parallel O_2'$,
i.e. $\forall\, o \in O_1 \parallel O_2 : \exists\, o' \in O_1' \parallel O_2' : o \rightsquigarrow_r o'$ by definition (46).

⟨1⟩1. $\exists\, o' \in O_1' \parallel O_2' : o \rightsquigarrow_r o'$ for arbitrary $o \in O_1 \parallel O_2$
⟨2⟩1. Choose $o_1 \in O_1$ and $o_2 \in O_2$ such that $o = o_1 \parallel o_2$
PROOF: Assumption 1.
⟨2⟩2. Choose $o_1' \in O_1'$ such that $o_1 \rightsquigarrow_r o_1'$
PROOF: Assumption 2.
⟨2⟩3. Choose $o_2' \in O_2'$ such that $o_2 \rightsquigarrow_r o_2'$
PROOF: Assumption 3.
⟨2⟩4. $o' = o_1' \parallel o_2' \in O_1' \parallel O_2'$
PROOF: Assumption 4.
⟨2⟩5. $o \rightsquigarrow_r o'$, i.e. $o_1 \parallel o_2 \rightsquigarrow_r o_1' \parallel o_2'$
PROOF: Lemma 31 with $(p, n) = o$, $(p', n') = o'$, $(p_1, n_1) = o_1$, $(p_2, n_2) = o_2$, $(p_1', n_1') = o_1'$ and $(p_2', n_2') = o_2'$.
⟨2⟩6. Q.E.D.
⟨1⟩2. Q.E.D.
PROOF: ∀-rule.

□

**Lemma 35.** *Monotonicity of $\rightsquigarrow_g$ with respect to $\wr$ on sets of interaction obligations*
ASSUME: 1. $O = O_1 \wr C$,
i.e. $O = \{o_1 \wr C \mid o_1 \in O_1\}$ by definition (20).
2. $O_1 \rightsquigarrow_g O_1'$,
i.e. $\forall\, o_1 \in O_1 : \exists\, o_1' \in O_1' : o_1 \rightsquigarrow_r o_1'$ by definition (46).
3. $O' = O_1' \wr C$,
i.e. $O' = \{o_1 \wr C \mid o_1 \in O_1'\}$ by definition (20).
PROVE: $O \rightsquigarrow_g O'$,
i.e. $O_1 \wr C \rightsquigarrow_g O_1' \wr C$,
i.e. $\forall\, o \in O_1 \wr C : \exists\, o' \in O_1' \wr C : o \rightsquigarrow_r o'$ by definition (46).

⟨1⟩1. $\exists\, o' \in O_1' \wr C : o \rightsquigarrow_r o'$ for arbitrary $o \in O_1 \wr C$
⟨2⟩1. Choose $o_1 \in O_1$ such that $o = o_1 \wr C$
PROOF: Assumption 1.
⟨2⟩2. Choose $o_1' \in O_1'$ such that $o_1 \rightsquigarrow_r o_1'$
PROOF: Assumption 2.
⟨2⟩3. $o' = o_1' \wr C \in O_1' \wr C$
PROOF: Assumption 3.
⟨2⟩4. $o \rightsquigarrow_r o'$, i.e. $o_1 \wr C \rightsquigarrow_r o_1' \wr C$
PROOF: Lemma 32 with $(p, n) = o$, $(p', n') = o'$, $(p_1, n_1) = o_1$, and $(p_1', n_1') = o_1'$.
⟨2⟩5. Q.E.D.
⟨1⟩2. Q.E.D.
PROOF: ∀-rule.

□

**Lemma 36.** *Monotonicity of $\rightsquigarrow_g$ with respect to $\uplus$ on sets of interaction obligations*

ASSUME: 1. $O = O_1 \uplus O_2$,
i.e. $O = \{(p_1 \cup p_2, n_1 \cup n_2) \mid (p_1, n_1) \in O_1 \wedge (p_2, n_2) \in O_2\}$ by definition (30).
2. $O_1 \rightsquigarrow_g O_1'$,
i.e. $\forall\, o_1 \in O_1 : \exists\, o_1' \in O_1' : o_1 \rightsquigarrow_r o_1'$ by definition (46).
3. $O_2 \rightsquigarrow_g O_2'$,
i.e. $\forall\, o_2 \in O_2 : \exists\, o_2' \in O_2' : o_2 \rightsquigarrow_r o_2'$ by definition (46).
4. $O' = O_1' \uplus O_2'$,
i.e. $O' = \{(p_1 \cup p_2, n_1 \cup n_2) \mid (p_1, n_1) \in O_1' \wedge (p_2, n_2) \in O_2'\}$ by definition (30).

PROVE: $O \rightsquigarrow_g O'$,
i.e. $\forall\, o \in O : \exists\, o' \in O' : o \rightsquigarrow_r o'$ by definition (46).

$\langle 1 \rangle 1$. $\exists\, o' \in O' : o \rightsquigarrow_r o'$ for arbitrary $o = (p, n) \in O$
$\langle 2 \rangle 1$. Choose $(p_1, n_1) \in O_1$ and $(p_2, n_2) \in O_2$ such that $p = p_1 \cup p_2$ and $n = n_1 \cup n_2$
PROOF: Assumption 1.
$\langle 2 \rangle 2$. Choose $(p_1', n_1') \in O_1'$ such that $(p_1, n_1) \rightsquigarrow_r (p_1', n_1')$
PROOF: Assumption 2.
$\langle 2 \rangle 3$. Choose $(p_2', n_2') \in O_2'$ such that $(p_2, n_2) \rightsquigarrow_r (p_2', n_2')$
PROOF: Assumption 3.
$\langle 2 \rangle 4$. $o' = (p', n') = (p_1' \cup p_2', n_1' \cup n_2') \in O'$
PROOF: Assumption 4.
$\langle 2 \rangle 5$. $(p, n) \rightsquigarrow_r (p', n')$
$\langle 3 \rangle 1$. Requirement 1: $n \subseteq n'$, i.e. $n_1 \cup n_2 \subseteq n_1' \cup n_2'$
PROOF: $n_1 \subseteq n_1'$ (by $\langle 2 \rangle 2$ and definition (45) of $\rightsquigarrow_r$) and $n_2 \subseteq n_2'$ (by $\langle 2 \rangle 3$ and definition (45) of $\rightsquigarrow_r$).
$\langle 3 \rangle 2$. Requirement 2: $p \subseteq p' \cup n'$, i.e. $p_1 \cup p_2 \subseteq (p_1' \cup p_2') \cup (n_1' \cup n_2')$
PROOF: $p_1 \subseteq p_1' \cup n_1'$ (by $\langle 2 \rangle 2$ and definition (45) of $\rightsquigarrow_r$) and $p_2 \subseteq p_2' \cup n_2'$ (by $\langle 2 \rangle 3$ and definition (45) of $\rightsquigarrow_r$).
$\langle 3 \rangle 3$. Q.E.D.
PROOF: Definition (45) of $\rightsquigarrow_r$.
$\langle 2 \rangle 6$. Q.E.D.
$\langle 1 \rangle 2$. Q.E.D.
PROOF: $\forall$-rule.

