University of Oslo
Department of Informatics

Relating computer systems to sequence diagrams with underspecification, inherent nondeterminism and probabilistic choice

Part 1

Ragnhild Kobro Runde, Atle Refsdal, Ketil Stølen

Research Report 346
ISBN 82-7368-303-6
ISSN 0806-3036

January 2007

Relating computer systems to sequence diagrams with underspecification, inherent nondeterminism and probabilistic choice

Part 1: underspecification and inherent nondeterminism

Ragnhild Kobro Runde[1], Atle Refsdal[1,2], and Ketil Stølen[1,2]

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

Abstract. Having a sequence diagram specification and a computer system, we need to answer the question: *Is the system compliant with the sequence diagram specification in the desired way?* We present a procedure for answering this question for three variations of sequence diagrams. The procedure is independent of the choice of programming language used for the system. The semantics of sequence diagrams is denotational and based on traces. In order to answer the initial question, the procedure starts by obtaining the trace-set of the system by e.g. testing, and then transforming this into the same semantic model as that used for the sequence diagram. In addition to extending our earlier work on refinement relations for sequence diagrams, we define conformance relations relating systems to sequence diagrams.
The work is split in two parts. This paper presents part 1, in which we introduce the necessary definitions for using the compliance checking procedure on sequence diagrams with underspecification and sequence diagrams with inherent nondeterminism. In part 2 [RRS07], we present the definitions for using the procedure on sequence diagrams with probabilistic choice.

1 Introduction

Having a sequence diagram specification and a computer system, we need to answer the question: *Is the system compliant with the specification in the desired way?*

Sequence diagrams are widely used for specifying computer systems within a broad range of application domains. They are used for different methodological purposes including requirements capture, illustrating example runs, test scenario specification and risk scenario documentation. Although sequence diagrams are widely used in practice, their relationship to real computer systems is nevertheless surprisingly unclear. This is partly caused by the fact that sequence diagrams are used for different purposes, but even more so because in contrast to

most other techniques for specifying dynamic behaviour they give only a partial view.

Answering the initial question above requires an understanding of what is meant by a computer system and to what extent such a system is different from a sequence diagram. Obviously, we need a formal model for computer systems. Also, the answer clearly depends on the expressiveness of the sequence diagram dialect we are using. In this paper we study the problem with respect to two different variations of sequence diagrams, as formally defined in the denotational trace semantics of STAIRS [HHRS05]. The two variations are sequence diagrams with underspecification and sequence diagrams with inherent nondeterminism.

The notion of compliance is closely related to that of refinement. Refinement is a way of relating different specifications of the same system, where the idea is that a refinement should be a more detailed description containing all the constraints given by the original specification, in addition to some new ones.[1] Different development stages may require different notions of refinement. The final specification used when implementing the system, is the result of several successive refinement steps. The system should be compliant not only with the final specification, but also with all specifications in the chain of refinements. Consequently, we may need several notions of compliance corresponding to the various notions of refinement.

In this paper we only consider compliance for sequence diagrams without external input and output. For such sequence diagrams, we propose the following compliance checking procedure:

1. Given a computer system I and a sequence diagram d, use e.g. testing on I to obtain the trace-set describing its behaviour.
2. Transform this trace-set into the same semantic model as that used for d.
3. Depending on the kind of compliance desired, select the appropriate compliance relation.
4. I is compliant with d if this compliance relation holds between the semantics of d and the representation of I obtained in step 2.

This paper is organized as follows: In Section 2 we state the requirements that a step-wise procedure for checking computer systems against sequence diagrams needs to fulfil. Section 3 gives a general introduction to sequence diagrams and their denotational trace semantics. Sections 4 and 5 introduce sequence diagrams with underspecification and inherent nondeterminism, respectively, and define what it means for a system to be compliant with such sequence diagrams. In Section 6 we present theoretical results related to the definitions of refinement and compliance. We discuss related work in Section 7, before concluding in Section 8. Appendices A and B give a detailed overview of the theoretical results, together with the necessary proofs.

[1] Note that we use the term "constraint" rather loosely. For instance, the addition of a new constraint may result in the specification requiring more behaviours of the system.

2 Requirements

In order to motivate the following discussion and formal definitions, we formulate a number of requirements that our procedure has been designed to fulfil. That these requirements are met, are demonstrated throughout the discussion and summed up in Section 8.

1. The procedure should be independent of the choice of programming language in which the system is implemented. A sequence diagram does not prescribe any particular programming language, and the procedure should be sufficiently general to capture all possible choices. In general, we cannot assume that we have access to the source code of the system. This means that the only knowledge about the system that may be used by the procedure, is what can be obtained by testing. Although not feasible in practice, we assume that we are able to observe infinite runs. Otherwise, we would have to restrict ourselves to safety properties.
2. The notion of compliance should be a special case of refinement. Given a sequence diagram and its refinement, the procedure should give that a system is compliant with the refinement only if the system is also compliant with the original sequence diagram.
3. There should be a natural correspondence between the compliance relations for the two variations of sequence diagrams. The language of sequence diagrams without inherent nondeterminism is a subclass of the language that also allows inherent nondeterminism. This means that the general compliance relation for sequence diagrams with inherent nondeterminism should at least capture everything allowed by the general compliance relation for sequence diagrams containing only underspecification.
4. The procedure should be faithful to the underlying ideas and principles of UML 2.1 [OMG06] sequence diagrams. UML is the leading specification language within the software industry of today, and our goal is that our approach should be of help for UML practitioners.

3 Sequence diagrams and trace semantics

This section gives necessary background for the following sections. It provides a general introduction to sequence diagrams and their denotational trace semantics. In this section we consider only three operators on sequence diagrams, namely the operators for refusal, sequential and parallel composition. The following two sections extend this basic set of operators with operators for underspecification and inherent nondeterminism, in each case focusing on how to determine whether a given system is in compliance with such a sequence diagram.

We use the simple sequence diagram in Figure 1 to introduce some terminology. S is the name of the sequence diagram, A and B are lifelines (corresponding to e.g. components or objects), while m is a message from A to B. We say that the diagram includes two events, the sending of m (denoted !m) and the reception of m (denoted ?m).

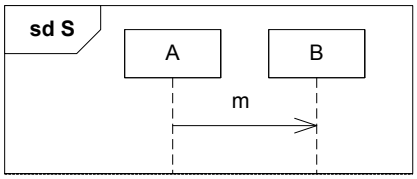

Fig. 1. Simple sequence diagram

To assign precise meaning to a sequence diagram, we use denotational trace semantics as defined in STAIRS [HHRS05]. This formal semantics is compliant with the semi-formal descriptions given in the UML 2.1 standard [OMG06]. A trace is a sequence of events representing a system run. An event is a pair (k, m) consisting of a kind k (either ! or ?) and a message m. A message is a triple (s, tr, re) consisting of a signal s, a transmitter lifeline tr and a receiver lifeline re.

For a trace to be well-formed, we require that for all messages:

- if both the sender and receiver lifeline are present in the diagram, then both the send and the receive event are present in the trace;
- the send event is ordered before the corresponding receive event if both events are present in the trace.

The semantics of a sequence diagram d is denoted $[\![d]\!]$. In the basic case, the semantics of a sequence diagram is an *interaction obligation* (p, n) where p is a set of positive (i.e. valid) traces and n is a set of negative (i.e. invalid) traces. Traces not in the diagram are called inconclusive, and may be introduced as positive or negative by later refinement steps. Letting $\mathcal{H}$ denote the universe of all well-formed traces, the traces $\mathcal{H} \setminus (p \cup n)$ are inconclusive in the interaction obligation (p, n).