□

**Lemma 37.** *Monotonicity of $\rightsquigarrow_g$ with respect to $\biguplus$ on sets of interaction obligations*

ASSUME: 1. $O = \biguplus_{i \in I} O_i$,
i.e. $O = \{\biguplus_{i \in I} o_i \mid o_i \in O_i\}$ by definition (31),
i.e. $O = \{(\cup_{i \in I} p_i, \cup_{i \in I} n_i) \mid (p_i, n_i) \in O_i\}$ by definition (32).
2. $\forall\, i \in I : O_i \rightsquigarrow_g O_i'$,
i.e. $\forall\, i \in I : \forall\, o \in O_i : \exists\, o' \in O_i' : o \rightsquigarrow_r o'$ by definition (46).

3. $O' = \biguplus_{i \in I} O'_i$,
i.e. $O' = \{\biguplus_{i \in I} o_i \mid o_i \in O'_i\}$ by definition (31),
i.e. $O' = \{(\cup_{i \in I} p_i, \cup_{i \in I} n_i) \mid (p_i, n_i) \in O'_i\}$ by definition (32).

PROVE: $O \rightsquigarrow_g O'$,
i.e. $\forall\, o \in O : \exists\, o' \in O' : o \rightsquigarrow_r o'$ by definition (46).

⟨1⟩1. $\exists\, o' \in O' : o \rightsquigarrow_r o'$ for arbitrary $o = (p, n) \in O$
⟨2⟩1. For all $i \in I$, choose $(p_i, n_i) \in O_i$ such that $p = \bigcup_{i \in I} p_i$ and $n = \bigcup_{i \in I} n_i$
PROOF: Assumption 1.
⟨2⟩2. For all $i \in I$, choose $(p'_i, n'_i) \in O'_i$ such that $(p_i, n_i) \rightsquigarrow_r (p'_i, n'_i)$
PROOF: Assumption 2.
⟨2⟩3. $o' = (p', n') = (\bigcup_{i \in I} p'_i, \bigcup_{i \in I} n'_i) \in O'$
PROOF: Assumption 3.
⟨2⟩4. $(p, n) \rightsquigarrow_r (p', n')$
⟨3⟩1. Requirement 1: $n \subseteq n'$, i.e. $\bigcup_{i \in I} n_i \subseteq \bigcup_{i \in I} n'_i$
PROOF: $\forall\, i \in I : n_i \subseteq n'_i$ by ⟨2⟩2 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩2. Requirement 2: $p \subseteq p' \cup n'$,
i.e. $\bigcup_{i \in I} p_i \subseteq \bigcup_{i \in I} p'_i \cup \bigcup_{i \in I} n'_i$
PROOF: $\forall\, i \in I : p_i \subseteq p'_i \cup n'_i$ by ⟨2⟩2 and definition (45) of $\rightsquigarrow_r$.
⟨3⟩3. Q.E.D.
PROOF: Definition (45) of $\rightsquigarrow_r$.
⟨1⟩2. Q.E.D.
PROOF: ∀-rule.

□

**Lemma 38.** *Monotonicity of $\rightsquigarrow_g$ with respect to the inductive definition of $\mu_n$*

ASSUME: $O \rightsquigarrow_g O'$,
i.e. $\forall\, o \in O : \exists\, o' \in O' : o \rightsquigarrow_r o'$ by definition (46).

PROVE: $\mu_n\, O \rightsquigarrow_g \mu_n\, O'$,
i.e. $\forall\, o \in \mu_n\, O : \exists\, o' \in \mu_n\, O' : o \rightsquigarrow_r o'$ by definition (46).

⟨1⟩1. $\exists\, o' \in \mu_n\, O' : o \rightsquigarrow_r o'$ for arbitrary $o \in \mu_n\, O$
⟨2⟩1. CASE: $n = 0$
⟨3⟩1. $\mu_0\, O = \{(\{\langle\rangle\}, \varnothing)\}$, i.e. $o = (\{\langle\rangle\}, \varnothing)$
PROOF: Definition (26) of $\mu_0$.
⟨3⟩2. $o = (\{\langle\rangle\}, \varnothing) \in \mu_0\, O'$
PROOF: Definition (26) of $\mu_0$.
⟨3⟩3. $o \rightsquigarrow_r o$
PROOF: Lemma 25 (reflexivity of $\rightsquigarrow_r$).
⟨3⟩4. Q.E.D.
⟨2⟩2. CASE: $n = 1$
⟨3⟩1. $o \in O$
PROOF: Definition (27) of $\mu_1$.
⟨3⟩2. Choose $o' \in O'$ such that $o \rightsquigarrow_r o'$
PROOF: ⟨3⟩1 and the assumption.
⟨3⟩3. $o' \in \mu_1\, O'$

PROOF: ⟨3⟩2 and definition (27) of $\mu_1$.

⟨3⟩4. Q.E.D.