Parallel composition ($\parallel$) of two trace sets corresponds to point-wise interleaving of their individual traces. The ordering of events within each trace is maintained in the result. For sequential composition ($\gtrsim$) we require in addition that for events on the same lifeline, all events from the first trace is ordered before the events from the second trace. Formally:

$$
\begin{aligned}
s_1 \parallel s_2 \quad &\stackrel{\mathsf{def}}{=} \quad \{h \in \mathcal{H} \mid \exists p \in \{1,2\}^\infty : \\
&\qquad \pi_2((\{1\} \times \mathcal{E}) \circledcirc (p, h)) \in s_1 \wedge \qquad (1) \\
&\qquad \pi_2((\{2\} \times \mathcal{E}) \circledcirc (p, h)) \in s_2\} \\
s_1 \gtrsim s_2 \quad &\stackrel{\mathsf{def}}{=} \quad \{h \in \mathcal{H} \mid \exists h_1 \in s_1, h_2 \in s_2 : \forall l \in \mathcal{L} : \qquad (2) \\
&\qquad e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2\}
\end{aligned}
$$

where $\mathcal{E}$ and $\mathcal{L}$ are the sets of all events and lifelines, respectively; $e.l$ is the set of events that may take place on the lifeline l; π_2 is a projection operator returning

the second element of a pair; and $\frown$ is the concatenation operator for sequences. $\circledS$ and $\circledT$ are filtering operators for traces and pairs of traces, respectively. $E \circledS h$ is the trace obtained from the trace h by removing from h all events that is not in the set of events E. For instance, we have that

$$\{e_1, e_3\} \circledS \langle e_1, e_1, e_2, e_1, e_3, e_2\rangle = \langle e_1, e_1, e_1, e_3\rangle$$

The operator $\circledT$ is a generalization of $\circledS$ filtering pairs of traces with respect to pairs of elements such that for instance

$$\begin{aligned}\{(1, e_1), (1, e_2)\} \circledS (\langle 1, 1, 2, 1, 2\rangle, \langle e_1, e_1, e_1, e_2, e_2\rangle)\\ = (\langle 1, 1, 1\rangle, \langle e_1, e_1, e_2\rangle)\end{aligned}$$

For formal definitions of $\circledS$ and $\circledT$, see [BS01].

For interaction obligations, parallel composition ($\parallel$), sequential composition ($\gtrsim$) and refusal ($\dagger$) are defined by:

$$(p_1, n_1) \parallel (p_2, n_2) \stackrel{\mathsf{def}}{=} (p_1 \parallel p_2, (n_1 \parallel p_2) \cup (n_1 \parallel n_2) \cup (p_1 \parallel n_2)) \tag{3}$$

$$(p_1, n_1) \gtrsim (p_2, n_2) \stackrel{\mathsf{def}}{=} (p_1 \gtrsim p_2, (n_1 \gtrsim p_2) \cup (n_1 \gtrsim n_2) \cup (p_1 \gtrsim n_2)) \tag{4}$$

$$\dagger(p_1, n_1) \stackrel{\mathsf{def}}{=} (\emptyset, p_1 \cup n_1) \tag{5}$$

Notice that composing a positive and a negative trace always yields a negative trace, while the result of composing an inconclusive trace with a positive or negative trace is always inconclusive.

Finally, the sequence diagram operators for parallel composition (par), sequential composition (seq) and negative behaviour (refuse) are defined by:

$$[\![\ d_1 \ \mathsf{par}\ d_2\ ]\!] \stackrel{\mathsf{def}}{=} [\![\ d_1\ ]\!] \parallel [\![\ d_2\ ]\!] \tag{6}$$

$$[\![\ d_1 \ \mathsf{seq}\ d_2\ ]\!] \stackrel{\mathsf{def}}{=} [\![\ d_1\ ]\!] \gtrsim [\![\ d_2\ ]\!] \tag{7}$$

$$[\![\ \mathsf{refuse}\ d_1\ ]\!] \stackrel{\mathsf{def}}{=} \dagger[\![\ d_1\ ]\!] \tag{8}$$

4 Relating computer systems to sequence diagrams with underspecification

When writing specifications, it is often useful to leave certain aspects of the system behaviour open. This is known as underspecification. Typically, underspecification is a consequence of abstraction and a desire to focus on the essential behaviour of the system. In sequence diagrams, underspecification may be the result of weak sequencing or specified using the operator alt, describing alternative behaviours that the system *may* exhibit. Underspecification may be removed either by later development steps (refinements) or during the implementation process.

Underspecification in the sense of alt corresponds to taking the pair-wise union of the positive and negative trace-sets of the operands. Formally:

$$[\![\ d_1 \ \mathsf{alt}\ d_2\ ]\!] \stackrel{\mathsf{def}}{=} [\![\ d_1\ ]\!] \uplus [\![\ d_2\ ]\!] \tag{9}$$

where inner union ($\uplus$) on interaction obligations is defined by:

$$(p_1, n_1) \uplus (p_2, n_2) \stackrel{\mathsf{def}}{=} (p_1 \cup p_2, n_1 \cup n_2) \tag{10}$$

4.1 Refinement

Refinement means to add more information to the specification in order to bring it closer to a real system. An important requirement is that any valid system that is compliant with the refinement should also be compliant with the original specification. In addition to the basic refinement relation defined in [HHRS05], we define another relation called restricted refinement. The idea is that the two refinement relations will be used in different phases of the development process.

Refinement As sequence diagrams are incomplete specifications describing only parts of the system behaviour, a refinement step may add more positive and/or negative behaviours to the specification, hence reducing the set of inconclusive traces. Also, a refinement step may reduce underspecification, i.e. redefine positive traces as negative. Negative traces always remain negative. Formally, refinement of interaction obligations is defined by:

$$(p, n) \rightsquigarrow_r (p', n') \stackrel{\mathsf{def}}{=} n \subseteq n' \wedge p \subseteq p' \cup n' \tag{11}$$

As can be seen from the definition, a refinement may legally redefine all of the original positive traces as negative. Having this possibility may be important in an early development phase focusing on exploring the desired system requirements.

Restricted refinement At some stage during the development process it may be natural to fix the set of positive traces, with the intention that at least one of these should be present in any system compliant with this sequence diagram. After that, valid refinement steps may only redefine positive and inconclusive traces as negative, in order to remove underspecification and decide on the exact set of traces that may be produced by the final system. Extending the set of positive traces is no longer allowed:

$$(p, n) \rightsquigarrow_{rr} (p', n') \stackrel{\mathsf{def}}{=} (p, n) \rightsquigarrow_r (p', n') \wedge p' \subseteq p \tag{12}$$

4.2 Compliance

As explained in Section 2, we assume that all we know about a computer system is its set of traces. The traces are assumed to be well-formed in the sense explained in Section 3. This allows us to reason independently of any particular programming language, and also to handle applications where different components may be written in different languages.

In order to check a system I represented by its set of traces against a sequence diagram specification d using the semantic model outlined in Section 3, we first transform I into an interaction obligation $\langle I \rangle_d$:

$$\langle I \rangle_d \stackrel{\mathsf{def}}{=} (traces(I), \mathcal{H}^{ll(d)} \setminus traces(I)) \tag{13}$$

where $\mathcal{H}^{ll(d)}$ is the set of all well-formed traces consisting only of events taking place on the lifelines in the sequence diagram d, denoted $ll(d)$.

A general refinement principle in STAIRS is that traces described as positive or negative in the original specification cannot become inconclusive by a refinement step, as this would mean deviation from earlier given constraints. Since compliance should be a special case of refinement, $\langle I \rangle_d$ must include (as positive or negative) at least all traces described by d. Definition (13) ensures this by the use of $\mathcal{H}^{ll(d)}$. Employing $\mathcal{H}^{ll(d)}$ and not $\mathcal{H}$, guarantees consistency when performing parallel composition of two systems with disjoint sets of lifelines.

Corresponding to the two refinement relations in Section 4.1, we then define two different compliance relations.

Compliance relation A system I complies to a sequence diagram d if the semantics we get by using definition (13) is a refinement of $[\![d]\!]$. Formally:

$$[\![d]\!] \mapsto_r \langle I \rangle_d \stackrel{\mathsf{def}}{=} [\![d]\!] \rightsquigarrow_r \langle I \rangle_d \tag{14}$$

Restricted compliance relation A sequence diagram specification gives a global view of the system behaviour, but is often implemented as a number of components. As each of these components only has a local view of the overall system behaviour, their independent behaviours may combine in unexpected ways resulting in so-called implied scenarios [AEY00], i.e. traces that are inconclusive in the sequence diagram specification.
With restricted compliance, the system should contain at least one of the positive traces from the sequence diagram. Because of implied scenarios, we do not require that all possible system traces are explicitly described as positive in the sequence diagram. For a system I, we instead require that $traces(I)$ and the positive traces of the sequence diagram have at least one trace in common. In addition, $traces(I)$ may contain arbitrary many of the positive and inconclusive traces from the sequence diagram. Formally:

$$[\![d]\!] \mapsto_{rr} \langle I \rangle_d \stackrel{\mathsf{def}}{=} [\![d]\!] \mapsto_r \langle I \rangle_d \ \wedge \ \pi_1([\![d]\!]) \cap \pi_1(\langle I \rangle_d) \neq \emptyset \tag{15}$$

where π_1 is a projection operator returning the first element of a pair, in this case the positive traces of d and $\langle I \rangle_d$ (i.e. $traces(I)$), respectively.

4.3 Example

As a simple example, consider the specification of a gambling machine in Figure 2. First, the machine receives either a dime or a quarter. As a result, the

machine either sends the message "You won" together with a dollar, or the message "You lost".[2] The veto operator is a high-level operator defined by:

$$\mathsf{veto}\ d \stackrel{\mathsf{def}}{=} \mathsf{skip}\ \mathsf{alt}\ \mathsf{refuse}\ d \tag{16}$$

where skip is the empty diagram defined by:

$$[\![\ \mathsf{skip}\ ]\!] \stackrel{\mathsf{def}}{=} (\{\langle\rangle\}, \emptyset) \tag{17}$$

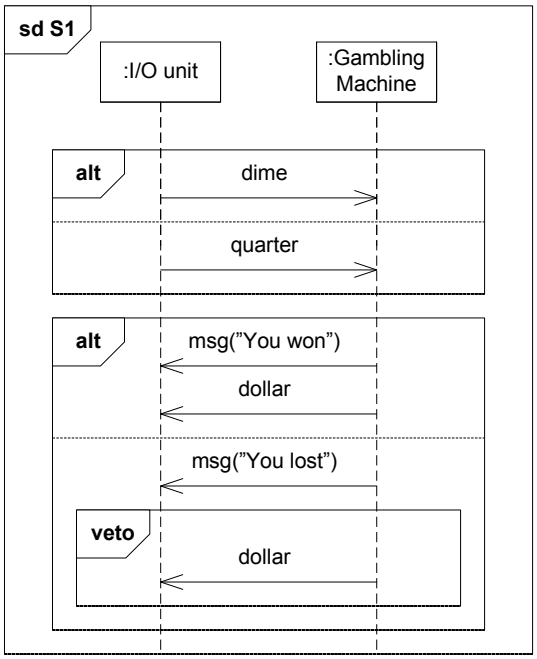

Fig. 2. Sequence diagram with underspecification (alt)

In this example, veto is used to specify that the message "You lost" should *not* be followed by a dollar. Shortening each message, the semantics $[\![\ S_1\ ]\!]$ of this sequence diagram is:

$$\begin{array}{l}
(\ \{\langle !di, ?di, !m(yw), ?m(yw), !do, ?do\rangle, \langle !qu, ?qu, !m(yw), ?m(yw), !do, ?do\rangle,\\
\quad \langle !di, ?di, !m(yw), !do, ?m(yw), ?do\rangle, \langle !qu, ?qu, !m(yw), !do, ?m(yw), ?do\rangle,\\
\quad \langle !di, ?di, !m(yl), ?m(yl)\rangle, \langle !qu, ?qu, !m(yl), ?m(yl)\rangle\}\ ,\\
\quad \{\langle !di, ?di, !m(yl), ?m(yl), !do, ?do\rangle, \langle !qu, ?qu, !m(yl), ?m(yl), !do, ?do\rangle,\\
\quad \langle !di, ?di, !m(yl), !do, ?m(yl), ?do\rangle, \langle !qu, ?qu, !m(yl), !do, ?m(yl), ?do\rangle\}\ )
\end{array}$$

[2] As we will come back to in Section 5, alt is not the best operator to use between these two last alternatives.

A possible way to implement this sequence diagram would be to build a system I_1 where the gambling machine receives a dime, after which it responds with a "You lost" message and then nothing more happens. This may be represented by the trace-set $traces(I_1) = \{\langle !di, ?di, !m(yl), ?m(yl)\rangle\}$, which gives $\langle I_1\rangle_{S_1} = (\{\langle !di, ?di, !m(yl), ?m(yl)\rangle\}, \mathcal{H}^{ll(S_1)} \setminus \{\langle !di, ?di, !m(yl), ?m(yl)\rangle\})$. It is straightforward to see that I_1 is in compliance with the sequence diagram S_1 according to both definitions (14) and (15), as the trace $\langle !di, ?di, !m(yl), ?m(yl)\rangle$ is positive in $[\![S_1]\!]$, and the negative traces of $[\![S_1]\!]$ are also negative in $\langle I_1\rangle_{S_1}$.

5 Relating computer systems to sequence diagrams with inherent nondeterminism

Using only underspecification, a system may be in compliance with the sequence diagram even if it contains one only of the positive traces and nothing else. In many cases this is not sufficient. One example is the gambling machine from Section 4.3, where the sequence diagram allowed a system where the only possible outcome was the user losing his money. A more realistic specification would be to require that both winning and losing should be possible outcomes. Also, the choice between winning and losing should be performed nondeterministically (or at least appear so to the user of the gambling machine).

For specifying inherent nondeterminism, or alternatives that must all be reflected in the specified system, the operator xalt (first introduced in [HS03]) may be used. To distinguish between underspecification and inherent nondeterminism also at the semantic level, the semantics of a sequence diagram d is no longer a single interaction obligation as in Sections 3 and 4, but instead a *set* of m interaction obligations $[\![d]\!] = \{(p_1, n_1), (p_2, n_2), \ldots, (p_m, n_m)\}$ for some natural number m. The idea is that each interaction obligation gives a requirement that must be fulfilled by any system in compliance with it. Formally, the xalt operator is defined by:

$$[\![d_1 \ \mathsf{xalt} \ d_2]\!] \stackrel{\mathsf{def}}{=} [\![d_1]\!] \cup [\![d_2]\!] \tag{18}$$

Hence, the composition of d_1 and d_2 by xalt requires all the inherent nondeterminism specified by d_1 in addition to all the inherent nondeterminism specified by d_2.

With this extended semantic model, we also need definitions for parallel composition ($\|$), sequential composition ($\gtrsim$), underspecification ($\uplus$) and refusal ($\dagger$) on sets of interaction obligations:

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

$$\dagger O_1 \stackrel{\mathsf{def}}{=} \{\dagger o_1 \mid o_1 \in O_1\} \tag{20}$$

where op is one of $\|$, $\gtrsim$ or $\uplus$.

5.1 Refinement

For a specification with both inherent nondeterminism and underspecification, we distinguish between four different refinement relations. The most general notion is general refinement, which is typically used initially, while the most specific notion is restricted limited refinement, which is more likely to be used near the end of the development process.