⟨2⟩3. CASE: $1 < n < \infty$

⟨3⟩1. ASSUME: $\mu_k\ O \rightsquigarrow_g \mu_k\ O'$ (induction hypothesis).
PROVE: $\mu_{k+1}\ O \rightsquigarrow_g \mu_{k+1}\ O'$,
i.e. $O \gtrsim \mu_k\ O \rightsquigarrow_g O' \gtrsim \mu_k\ O'$ by definition (28) of $\mu_n$.

⟨4⟩1. $O \rightsquigarrow_g O'$
PROOF: The main assumption.

⟨4⟩2. $\mu_k\ O \rightsquigarrow_g \mu_k\ O'$
PROOF: The induction hypothesis.

⟨4⟩3. Q.E.D.
PROOF: Lemma 33 (monotonicity of $\rightsquigarrow_g$ with respect to $\gtrsim$) with $O_1 = O$, $O_2 = \mu_k\ O$, $O_1' = O'$ and $O_2' = \mu_k\ O'$.

⟨3⟩2. Q.E.D.
PROOF: Standard induction on n, base cases proved by ⟨2⟩1 and ⟨2⟩2.

⟨2⟩4. CASE: $n = \infty$

PROOF SKETCH: For each chain $\bar{o} \in \mathit{chains}(O)$, we may construct a chain $\bar{o}'$ such that $\forall j \in \mathbb{N} : \bar{o}[j] \rightsquigarrow_r \bar{o}'[j]$ due to the lemma assumption. Each trace in the chains $\mathit{pos}(\bar{o})$ and $\mathit{negs}(\bar{o})$ is selected from a corresponding obligation in the chain $\bar{o}$. By definition (45) of $\rightsquigarrow_r$, these traces are also contained (as positive or negative) in the corresponding obligations in $\bar{o}'$.

⟨3⟩1. $\exists\, o' \in \{\sqcup\bar{o}' \mid \bar{o}' \in \mathit{chains}(O')\} : o \rightsquigarrow_r o'$ for arbitrary $o \in \{\sqcup\bar{o} \mid \bar{o} \in \mathit{chains}(O)\}$

⟨4⟩1. $\exists\, \bar{o}' \in \mathit{chains}(O') : \sqcup\bar{o} \rightsquigarrow_r \sqcup\bar{o}'$ for arbitrary $\bar{o} \in \mathit{chains}(O)$

⟨5⟩1. Choose $\bar{o}' \in \mathit{chains}(O')$ such that $\forall j \in \mathbb{N} : \bar{o}[j] \rightsquigarrow_r \bar{o}'[j]$

⟨6⟩1. $\bar{o}[1] \in O$
PROOF: Definition (21) of *chains*.

⟨6⟩2. Choose $\bar{o}'[1] \in O'$ such that $\bar{o}[1] \rightsquigarrow_r \bar{o}'[1]$
PROOF: ⟨6⟩1 and the assumption.

⟨6⟩3. $\forall j \in \mathbb{N}$ :
choose $o \in O$ such that $\bar{o}[j+1] = \bar{o}[j] \gtrsim o$,
choose $o' \in O'$ such that $o \rightsquigarrow_r o'$,
and let $\bar{o}'[j+1] = \bar{o}'[j] \gtrsim o'$
PROOF: Definition (21) of *chains* and the assumption.

⟨6⟩4. $\forall j > 1 : \bar{o}[j] \rightsquigarrow_r \bar{o}'[j]$

⟨7⟩1. ASSUME: $\bar{o}[j] \rightsquigarrow_r \bar{o}'[j]$ (induction hypothesis).
PROVE: $\bar{o}[j+1] \rightsquigarrow_r \bar{o}'[j+1]$,
i.e. $\bar{o}[j] \gtrsim o \rightsquigarrow_r \bar{o}'[j] \gtrsim o'$by ⟨6⟩3.

⟨8⟩1. $\bar{o}[j] \rightsquigarrow_r \bar{o}'[j]$
PROOF: The induction hypothesis.

⟨8⟩2. $o \rightsquigarrow_r o'$
PROOF: ⟨6⟩3.

⟨8⟩3. Q.E.D.

PROOF: Lemma 30 (monotonicity of $\rightsquigarrow_r$ with respect to $\gtrsim$) with $(p, n) = \bar{o}[j+1]$, $(p', n') = \bar{o}'[j+1]$, $(p_1, n_1) = \bar{o}[j]$, $(p_2, n_2) = o$, $(p'_1, n'_1) = \bar{o}'[j]$ and $(p'_2, n'_2) = o'$.

⟨7⟩2. Q.E.D.

PROOF: Standard induction on $j$, base case proved by ⟨6⟩2.

⟨6⟩5. $\bar{o}' \in \mathit{chains}(O')$

PROOF: ⟨6⟩2, ⟨6⟩3 and definition (21) of *chains*.

⟨6⟩6. Q.E.D.

PROOF: ⟨6⟩2, ⟨6⟩4 and ⟨6⟩5.

⟨5⟩2. $\sqcup\bar{o} \rightsquigarrow_r \sqcup\bar{o}'$

⟨6⟩1. Requirement 1: $\cup_{\bar{t}\in \mathit{negs}(\bar{o})} \sqcup \bar{t} \subseteq \cup_{\bar{t}\in \mathit{negs}(\bar{o}')} \sqcup \bar{t}$

⟨7⟩1. $\forall\, \bar{t} \in \mathit{negs}(\bar{o}) : \bar{t} \in \mathit{negs}(\bar{o}')$

⟨8⟩1. $\bar{t} \in \mathit{negs}(\bar{o}')$ for arbitrary $\bar{t} \in \mathit{negs}(\bar{o})$

⟨9⟩1. Choose $i$ such that $\forall\, j \in \mathbb{N} : \bar{t}[j] \in \pi_2(\bar{o}[j+i-1])$

PROOF: Definition (23) of *negs*.

⟨9⟩2. $\forall\, j \in \mathbb{N} : \bar{t}[j] \in \pi_2(\bar{o}'[j+i-1])$

PROOF: $\forall\, j \in \mathbb{N} : \pi_2(\bar{o}[j]) \subseteq \pi_2(\bar{o}'[j])$ by ⟨5⟩1 and definition (45) of $\rightsquigarrow_r$.