General and restricted general refinement For a specification having several interaction obligations as its semantics, we require that each one should be refined by at least one interaction obligation in any valid refinement. The only difference between general ($\rightsquigarrow_g$) and restricted general ($\rightsquigarrow_{rg}$) refinement, is with respect to the refinement definition used between the two interaction obligations:

$$[\![\ d\ ]\!] \rightsquigarrow_{(r)g} [\![\ d'\ ]\!] \stackrel{\mathsf{def}}{=} \forall o \in [\![\ d\ ]\!] : \exists o' \in [\![\ d'\ ]\!] : o \rightsquigarrow_{(r)r} o' \tag{21}$$

where $\rightsquigarrow_r$ and $\rightsquigarrow_{rr}$ are refinement of interaction obligations as defined by definitions (11) and (12), respectively.

Limited and restricted limited refinement Both versions of definition (21) allow a refinement to introduce new interaction obligations that are not refinements of any interaction obligations in the original specification, possibly increasing the inherent nondeterminism required of the final system. A trace may be positive in one of these new interaction obligations even if it is negative in all other interaction obligations. In limited ($\rightsquigarrow_l$) and restricted limited ($\rightsquigarrow_{rl}$) refinement, adding new interaction obligations like this is not allowed:

$$\begin{aligned}[\![\ d\ ]\!] \rightsquigarrow_{(r)l} [\![\ d'\ ]\!] \stackrel{\mathsf{def}}{=}\ & [\![\ d\ ]\!] \rightsquigarrow_{(r)g} [\![\ d'\ ]\!] \\ & \wedge \forall o' \in [\![\ d'\ ]\!] : \exists o \in [\![\ d\ ]\!] : o \rightsquigarrow_{(r)r} o'\end{aligned} \tag{22}$$

5.2 Compliance

In order to characterize compliance between a system I and a sequence diagram d with inherent nondeterminism (as well as underspecification), we redefine $\langle I \rangle_d$ to consist of one interaction obligation for each trace in $traces(I)$:

$$\langle I \rangle_d \stackrel{\mathsf{def}}{=} \{(\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\}) \mid h \in traces(I)\} \tag{23}$$

Corresponding to the four refinement relations in Section 5.1, we then define four different compliance relations.

General and restricted general compliance Similar to (restricted) general refinement, a system I is in (restricted) general compliance with a sequence diagram d if every interaction obligation in $[\![\ d\ ]\!]$ is reflected in at least one of the interaction obligations we get by using definition (23). The only difference between general ($\mapsto_g$) and restricted general ($\mapsto_{rg}$) compliance

is with respect to the compliance relation used for each single interaction obligation.

$$[\![\ d\ ]\!] \mapsto_{(r)g} \langle I \rangle_d \stackrel{\mathsf{def}}{=} \forall o \in [\![\ d\ ]\!] : \exists o' \in \langle I \rangle_d : o \mapsto_{(r)r} o' \tag{24}$$

Limited and restricted limited compliance Similar to (restricted) limited refinement, (restricted) limited compliance requires that every interaction obligation obtained by definition (23) complies with at least one interaction obligation in $[\![\ d\ ]\!]$:

$$\begin{aligned} [\![\ d\ ]\!] \mapsto_{(r)l} \langle I \rangle_d \stackrel{\mathsf{def}}{=}\ & [\![\ d\ ]\!] \mapsto_{(r)g} \langle I \rangle_d \\ & \wedge\ \forall o' \in \langle I \rangle_d : \exists o \in [\![\ d\ ]\!] : o \mapsto_{(r)r} o' \end{aligned} \tag{25}$$

5.3 Example

Figure 3 is a revised specification of the gambling machine, replacing the second alt operator with xalt and adding some more negative behaviours. The ref-construct may be understood as a syntactical shorthand for the contents of the referenced sequence diagram.

The semantics $[\![\ S_2\ ]\!]$ of this sequence diagram is a set of two interaction obligations:

$$\begin{aligned} \{\ (\ & \{\langle !di, ?di, !m(yw), ?m(yw), !do, ?do\rangle, \langle !qu, ?qu, !m(yw), ?m(yw), !do, ?do\rangle, \\ & \langle !di, ?di, !m(yw), !do, ?m(yw), ?do\rangle, \langle !qu, ?qu, !m(yw), !do, ?m(yw), ?do\rangle\}\ , \\ & \{\langle !di, ?di, !m(yl), ?m(yl)\rangle, \langle !qu, ?qu, !m(yl), ?m(yl)\rangle, \\ & \langle !di, ?di, !m(yl), ?m(yl), !do, ?do\rangle, \langle !qu, ?qu, !m(yl), ?m(yl), !do, ?do\rangle, \\ & \langle !di, ?di, !m(yl), !do, ?m(yl), ?do\rangle, \langle !qu, ?qu, !m(yl), !do, ?m(yl), ?do\rangle\}\ )\ , \\ (\ & \{\langle !di, ?di, !m(yl), ?m(yl)\rangle, \langle !qu, ?qu, !m(yl), ?m(yl)\rangle\}\ , \\ & \{\langle !di, ?di, !m(yw), ?m(yw), !do, ?do\rangle, \langle !qu, ?qu, !m(yw), ?m(yw), !do, ?do\rangle, \\ & \langle !di, ?di, !m(yw), !do, ?m(yw), ?do\rangle, \langle !qu, ?qu, !m(yw), !do, ?m(yw), ?do\rangle, \\ & \langle !di, ?di, !m(yl), ?m(yl), !do, ?do\rangle, \langle !qu, ?qu, !m(yl), ?m(yl), !do, ?do\rangle, \\ & \langle !di, ?di, !m(yl), !do, ?m(yl), ?do\rangle, \langle !qu, ?qu, !m(yl), !do, ?m(yl), ?do\rangle\}\ )\ \} \end{aligned}$$

The system I_1 given in section 4.3 is not in compliance with S_2, as the only trace $\langle !di, ?di, !m(yl), ?m(yl)\rangle$ in $\langle I_1 \rangle_{S_2}$ is negative in the first interaction obligation in $[\![\ S_2\ ]\!]$, meaning that this interaction obligation is not reflected in the system as required by both definitions (24) and (25).

However, a system I_2 with trace-set as given by $traces(I_2) = \{t_1, t_2, t_3\}$ where $t_1 = \langle !di, ?di, !m(yw), ?m(yw), !do, ?do\rangle$, $t_2 = \langle !di, ?di, !m(yw), !do, ?m(yw), ?do\rangle$ and $t_3 = \langle !di, ?di, !m(yl), ?m(yl)\rangle$, is in compliance with S_2 according to all of the compliance relations $\mapsto_g$, $\mapsto_{rg}$, $\mapsto_l$ and $\mapsto_{rl}$. To realize this notice that the three traces in $traces(I_2)$ will give rise to three different interaction obligations when using definition (23) of $\langle I_2 \rangle_{S_2}$. The interaction obligation $(\{t_1\}, \mathcal{H}^{ll(S_2)} \setminus \{t_1\})$ is in compliance with the first interaction obligation in $[\![\ S_2\ ]\!]$ according to both $\mapsto_r$ and $\mapsto_{rr}$. The same is the case for the interaction obligation $(\{t_2\}, \mathcal{H}^{ll(S_2)} \setminus \{t_2\})$. Similarly, the interaction obligation $(\{t_3\}, \mathcal{H}^{ll(S_2)} \setminus \{t_3\})$ is in compliance with