⟨9⟩3. $\forall\, j \in \mathbb{N} : \exists\, t \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \gtrsim \{t\}$

PROOF: ⟨8⟩1 and definition (23) of *negs*.

⟨9⟩4. Q.E.D.

PROOF: ⟨9⟩1, ⟨9⟩2, ⟨9⟩3 and definition (23) of *negs*.

⟨8⟩2. Q.E.D.

PROOF: ∀-rule.

⟨7⟩2. Q.E.D.

PROOF: Definition of $\cup$ and $\subseteq$.

⟨6⟩2. Requirement 2: $\cup_{\bar{t}\in \mathit{pos}(\bar{o})} \sqcup \bar{t} \subseteq (\cup_{\bar{t}\in \mathit{pos}(\bar{o}')} \sqcup t) \cup (\cup_{\bar{t}\in \mathit{negs}(\bar{o}')} \sqcup \bar{t})$

⟨7⟩1. $\forall\, \bar{t} \in \mathit{pos}(\bar{o}) : \sqcup\bar{t} \subseteq (\cup_{\bar{t}\in \mathit{pos}(\bar{o}')} \sqcup t) \cup (\cup_{\bar{t}\in \mathit{negs}(\bar{o}')} \sqcup \bar{t})$

⟨8⟩1. $\sqcup\bar{t} \subseteq (\cup_{\bar{t}\in \mathit{pos}(\bar{o}')} \sqcup t) \cup (\cup_{\bar{t}\in \mathit{negs}(\bar{o}')} \sqcup \bar{t})$ for arbitrary $\bar{t} \in \mathit{pos}(\bar{o})$

⟨9⟩1. $\forall\, j \in \mathbb{N} : \bar{t}[j] \in \pi_1(\bar{o}[j])$

PROOF: Definition (22) of *pos*.

⟨9⟩2. $\forall\, j \in \mathbb{N} : \bar{t}[j] \in \pi_1(\bar{o}'[j]) \cup \pi_2(\bar{o}'[j])$

PROOF: ⟨5⟩1 and definition (45) of $\rightsquigarrow_r$.

⟨9⟩3. $\forall\, j \in \mathbb{N} : \exists\, t \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \gtrsim \{t\}$

PROOF: ⟨8⟩1 and definition (22) of *pos*.

⟨9⟩4. CASE: ASSUME: $\forall\, j \in \mathbb{N} : \bar{t}[j] \in \pi_1(\bar{o}'[j])$
PROVE: $\sqcup\bar{t} \subseteq \cup_{\bar{t}\in \mathit{pos}(\bar{o}')} \sqcup \bar{t}$

PROOF: $\bar{t} \in \mathit{pos}(\bar{o}')$ by ⟨9⟩3, the case assumption and definition (22) of *pos*.

⟨9⟩5. CASE: ASSUME: $\exists\, i \in \mathbb{N} : (\bar{t}[i] \in \pi_2(\bar{o}'[i]) \wedge \forall\, j < i : \bar{t}[j] \in \pi_1(\bar{o}'[j]))$
PROVE: $\sqcup\bar{t} \subseteq \cup_{\bar{t}\in \mathit{negs}(\bar{o}')} \sqcup \bar{t}$

⟨10⟩1. $\forall\, j > i : \bar{t}[j] \in \pi_2(\bar{o}'[j])$

⟨11⟩1. ASSUME: $\bar{t}[j] \in \pi_2(\bar{o}'[j])$ (induction hypothesis)

PROVE: $\bar{t}[j+1] \in \pi_2(\bar{o}'[j+1])$

$\langle 12\rangle 1$. $\bar{t}[j] \in \pi_2(\bar{o}'[j])$

PROOF: The induction hypothesis.

$\langle 12\rangle 2$. $\bar{t}[j+1] \in \pi_1(\bar{o}'[j+1]) \cup \pi_2(\bar{o}'[j+1])$

PROOF: $\langle 9\rangle 2$.

$\langle 12\rangle 3$. Choose $t \in \mathcal{H}$ such that $\bar{t}[j+1] \in \{\bar{t}[j]\} \succsim \{t\}$

PROOF: $\langle 9\rangle 3$.

$\langle 12\rangle 4$. Choose $o' \in O'$ such that $\bar{o}'[j+1] = \bar{o}'[j] \succsim o'$

PROOF: Definition (21) of *chains*.

$\langle 12\rangle 5$. $t \in \pi_1(o') \cup \pi_2(o')$

PROOF: $\langle 12\rangle 1$, $\langle 12\rangle 2$, $\langle 12\rangle 3$ and $\langle 12\rangle 4$.

$\langle 12\rangle 6$. $\pi_2(\bar{o}'[j]) \succsim (\pi_1(o') \cup \pi_2(o')) \subseteq \pi_2(\bar{o}'[j+1])$

PROOF: $\langle 12\rangle 4$ and definition (13) of $\succsim$.

$\langle 12\rangle 7$. $\bar{t}[j+1] \in \pi_2(\bar{o}'[j+1])$

PROOF: $\langle 12\rangle 1$, $\langle 12\rangle 3$, $\langle 12\rangle 5$ and $\langle 12\rangle 6$.

$\langle 12\rangle 8$. Q.E.D.

$\langle 11\rangle 2$. Q.E.D.

PROOF: Standard induction on $j$, base case proved by the assumption in $\langle 9\rangle 5$.

$\langle 10\rangle 2$. $\bar{t}[i \ldots \infty\rangle \in \mathit{negs}(\bar{o}')$ [7]

$\langle 11\rangle 1$. $\forall j \in \mathbb{N} : \bar{t}[i \ldots \infty\rangle[j] \in \pi_2(\bar{o}'[j+i-1])$

PROOF: $\forall j \in [i \ldots \infty\rangle : \bar{t}[j] \in \pi_2(\bar{o}'[j])$ by $\langle 9\rangle 5$ and $\langle 10\rangle 1$.

$\langle 11\rangle 2$. $\forall j \in \mathbb{N} : \exists t \in \mathcal{H} : \bar{t}[i \ldots \infty\rangle[j+1] \in \{\bar{t}[i \ldots \infty\rangle[j]\} \succsim \{t\}$

PROOF: $\langle 9\rangle 3$.

$\langle 11\rangle 3$. Q.E.D.

PROOF: $\langle 11\rangle 1$, $\langle 11\rangle 2$, and definition (23) of *negs*.

$\langle 10\rangle 3$. $\sqcup \bar{t} = \sqcup \bar{t}[i \ldots \infty\rangle$

$\langle 11\rangle 1$. $\forall l \in \mathcal{L} : \sqcup_l \bar{t} = \sqcup_l \bar{t}[i \ldots \infty\rangle$

PROOF: $\sqcup_l \bar{t}$ is the least upper bound.

$\langle 11\rangle 2$. Q.E.D.

PROOF: Definition (24) of $\sqcup \bar{t}$.

$\langle 10\rangle 4$. $\sqcup \bar{t} \subseteq \cup_{\bar{t} \in \mathit{negs}(\bar{o}')} \sqcup \bar{t}$

PROOF: $\langle 10\rangle 2$ and $\langle 10\rangle 3$.

$\langle 10\rangle 5$. Q.E.D.

$\langle 9\rangle 6$. Q.E.D.

PROOF: By $\langle 9\rangle 2$, the cases in $\langle 9\rangle 4$ and $\langle 9\rangle 5$ are exhaustive.

$\langle 8\rangle 2$. Q.E.D.

PROOF: $\forall$-rule.

$\langle 7\rangle 2$. Q.E.D.

PROOF: Definition of $\subseteq$.

$\langle 6\rangle 3$. Q.E.D.

[7] For a sequence $q$, $q[i \ldots \infty\rangle$ denotes the subsequence starting at index $i$, i.e. the sequence $q$ with the first $i-1$ elements removed.

PROOF: Definition (45) of $\rightsquigarrow_r$.

⟨5⟩3. Q.E.D.

⟨4⟩2. Q.E.D.

⟨3⟩2. Q.E.D.

PROOF: By definition (29) of $\mu_\infty$.

⟨2⟩5. Q.E.D.

PROOF: The cases are exhaustive, as $n$ is required to be a non-negative number or $\infty$.

⟨1⟩2. Q.E.D.

PROOF: $\forall$-rule.

□

### E.3 Monotonicity of $\rightsquigarrow_g$ with respect to the sequence diagram operators

**Theorem 10.** *Monotonicity of $\rightsquigarrow_g$ with respect to the* neg *operator*

ASSUME: 1. $d = \mathsf{neg}\ d_1$,
i.e. $[\![ d ]\!] = \{(\{\langle\rangle\}, p \cup n) \mid (p, n) \in [\![ d_1 ]\!]\}$ by definition (34).
2. $d_1 \rightsquigarrow_g d_1'$,
i.e. $[\![ d_1 ]\!] \rightsquigarrow_g [\![ d_1' ]\!]$ by definition (47),
i.e. $\forall\, o_1 \in [\![ d_1 ]\!] : \exists\, o_1' \in [\![ d_1' ]\!] : o_1 \rightsquigarrow_r o_1'$ by definition (46).
3. $d' = \mathsf{neg}\ d_1'$,
i.e. $[\![ d' ]\!] = \{(\{\langle\rangle\}, p \cup n) \mid (p, n) \in [\![ d_1' ]\!]\}$ by definition (34).

PROVE: $d \rightsquigarrow_g d'$,
i.e. $[\![ d ]\!] \rightsquigarrow_g [\![ d' ]\!]$ by definition (47),
i.e. $\forall\, o \in [\![ d ]\!] : \exists\, o' \in [\![ d' ]\!] : o \rightsquigarrow_r o'$ by definition (46).

$\langle 1\rangle 1$. $\exists\, o' \in [\![ d' ]\!] : o \rightsquigarrow_r o'$ for arbitrary $o = (p, n) \in [\![ d ]\!]$
  $\langle 2\rangle 1$. $p = \{\langle\rangle\}$
    PROOF: Assumption 1.
  $\langle 2\rangle 2$. Choose $(p_1, n_1) \in [\![ d_1 ]\!]$ such that $n = p_1 \cup n_1$
    PROOF: Assumption 1.
  $\langle 2\rangle 3$. Choose $(p_1', n_1') \in [\![ d_1' ]\!]$ such that $(p_1, n_1) \rightsquigarrow_r (p_1', n_1')$
    PROOF: Assumption 2.
  $\langle 2\rangle 4$. $o' = (p', n') = (\{\langle\rangle\}, p_1' \cup n_1') \in [\![ d' ]\!]$
    PROOF: Assumption 3.
  $\langle 2\rangle 5$. $(p, n) \rightsquigarrow_r (p', n')$
    $\langle 3\rangle 1$. Requirement 1: $n \subseteq n'$, i.e. $p_1 \cup n_1 \subseteq p_1' \cup n_1'$
      PROOF: $p_1 \subseteq p_1' \cup n_1'$ and $n_1 \subseteq n_1'$ by $\langle 2\rangle 3$ and definition (45) of $\rightsquigarrow_r$.
    $\langle 3\rangle 2$. Requirement 2: $p \subseteq p' \cup n'$
      PROOF: Trivial, since $p = p' = \{\langle\rangle\}$ by $\langle 2\rangle 1$ and $\langle 2\rangle 4$.
    $\langle 3\rangle 3$. Q.E.D.
      PROOF: Definition (45) of $\rightsquigarrow_r$.
  $\langle 2\rangle 6$. Q.E.D.
$\langle 1\rangle 2$. Q.E.D.
  PROOF: $\forall$-rule.