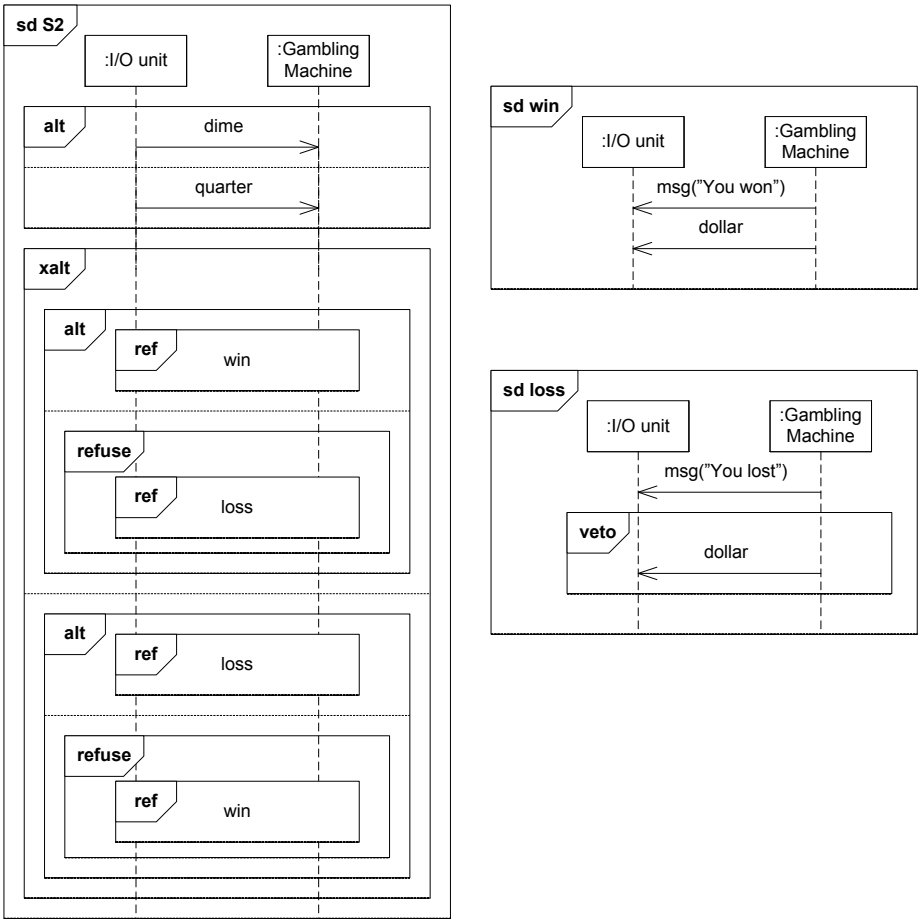

Fig. 3. Sequence diagram with inherent nondeterminism (xalt)

the second interaction obligation in $[\![S_2]\!]$ according to both $\mapsto_r$ and $\mapsto_{rr}$. Hence, both interaction obligations in $[\![S_2]\!]$ are reflected in an interaction obligation in $\langle I_2 \rangle_{S_2}$, and each of the three interaction obligations in $\langle I_2 \rangle_{S_2}$ is in compliance with an interaction obligation in $[\![S_2]\!]$.

6 Results

In this section we present a number of essential properties that has been established, such as transitivity, monotonicity and the relationship between refinement and compliance. For proofs, we refer to the appendices.

Theorem 1. (Transitivity of refinement.) *For every refinement relation $\rightsquigarrow$:*

$$[\![d]\!] \rightsquigarrow [\![d']\!] \wedge [\![d']\!] \rightsquigarrow [\![d'']\!] \Rightarrow [\![d]\!] \rightsquigarrow [\![d'']\!]$$

Transitivity is important, as it ensures that the result of successive refinement steps is a valid refinement of the original sequence diagram. As all other refinement relations are special cases of general refinement, the use of different refinement relations in the various steps ensures that the resulting sequence diagram is at least a general refinement of the original sequence diagram.

Theorem 2. (Monotonicity of refinement.) *For every refinement relation $\rightsquigarrow$:*

$$\begin{aligned}
&[\![d_1]\!] \rightsquigarrow [\![d_1']\!] \ \wedge \ [\![d_2]\!] \rightsquigarrow [\![d_2']\!] \Rightarrow \\
&\ [\![\mathsf{refuse}\ d_1]\!] \rightsquigarrow [\![\mathsf{refuse}\ d_1']\!] \\
&\wedge \ [\![d_1 \ \mathsf{seq}\ d_2]\!] \rightsquigarrow [\![d_1' \ \mathsf{seq}\ d_2']\!] \\
&\wedge \ [\![d_1 \ \mathsf{par}\ d_2]\!] \rightsquigarrow [\![d_1' \ \mathsf{par}\ d_2']\!] \\
&\wedge \ [\![d_1 \ \mathsf{alt}\ d_2]\!] \rightsquigarrow [\![d_1' \ \mathsf{alt}\ d_2']\!] \\
&\wedge \ [\![d_1 \ \mathsf{xalt}\ d_2]\!] \rightsquigarrow [\![d_1' \ \mathsf{xalt}\ d_2']\!]
\end{aligned}$$

Monotonicity ensures that the different parts of a sequence diagram may be refined separately. Again, using different refinement relations means that the resulting sequence diagram will at least be a general refinement of the original one.

Theorem 3. (Transitivity between refinement and compliance.) *For every refinement relation $\rightsquigarrow$ and compliance relation $\mapsto$ with the same subscript:*

$$[\![d_1]\!] \rightsquigarrow [\![d_2]\!] \wedge [\![d_2]\!] \mapsto \langle I \rangle_{d_2} \Rightarrow [\![d_1]\!] \mapsto \langle I \rangle_{d_1}$$

In general, the compliance relation used for relating I to d_2 may be more restrictive (i.e. allowing only a subset of the implementations) than the one that must be used for relating I to d_1. Theorem 3 is important as it tells us that in this case, the compliance relation to be used for relating I to d_1 is the one corresponding to the refinement relation used when going from d_1 to d_2.

A sequence diagram with no xalt operator can be viewed either as a diagram with underspecification or as a diagram with inherent nondeterminism, since it contains only operators that are legal in both of these variations. The following Theorem 4 characterizes the relationships between the different interpretations with respect to refinement and compliance.

Until now we have overloaded the notation for the semantic representation of diagrams and computer systems in order to enhance readability. We now need to introduce the full notation. Let $[\![d]\!]^u$ and $[\![d]\!]^i$ denote the semantics of the sequence diagram d when viewed as a sequence diagram with underspecification (a single interaction obligation) or inherent nondeterminism (a set of interaction obligations), respectively. Similarly, for a system I we use $\langle I \rangle_d^u$ and $\langle I \rangle_d^i$ to denote its semantic representation with respect to d according to definition (13) or definition (23), respectively.

Theorem 4. (Correspondence.) *For a sequence diagram d without inherent nondeterminism:*

$$[\![d]\!]^u \mapsto_r \langle I \rangle_d^u \Rightarrow [\![d]\!]^i \mapsto_g \langle I \rangle_d^i$$
$$[\![d]\!]^u \mapsto_{rr} \langle I \rangle_d^u \Rightarrow [\![d]\!]^i \mapsto_{rg} \langle I \rangle_d^i$$
$$[\![d]\!]^u \mapsto_r \langle I \rangle_d^u \Leftrightarrow [\![d]\!]^i \mapsto_l \langle I \rangle_d^i$$
$$[\![d]\!]^u \mapsto_{rr} \langle I \rangle_d^u \Leftarrow [\![d]\!]^i \mapsto_{rl} \langle I \rangle_d^i$$

From Theorem 4, we see that (restricted) general compliance $\mapsto_{(r)g}$ allows at least all of the implementations allowed by (restricted) compliance $\mapsto_{(r)r}$. We now explain why (restricted) general compliance $\mapsto_{(r)g}$ may allow other implementations as well. As an example, assume that d is a sequence diagram where the trace t is negative in every interaction obligation in $[\![d]\!]^i$. According to (restricted) general refinement $\rightsquigarrow_{(r)g}$, a valid refinement of d is the sequence diagram d' with semantics $[\![d']\!]^i = [\![d]\!]^i \cup \{(\{t\}, \emptyset)\}$. In other words, (restricted) general refinement $\rightsquigarrow_{(r)g}$ allows the addition of new interaction obligations where a trace may be positive even if it is negative in all of the original interaction obligations. This is also true for (restricted) general compliance $\mapsto_{(r)g}$, which is meant to reflect this refinement relation. However, implementing a negative trace is not allowed by (restricted) compliance $\mapsto_{(r)r}$, where a single interaction obligation is the semantic model used for representing both the sequence diagram and the system.