□

**Theorem 11.** *Monotonicity of $\rightsquigarrow_g$ with respect to the* alt *operator*

ASSUME: 1. $d = d_1\ \mathsf{alt}\ d_2$,
i.e. $[\![ d ]\!] = [\![ d_1 ]\!] \uplus [\![ d_2 ]\!]$ by definition (37).
2. $d_1 \rightsquigarrow_g d_1'$,
i.e. $[\![ d_1 ]\!] \rightsquigarrow_g [\![ d_1' ]\!]$ by definition (47).
3. $d_2 \rightsquigarrow_g d_2'$,
i.e. $[\![ d_2 ]\!] \rightsquigarrow_g [\![ d_2 ]\!]$ by definition (47).
4. $d' = d_1'\ \mathsf{alt}\ d_2$,
i.e. $[\![ d' ]\!] = [\![ d_1' ]\!] \uplus [\![ d_2' ]\!]$ by definition (37).

PROVE: $d \rightsquigarrow_g d'$,
i.e. $[\![ d ]\!] \rightsquigarrow_g [\![ d' ]\!]$ by definition (47).

PROOF: Lemma 36 with $O = [\![\ d\ ]\!]$, $O_1 = [\![\ d_1\ ]\!]$, $O_2 = [\![\ d_2\ ]\!]$, $O' = [\![\ d'\ ]\!]$, $O'_1 = [\![\ d'_1\ ]\!]$ and $O'_2 = [\![\ d'_2\ ]\!]$.

□

**Theorem 12.** *Monotonicity of $\rightsquigarrow_g$ with respect to the* xalt *operator*

ASSUME: 1. $d = d_1$ xalt $d_2$,
i.e. $[\![\ d\ ]\!] = [\![\ d_1\ ]\!] \cup [\![\ d_2\ ]\!]$ by definition (38).
2. $d_1 \rightsquigarrow_g d'_1$,
i.e. $[\![\ d_1\ ]\!] \rightsquigarrow_g [\![\ d'_1\ ]\!]$ by definition (47),
i.e. $\forall\, o_1 \in [\![\ d_1\ ]\!] : \exists\, o'_1 \in [\![\ d'_1\ ]\!] : o_1 \rightsquigarrow_r o'_1$ by definition (46).
3. $d_2 \rightsquigarrow_g d'_2$,
i.e. $[\![\ d_2\ ]\!] \rightsquigarrow_g [\![\ d'_2\ ]\!]$ by definition (47),
i.e. $\forall\, o_2 \in [\![\ d_2\ ]\!] : \exists\, o'_2 \in [\![\ d'_2\ ]\!] : o_2 \rightsquigarrow_r o'_2$ by definition (46).
4. $d' = d'_1$ xalt $d'_2$,
i.e. $[\![\ d'\ ]\!] = [\![\ d'_1\ ]\!] \cup [\![\ d'_2\ ]\!]$ by definition (38).

PROVE: $d \rightsquigarrow_g d'$,
i.e. $[\![\ d\ ]\!] \rightsquigarrow_g [\![\ d'\ ]\!]$ by definition (47),
i.e. $\forall\, o \in [\![\ d\ ]\!] : \exists\, o' \in [\![\ d'\ ]\!] : o \rightsquigarrow_r o'$ by definition (46).

⟨1⟩1. $\exists\, o' \in [\![\ d'\ ]\!] : o \rightsquigarrow_r o'$ for arbitrary $o \in [\![\ d\ ]\!]$
⟨2⟩1. CASE: $o \in [\![\ d_1\ ]\!]$
⟨3⟩1. Choose $o' \in [\![\ d'_1\ ]\!]$ such that $o \rightsquigarrow_r o'$
PROOF: Assumption 2.
⟨3⟩2. $o' \in [\![\ d'\ ]\!]$
PROOF: Assumption 4.
⟨3⟩3. Q.E.D.
⟨2⟩2. CASE: $o \in [\![\ d_2\ ]\!]$
⟨3⟩1. Choose $o' \in [\![\ d'_2\ ]\!]$ such that $o \rightsquigarrow_r o'$
PROOF: Assumption 3.
⟨3⟩2. $o' \in [\![\ d'\ ]\!]$
PROOF: Assumption 4.
⟨3⟩3. Q.E.D.
⟨2⟩3. Q.E.D.
PROOF: By ⟨2⟩1, ⟨2⟩2, assumption 1 and definition of $\cup$.
⟨1⟩2. Q.E.D.
PROOF: $\forall$-rule.

□

**Theorem 13.** *Monotonicity of $\rightsquigarrow_g$ with respect to the* seq *operator*

ASSUME: 1. $d =$ seq $[d_1, \ldots, d_n]$
2. $\forall\, i \in [1 \ldots n] : d_i \rightsquigarrow_g d'_i$,
i.e. $\forall\, i \in [1 \ldots n] : [\![\ d_i\ ]\!] \rightsquigarrow_g [\![\ d'_i\ ]\!]$ by definition (47).
3. $d' =$ seq $[d'_1, \ldots, d'_n]$

PROVE: $d \rightsquigarrow_g d'$,
i.e. $[\![\ d\ ]\!] \rightsquigarrow_g [\![\ d'\ ]\!]$ by definition (47).

⟨1⟩1. CASE: $n = 1$

⟨2⟩1. $[\![\ d\ ]\!] = [\![\ d_1\ ]\!]$
PROOF: Assumption 1, ⟨1⟩1 and definition (36) of $\mathsf{seq}$.
⟨2⟩2. $[\![\ d'\ ]\!] = [\![\ d'_1\ ]\!]$
PROOF: Assumption 3, ⟨1⟩1 and definition (36) of $\mathsf{seq}$.
⟨2⟩3. $[\![\ d_1\ ]\!] \rightsquigarrow_g [\![\ d'_1\ ]\!]$
PROOF: Assumption 2.
⟨2⟩4. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩2 and ⟨2⟩3.
⟨1⟩2. CASE: $n > 1$
⟨2⟩1. ASSUME: $[\![\ \mathsf{seq}\ [d_1, \ldots d_{n-1}]\ ]\!] \rightsquigarrow_g [\![\ \mathsf{seq}\ [d'_1, \ldots d'_{n-1}]\ ]\!]$
(induction hypothesis)
PROVE: $[\![\ \mathsf{seq}\ [d_1, \ldots d_n]\ ]\!] \rightsquigarrow_g [\![\ \mathsf{seq}\ [d'_1, \ldots d'_n]\ ]\!]$
⟨3⟩1. $[\![\ \mathsf{seq}\ [d_1, \ldots d_n]\ ]\!] = [\![\ \mathsf{seq}\ [d_1, \ldots d_{n-1}]\ ]\!] \succsim [\![\ d_n\ ]\!]$
PROOF: Definition (36) of $\mathsf{seq}$.
⟨3⟩2. $[\![\ \mathsf{seq}\ [d'_1, \ldots d'_n]\ ]\!] = [\![\ \mathsf{seq}\ [d'_1, \ldots d'_{n-1}]\ ]\!] \succsim [\![\ d'_n\ ]\!]$
PROOF: Definition (36) of $\mathsf{seq}$.
⟨3⟩3. $[\![\ \mathsf{seq}\ [d_1, \ldots d_{n-1}]\ ]\!] \rightsquigarrow_g [\![\ \mathsf{seq}\ [d'_1, \ldots d'_{n-1}]\ ]\!]$
PROOF: The induction hypothesis.
⟨3⟩4. $[\![\ d_n\ ]\!] \rightsquigarrow_g [\![\ d'_n\ ]\!]$
PROOF: Assumption 2.
⟨3⟩5. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩2, ⟨3⟩3, ⟨3⟩4 and lemma 33 with $O = [\![\ d\ ]\!]$, $O_1 = [\![\ \mathsf{seq}\ [d_1, \ldots d_{n-1}]\ ]\!]$, $O_2 = [\![\ d_n\ ]\!]$, $O' = [\![\ d'\ ]\!]$, $O'_1 = [\![\ \mathsf{seq}\ [d'_1, \ldots d'_{n-1}]\ ]\!]$ and $O'_2 = [\![\ d'_n\ ]\!]$.
⟨2⟩2. Q.E.D.
PROOF: Assumptions 1 and 3 and standard induction on $n$, base case proved by ⟨1⟩1.
⟨1⟩3. Q.E.D.
PROOF: The cases are exhaustive.

□

**Theorem 14.** *Monotonicity of $\rightsquigarrow_g$ with respect to the* $\mathsf{par}$ *operator*
ASSUME: 1. $d = d_1\ \mathsf{par}\ d_2$,
i.e. $[\![\ d\ ]\!] = [\![\ d_1\ ]\!] \parallel [\![\ d_2\ ]\!]$ by definition (39).
2. $d_1 \rightsquigarrow_g d'_1$,
i.e. $[\![\ d_1\ ]\!] \rightsquigarrow_g [\![\ d'_1\ ]\!]$ by definition (47).
3. $d_2 \rightsquigarrow_g d'_2$,
i.e. $[\![\ d_2\ ]\!] \rightsquigarrow_g [\![\ d'_2\ ]\!]$ by definition (47).
4. $d' = d'_1\ \mathsf{par}\ d'_2$,
i.e. $[\![\ d'\ ]\!] = [\![\ d'_1\ ]\!] \parallel [\![\ d'_2\ ]\!]$ by definition (39).
PROVE: $d \rightsquigarrow_g d'$,
i.e. $[\![\ d\ ]\!] \rightsquigarrow_g [\![\ d'\ ]\!]$ by definition (47).

PROOF: Lemma 34 with $O = [\![\ d\ ]\!]$, $O_1 = [\![\ d_1\ ]\!]$, $O_2 = [\![\ d_2\ ]\!]$, $O' = [\![\ d'\ ]\!]$, $O'_1 = [\![\ d'_1\ ]\!]$ and $O'_2 = [\![\ d'_2\ ]\!]$.

□

**Theorem 15.** *Monotonicity of $\rightsquigarrow_g$ with respect to the* tc *operator*

ASSUME: 1. $d = d_1 \,\mathsf{tc}\, C$,
i.e. $[\![\ d\ ]\!] = [\![\ d_1\ ]\!] \wr C$ by definition (40).
2. $d_1 \rightsquigarrow_g d_1'$,
i.e. $[\![\ d_1\ ]\!] \rightsquigarrow_g [\![\ d_1'\ ]\!]$ by definition (47).
3. $d' = d_1' \,\mathsf{tc}\, C$,
i.e. $[\![\ d'\ ]\!] = [\![\ d_1'\ ]\!] \wr C$ by definition (40).

PROVE: $d \rightsquigarrow_g d'$,
i.e. $[\![\ d\ ]\!] \rightsquigarrow_g [\![\ d'\ ]\!]$ by definition (47).

PROOF: Lemma 35 with $O = [\![\ d\ ]\!]$, $O_1 = [\![\ d_1\ ]\!]$, $O' = [\![\ d'\ ]\!]$ and $O_1' = [\![\ d_1'\ ]\!]$.

□

**Theorem 16.** *Monotonicity of $\rightsquigarrow_g$ with respect to the* loop *operator*

ASSUME: 1. $d = \mathsf{loop}\ I\ d_1$,
i.e. $[\![\ d\ ]\!] = \biguplus_{i \in I} \mu_i\ [\![\ d_1\ ]\!]$ by definition (41).
2. $d_1 \rightsquigarrow_g d_1'$,
i.e. $[\![\ d_1\ ]\!] \rightsquigarrow_g [\![\ d_1'\ ]\!]$ by definition (47).
3. $d' = \mathsf{loop}\ I\ d_1'$,
i.e. $[\![\ d'\ ]\!] = \biguplus_{i \in I} \mu_i\ [\![\ d_1'\ ]\!]$ by definition (41).

PROVE: $d \rightsquigarrow_g d'$,
i.e. $[\![\ d\ ]\!] \rightsquigarrow_g [\![\ d'\ ]\!]$ by definition (47).

⟨1⟩1. $\forall\, i \in I : \mu_i\ [\![\ d_1\ ]\!] \rightsquigarrow_g \mu_i\ [\![\ d_1'\ ]\!]$
PROOF: Assumption 2 and lemma 38 (monotonicity of $\rightsquigarrow_g$ with respect to $\mu_n$) with $O = [\![\ d_1\ ]\!]$ and $O' = [\![\ d_1'\ ]\!]$.