Implementing negative traces is also not allowed by limited compliance $\mapsto_l$, and we see from Theorem 4 that for sequence diagrams without inherent nondeterminism, compliance $\mapsto_r$ and limited compliance $\mapsto_l$ are always in accordance with each other.

On the other hand, restricted compliance $\mapsto_{rr}$ allows implementations not allowed by restricted limited compliance $\mapsto_{rl}$. As an example, assume that d is a sequence diagram with semantics $[\![d]\!]^u = (\{t_1\}, \emptyset)$ (i.e. $[\![d]\!]^i = \{(\{t_1\}, \emptyset)\}$) and that I is a system such that $traces(I) = \{t_1, t_2\}$. Using definition (13) we get that the single interaction obligation of $\langle I \rangle_d^u$ is $(\{t_1, t_2\}, \mathcal{H}^{ll(d)} \setminus \{t_1, t_2\})$, meaning that I is in restricted compliance $\mapsto_{rr}$ to d. However, using definition (23), we

get $\langle I \rangle_d^i = \{(\{t_1\}, \mathcal{H}^{ll(d)} \setminus \{t_1\}), (\{t_2\}, \mathcal{H}^{ll(d)} \setminus \{t_2\})\}$, which is not in restricted limited compliance $\mapsto_{rl}$ with d as d does not contain any interaction obligation where t_2 is positive. In other words, restricted compliance $\mapsto_{rr}$ allows a system to perform traces that are inconclusive in the sequence diagram, while this is not allowed by restricted limited compliance $\mapsto_{rl}$.

7 Related work

To the best of our knowledge, there is no other paper treating the relationship between computer systems and sequence diagrams as thoroughly as we have done in this paper. The closest is the work by Cengarle and Knapp in [CK04], which defines an implementation notion which has inspired our notion of restricted compliance. However, inherent nondeterminism is not treated.

The basis of this paper is sequence diagrams as defined in e.g. UML 2.1 [OMG06]. As the focus of this paper is on compliance relations and not sequence diagrams as such, we have covered only the most essential of the UML 2.1 operators. In addition, we have considered an operator for specifying inherent nondeterminism This operator is not found in UML 2.1, and neither in most other variants of sequence diagrams such as Message Sequence Charts (MSCs) [ITU99].

Live Sequence Charts (LSCs) [DH99,HM03] is an extension of MSCs, where elements in the diagram may be specified as either universal (mandatory) or existential (optional). An existential chart specifies a behaviour (one or more traces) that must be satisfied by at least one system run. A universal chart specify all allowed traces, but does not require that more than one is implemented. [HM03] defines an operational semantics for LSCs, but as the focus is on simulating the specifications, neither refinement nor compliance relations are defined.

Related to implementations, [Krü00] defines four possible interpretations of a single MSC: negated, exact, existential and universal. The negated interpretation means that the MSC describes behaviour that should not happen, and corresponds to the UML use of negation operators within a sequence diagram. For the exact interpretation, all behaviours that are not in the MSC are forbidden. An existential MSC describes behaviour that cannot be prohibited by the system in all executions. We obtain the same result by letting an interaction obligation contain the existential behaviour as the only positive, while all other behaviours are negative in that interaction obligation. Finally, the universal interpretation describes behaviour that must occur as part of all executions of the system. We have no similar notion to this. As for most work on MSCs, [Krü00] does not include a notion of inherent nondeterminism.

Looking at other specification languages than sequence diagrams, more work has been done on nondeterminism. As the main focus of this paper is on computer systems in relation to sequence diagrams, we refer to [RRS06] for a more general treatment of related work with respect to nondeterminism.

8 Conclusions

For sequence diagrams with underspecification and inherent nondeterminism, we have in this paper defined different refinement relations, their corresponding compliance relations, and investigated the mathematical properties of these relations. Which of the refinement and compliance relations will turn out to be most useful in practice, is an open question that should be subject to future research.

Our general compliance checking procedure for relating systems and sequence diagrams was given in section 1. Together with the defined refinement and compliance relations, the procedure meets the requirements of Section 2 in the following sense:

1. The procedure is independent of any particular programming language or paradigm. All we require, is that there exists some means to obtain the traces of the system.
2. The notion of compliance is a special case of refinement. With the exception of restricted compliance, all compliance relations are special cases of the corresponding refinement relations. Whatever refinement relation is used between two sequence diagrams, any system compliant with the refinement is also compliant with the original diagram.
3. If a system is in compliance with a sequence diagram using the most general compliance relation for sequence diagrams with underspecification, $\mapsto_r$, the most general compliance relation for sequence diagrams with inherent nondeterminism, $\mapsto_g$, may be used with the same result. This means that $\mapsto_g$ allows all systems that are allowed by $\mapsto_r$. For correspondences between the other compliance relations, we refer to Theorem 4 in Section 6.
4. The approach is faithful to the UML 2.1 standard, both with respect to the underlying semantic model using sets of positive and negative traces, and with respect to the semantics given for each concrete operator. In particular, all of our definitions take into account the partial nature of sequence diagrams.

In this paper we have only considered sequence diagrams without external input and output. Our results may be generalized to handle also sequence diagrams with such external communication by in each case defining an adversary representing the environment of the system, and then checking compliance under the assumption of this adversary.

References

[AEY00] Rajeev Alur, Kousha Etessami, and Mihalis Yannakakis. Inference of message sequence charts. In *Proc. 22nd International Confonference on Software Engineering (ICSE 2000)*, pages 304–313. ACM, 2000.

[BS01] Manfred Broy and Ketil Stølen. *Specification and Development of Interactive Systems :* FOCUS *on Streams, Interfaces, and Refinement.* Springer, 2001.

[CK04] María Victoria Cengarle and Alexander Knapp. UML 2.0 interactions: Semantics and refinement. In *Proc. 3rd Int. Wsh. Critical Systems Development with UML (CSDUML'04)*, Technical report TUM-I0415, pages 85–99. Institut für Informatik, Technische Universität München, 2004.

[DH99] Werner Damm and David Harel. LSC's: Breathing life into message sequence charts. In *Proc. 3rd IFIP Int. Conf. on Formal Methods for Open Object-Based Distributed Systems (FMOODS'99)*. Kluwer, 1999.

[HHRS05] Øystein Haugen, Knut Eilif Husa, Ragnhild Kobro Runde, and Ketil Stølen. STAIRS towards formal design with sequence diagrams. *Journal of Software and Systems Modeling*, 22(4):349–458, 2005.

[HHRS06] Øystein Haugen, Knut Eilif Husa, Ragnhild Kobro Runde, and Ketil Stølen. Why timed sequence diagrams require three-event semantics. Technical Report 309, Department of Informatics, University of Oslo, 2006.

[HM03] David Harel and Rami Marelly. *Come, Let's Play.: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer, 2003.

[HS03] Øystein Haugen and Ketil Stølen. STAIRS — Steps to analyze interactions with refinement semantics. In *Proc. International Conference on UML (UML'2003)*, volume 2863 of *LNCS*, pages 388–402. Springer, 2003.

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

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

[OMG06] Object Management Group. *UML 2.1 Superstructure Specification*, document: ptc/06-04-02 edition, 2006.

[RHS07] Ragnhild Kobro Runde, Øystein Haugen, and Ketil Stølen. Refining UML interactions with underspecification and nondeterminsim. Technical Report 325, Department of Informatics, University of Oslo, 2007.

[RRS06] Atle Refsdal, Ragnhild Kobro Runde, and Ketil Stølen. Underspecification, inherent nondeterminism and probability in sequence diagrams. In *Proc. 8th IFIP Int. Conf. on Formal Methods for Open Object-Based Distributed Systems (FMOODS'06)*, volume 4037 of *LNCS*, pages 138–155. Springer, 2006.

[RRS07] Atle Refsdal, Ragnhild Kobro Runde, and Ketil Stølen. Relating computer systems to sequence diagrams with underspecification, inherent nondeterminism and probabilistic choice. Part 2: probabilistic choice. Technical Report 347 (to appear), Department of Informatics, University of Oslo, 2007.

A Summary of results

The following tables summarize results with reference to the relevant theorems.

Property	$\rightsquigarrow_r$	$\rightsquigarrow_{rr}$
Trans.	Lemma 26 in [HHRS06]	Theorem 5
Trans. ref./impl.	Theorem 6	Theorem 7
Mon. w.r.t refuse	Lemma 4 in [RHS07]	Theorem 8
Mon. w.r.t seq	Lemma 30 in [HHRS06]	Theorem 9
Mon. w.r.t par	Lemma 31 in [HHRS06]	Theorem 10
Mon. w.r.t alt	Theorem 11	Theorem 12

Property	$\rightsquigarrow_g$	$\rightsquigarrow_{rg}$
Trans.	Theorem 9 in [HHRS06]	Theorem 13
Trans. ref./impl.	Theorem 15	Theorem 15
Mon. w.r.t refuse	Theorem 7 in [RHS07]	Theorem 17
Mon. w.r.t seq	Theorem 13 in [HHRS06]	Theorem 18
Mon. w.r.t par	Theorem 14 in [HHRS06]	Theorem 19
Mon. w.r.t alt	Theorem 11 in [HHRS06]	Theorem 20
Mon. w.r.t xalt	Theorem 12 in [HHRS06]	Theorem 21

Property	$\rightsquigarrow_l$	$\rightsquigarrow_{rl}$
Trans.	Theorem 6 in [RHS07]	Theorem 14
Trans. ref./impl.	Theorem 16	Theorem 16
Mon. w.r.t refuse	Theorem 8 in [RHS07]	Theorem 22
Mon. w.r.t seq	Theorem 10 in [RHS07]	Theorem 23
Mon. w.r.t par	Theorem 24	Theorem 24
Mon. w.r.t alt	Theorem 11 in [RHS07]	Theorem 25
Mon. w.r.t xalt	Theorem 12 in [RHS07]	Theorem 26

Correspondence properties:

	$\mapsto_g$	$\mapsto_l$
$\mapsto_r$	$\Rightarrow$: Theorem 27	$\Rightarrow$: Theorem 29 $\Leftarrow$: Theorem 30

	$\mapsto_{rg}$	$\mapsto_{rl}$
$\mapsto_{rr}$	$\Rightarrow$: Theorem 28	$\Leftarrow$: Theorem 31

B Proofs

In this section we state and prove each individual theorem. Theorems that are proved in other technical reports are not included.

B.1 Specifications with underspecification

Transitivity

Theorem 5. (Transitivity of $\rightsquigarrow_{rr}$.) *Let d, d' and d'' be sequence diagrams without* xalt. *Then*

$$[\![\ d\ ]\!]^u \rightsquigarrow_{rr} [\![\ d'\ ]\!]^u \wedge [\![\ d'\ ]\!]^u \rightsquigarrow_{rr} [\![\ d''\ ]\!]^u \Rightarrow [\![\ d\ ]\!]^u \rightsquigarrow_{rr} [\![\ d''\ ]\!]^u$$

Proof.

Let: $[\![\ d\ ]\!]^u = (p, n)$
$[\![\ d'\ ]\!]^u = (p', n')$
$[\![\ d''\ ]\!]^u = (p'', n'')$
⟨1⟩1. Assume: 1. $(p, n) \rightsquigarrow_{rr} (p', n')$
2. $(p', n') \rightsquigarrow_{rr} (p'', n'')$
Prove: $(p, n) \rightsquigarrow_{rr} (p'', n'')$
⟨2⟩1. Requirement 1: $(p, n) \rightsquigarrow_{r} (p'', n'')$
⟨3⟩1. $(p, n) \rightsquigarrow_{r} (p', n')$
Proof: ⟨1⟩1:1 and definition 12 of $\rightsquigarrow_{rr}$.
⟨3⟩2. $(p', n') \rightsquigarrow_{r} (p'', n'')$
Proof: ⟨1⟩1:2 and definition 12 of $\rightsquigarrow_{rr}$.
⟨3⟩3. Q.E.D.
Proof: ⟨2⟩1, ⟨3⟩2 and lemma 26 in [HHRS06] (transitivity of $\rightsquigarrow_{r}$).
⟨2⟩2. Requirement 2: $p'' \subseteq p$
⟨3⟩1. $p' \subseteq p$
Proof: ⟨1⟩1:1 and definition 12 of $\rightsquigarrow_{rr}$.
⟨3⟩2. $p'' \subseteq p'$
Proof: ⟨1⟩1:2 and definition 12 of $\rightsquigarrow_{rr}$.
⟨3⟩3. Q.E.D.
Proof: ⟨2⟩1, ⟨2⟩2 and transitivity of $\subseteq$.
⟨2⟩3. Q.E.D.
Proof: ⟨2⟩1, ⟨2⟩2 and definition 12 of $\rightsquigarrow_{rr}$.
⟨1⟩2. Q.E.D.
Proof: $\Rightarrow$-rule.

□

Transitivity between refinement and implementation

Theorem 6. (Transitivity between refinement and implementation for $\rightsquigarrow_{r}$.) *Let d_1 and d_2 be sequence diagrams without* xalt. *Then*

$$[\![\ d_1\ ]\!]^u \rightsquigarrow_{r} [\![\ d_2\ ]\!]^u \wedge [\![\ d_2\ ]\!]^u \mapsto_{r} \langle I \rangle^u_{d_2} \Rightarrow [\![\ d_1\ ]\!]^u \mapsto_{r} \langle I \rangle^u_{d_1}$$

PROOF.