⟨1⟩2. Q.E.D.
PROOF: ⟨1⟩1, lemma 37 (monotonicity of $\rightsquigarrow_g$ with respect to $\biguplus$) with $O_i = \mu_i[\![\ d_1\ ]\!]$ and $O_i' = \mu_i[\![\ d_1'\ ]\!]$.

□

**Theorem 17.** *Monotonicity of $\rightsquigarrow_g$, restricted to narrowing, with respect to the* assert *operator*

ASSUME: 1. $d =$ assert $d_1$,
i.e. $[\![\ d\ ]\!] = \{(p_1, n_1 \cup (\mathcal{H} \setminus p_1)) \mid (p_1, n_1) \in [\![\ d_1\ ]\!]\}$ by definition (35).
2. $d_1 \rightsquigarrow_{g,n} d_1'$,
i.e. $[\![\ d_1\ ]\!] \rightsquigarrow_{g,n} [\![\ d_1'\ ]\!]$ by definition (47),
i.e. $\forall\, o_1 \in [\![\ d_1\ ]\!] : \exists\, o_1' \in [\![\ d_1'\ ]\!] : o_1 \rightsquigarrow_n o_1'$ by definition (46) restricted to narrowing.
3. $d' =$ assert $d_1'$, i.e. $[\![\ d'\ ]\!] = \{(p_1', n_1' \cup (\mathcal{H} \setminus p_1')) \mid (p_1', n_1') \in [\![\ d_1'\ ]\!]\}$ by definition (35).

PROVE: $d \rightsquigarrow_{g,n} d'$,
i.e. $[\![\ d\ ]\!] \rightsquigarrow_{g,n} [\![\ d'\ ]\!]$ by definition (47),
i.e. $\forall\, o \in [\![\ d\ ]\!] : \exists\, o' \in [\![\ d'\ ]\!] : o \rightsquigarrow_n o'$ by definition (46) restricted to narrowing.

⟨1⟩1. $\exists\, o' \in [\![\ d'\ ]\!] : o \rightsquigarrow_n o'$ for arbitrary $o = (p, n) \in [\![\ d\ ]\!]$

⟨2⟩1. Choose $(p_1, n_1) \in [\![\ d_1\ ]\!]$ such that $p = p_1$ and $n = n_1 \cup (\mathcal{H} \setminus p_1)$
PROOF: Assumption 1.
⟨2⟩2. Choose $(p'_1, n'_1) \in [\![\ d'_1\ ]\!]$ such that $(p_1, n_1) \rightsquigarrow_n (p'_1, n'_1)$
PROOF: Assumption 2.
⟨2⟩3. $o' = (p', n') = (p'_1, n'_1 \cup (\mathcal{H} \setminus p'_1)) \in [\![\ d'\ ]\!]$
PROOF: Assumption 3.
⟨2⟩4. $(p, n) \rightsquigarrow_n (p', n')$
⟨3⟩1. Requirement 1: $p' \subset p$, i.e. $p'_1 \subset p_1$
PROOF: $p'_1 \subset p_1$ by ⟨2⟩2 and definition (49) of $\rightsquigarrow_n$.
⟨3⟩2. Requirement 2: $n' = n \cup (p \setminus p')$,
i.e. $n'_1 \cup (\mathcal{H} \setminus p'_1) = (n_1 \cup (\mathcal{H} \setminus p_1)) \cup (p_1 \setminus p'_1)$
⟨4⟩1. $n'_1 = n_1 \cup (p_1 \setminus p'_1)$
PROOF: ⟨2⟩2 and definition (49) of $\rightsquigarrow_n$.
⟨4⟩2. $(\mathcal{H} \setminus p'_1) = (\mathcal{H} \setminus p_1) \cup (p_1 \setminus p'_1)$
⟨5⟩1. $\mathcal{H} \setminus p_1 \subset \mathcal{H} \setminus p'_1$
PROOF: $p'_1 \subset p_1$ by ⟨2⟩2 and definition (49) of $\rightsquigarrow_n$.
⟨5⟩2. $(\mathcal{H} \setminus p'_1) = (\mathcal{H} \setminus p_1) \cup ((\mathcal{H} \setminus p'_1) \setminus (\mathcal{H} \setminus p_1))$
PROOF: ⟨5⟩1 and the general lemma $A \subset B \Rightarrow B = A \cup (B \setminus A)$.
⟨5⟩3. $(\mathcal{H} \setminus p'_1) \setminus (\mathcal{H} \setminus p_1) = p_1 \setminus p'_1$
PROOF: $(\mathcal{H} \setminus a) \setminus (\mathcal{H} \setminus b) = b \setminus a$ by the general lemma illustrated in figure E.3.
⟨5⟩4. Q.E.D.
PROOF: ⟨5⟩2 and $(\mathcal{H} \setminus p_1) \cup ((\mathcal{H} \setminus p'_1) \setminus (\mathcal{H} \setminus p_1)) = (\mathcal{H} \setminus p_1) \cup (p_1 \setminus p'_1)$ by ⟨5⟩3.
⟨4⟩3. Q.E.D.
PROOF: ⟨4⟩1, ⟨4⟩2 and associativity of $\cup$.
⟨3⟩3. Q.E.D.
PROOF: Definition (49) of $\rightsquigarrow_n$.
⟨2⟩5. Q.E.D.
⟨1⟩2. Q.E.D.
PROOF: $\forall$-rule.

□

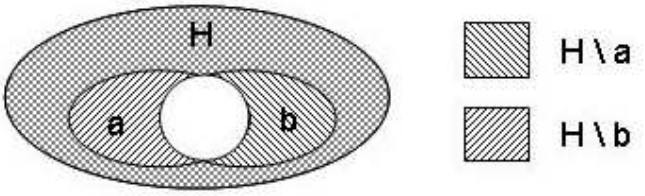


**Fig. 15.** Lemma: $(H \setminus a) \setminus (H \setminus b) = b \setminus a$