LET: $[\![d_1]\!]^u = (p_1, n_1)$
$[\![d_2]\!]^u = (p_2, n_2)$
⟨1⟩1. $\langle I \rangle^u_{d_1} = (\mathit{traces}(I), \mathcal{H}^{ll(d_1)} \setminus \mathit{traces}(I))$
PROOF: Definition 13 of $\langle I \rangle^u_d$.
⟨1⟩2. $\langle I \rangle^u_{d_2} = (\mathit{traces}(I), \mathcal{H}^{ll(d_2)} \setminus \mathit{traces}(I))$
PROOF: Definition 13 of $\langle I \rangle^u_d$.
⟨1⟩3. ASSUME: 1. $(p_1, n_1) \rightsquigarrow_r (p_2, n_2)$
2. $(p_2, n_2) \mapsto_r (\mathit{traces}(I), \mathcal{H}^{ll(d_2)} \setminus \mathit{traces}(I))$
i.e. $(p_2, n_2) \rightsquigarrow_r (\mathit{traces}(I), \mathcal{H}^{ll(d_2)} \setminus \mathit{traces}(I))$ by definition 14 of $\mapsto_r$.
PROVE: $(p_1, n_1) \mapsto_r (\mathit{traces}(I), \mathcal{H}^{ll(d_1)} \setminus \mathit{traces}(I))$
⟨2⟩1. Requirement 1: $n_1 \subseteq \mathcal{H}^{ll(d_1)} \setminus \mathit{traces}(I)$
⟨3⟩1. $n_1 \subseteq \mathcal{H}^{ll(d_2)} \setminus \mathit{traces}(I)$
PROOF: ⟨1⟩3:1, ⟨1⟩3:2, lemma 26 in [HHRS06] (transitivity of $\rightsquigarrow_r$) and definition 11 of $\rightsquigarrow_r$.
⟨3⟩2. $n_1 \subseteq \mathcal{H}^{ll(d_1)}$
PROOF: $[\![d_1]\!]^u = (p_1, n_1)$, definition of $ll(d)$ and definition of $\mathcal{H}^L$.
⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩2 and $A \subseteq B \wedge A \subseteq C \setminus X \Rightarrow A \subseteq B \setminus X$ for arbitrary sets A, B, C and X.
⟨2⟩2. Requirement 2: $p_1 \subseteq \mathit{traces}(I) \cup (\mathcal{H}^{ll(d_1)} \setminus \mathit{traces}(I))$,
i.e. $p_1 \subseteq \mathit{traces}(I) \cup \mathcal{H}^{ll(d_1)}$
PROOF: $p_1 \subseteq \mathcal{H}^{ll(d)}$ by $[\![d_1]\!]^u = (p_1, n_1)$, definition of $ll(d)$ and definition of $\mathcal{H}^L$.
⟨2⟩3. $(p_1, n_1) \rightsquigarrow_r (\mathit{traces}(I), \mathcal{H}^{ll(d_1)} \setminus \mathit{traces}(I))$
PROOF: ⟨2⟩1, ⟨2⟩2 and definition 11 of $\rightsquigarrow_r$.
⟨2⟩4. Q.E.D.
PROOF: ⟨2⟩3 and definition 14 of $\mapsto_r$.
⟨1⟩4. Q.E.D.
PROOF: ⟨1⟩3 and ⇒-rule.

□

Theorem 7. (Transitivity between refinement and implementation for $\rightsquigarrow_{rr}$.) *Let d_1 and d_2 be sequence diagrams without* xalt. *Then*

$$[\![d_1]\!]^u \rightsquigarrow_{rr} [\![d_2]\!]^u \wedge [\![d_2]\!]^u \mapsto_{rr} \langle I \rangle^u_{d_2} \Rightarrow [\![d_1]\!]^u \mapsto_{rr} \langle I \rangle^u_{d_1}$$

PROOF.

LET: $[\![d_1]\!]^u = (p_1, n_1)$
$[\![d_2]\!]^u = (p_2, n_2)$
⟨1⟩1. ASSUME: 1. $[\![d_1]\!]^u \rightsquigarrow_{rr} [\![d_2]\!]^u$
2. $[\![d_2]\!]^u \mapsto_{rr} \langle I \rangle^u_{d_2}$
PROVE: $[\![d_1]\!]^u \mapsto_{rr} \langle I \rangle^u_{d_1}$
⟨2⟩1. Requirement 1: $[\![d_1]\!]^u \mapsto_r \langle I \rangle^u_{d_1}$
⟨3⟩1. $[\![d_1]\!]^u \rightsquigarrow_r [\![d_2]\!]^u$

PROOF: $\langle 1 \rangle 1$:1 and definition 12 of $\rightsquigarrow_{rr}$.

$\langle 3 \rangle 2$. $[\![d_2]\!]^u \mapsto_r \langle I \rangle^u_{d_2}$

PROOF: $\langle 1 \rangle 1$:2 and definition 15 of $\mapsto_{rr}$.

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

PROOF: $\langle 3 \rangle 1$, $\langle 3 \rangle 2$ and theorem 6 (transitivity between refinement and implementation for $\rightsquigarrow_r$).

$\langle 2 \rangle 2$. Requirement 2: $\pi_1([\![d_1]\!]^u) \cap \pi_1(\langle I \rangle^u_{d_1}) \neq \emptyset$

$\langle 3 \rangle 1$. $\pi_1([\![d_1]\!]^u) = p_1$

PROOF: $[\![d_1]\!]^u = (p_1, n_1)$ and definition of π_1.

$\langle 3 \rangle 2$. $\pi_1(\langle I \rangle^u_{d_1}) = \mathit{traces}(I)$

PROOF: Definition 13 of $\langle I \rangle_d$ and definition of π_1.

$\langle 3 \rangle 3$. $p_2 \subseteq p_1$

PROOF: $\langle 1 \rangle 1$:1 and definition 12 of $\rightsquigarrow_{rr}$.

$\langle 3 \rangle 4$. $p_2 \cap \mathit{traces}(I) \neq \emptyset$

PROOF: $\langle 1 \rangle 1$:2 and definition 15 of $\mapsto_{rr}$.

$\langle 3 \rangle 5$. $p_1 \cap \mathit{traces}(I) \neq \emptyset$

PROOF: $\langle 3 \rangle 3$, $\langle 3 \rangle 4$ and $A \subseteq B \wedge A \cap X \neq \emptyset \Rightarrow B \cap X \neq \emptyset$ for arbitrary sets A, B and X.

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

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

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

PROOF: $\langle 2 \rangle 1$, $\langle 2 \rangle 2$ and definition 15 of $\mapsto_{rr}$.

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

PROOF: $\Rightarrow$-rule.

□

Monotonicity

Lemma 1. *(To be used when proving monotonicity with respect to* seq.*)*

ASSUME: 1. $s_1 \subseteq s_1'$
2. $s_2 \subseteq s_2'$
PROVE: $s_1 \succsim s_2 \subseteq s_1' \succsim s_2'$
PROOF.
This is lemma 27 in [HHRS06]. □

Lemma 2. *(To be used when proving monotonicity with respect to* par.*)*

ASSUME: 1. $s_1 \subseteq s_1'$
2. $s_2 \subseteq s_2'$
PROVE: $s_1 \parallel s_2 \subseteq s_1' \parallel s_2'$
PROOF.
This is lemma 28 in [HHRS06]. □

Theorem 8. (Monotonicity of $\rightsquigarrow_{rr}$ w.r.t refuse.**)** *Let d be a sequence diagram without* xalt. *Then*

$$[\![d]\!]^u \rightsquigarrow_{rr} [\![d']\!]^u \Rightarrow [\![\mathsf{refuse}\ d]\!]^u \rightsquigarrow_{rr} [\![\mathsf{refuse}\ d']\!]^u$$

PROOF.

LET: $[\![d]\!]^u = (p, n)$
$[\![d']\!]^u = (p', n')$
⟨1⟩1. ASSUME: $[\![d]\!]^u \rightsquigarrow_{rr} [\![d']\!]^u$,
i.e. $(p, n) \rightsquigarrow_{rr} (p', n')$
PROVE: $[\![\mathsf{refuse}\ d]\!]^u \rightsquigarrow_{rr} [\![\mathsf{refuse}\ d']\!]^u$,
i.e. $(\emptyset, p \cup n) \rightsquigarrow_{rr} (\emptyset, p' \cup n')$ by definition 8 of refuse.
⟨2⟩1. Requirement 1: $(\emptyset, p \cup n) \rightsquigarrow_r (\emptyset, p' \cup n')$
⟨3⟩1. $(p, n) \rightsquigarrow_r (p', n')$
PROOF: ⟨1⟩1 and definition 12 of $\rightsquigarrow_{rr}$.
⟨3⟩2. Q.E.D.
PROOF: ⟨3⟩1 and lemma 4 in [RHS07] (monotonicity of $\rightsquigarrow_r$ w.r.t refuse).
⟨2⟩2. Requirement 2: $\emptyset \subseteq \emptyset$
PROOF: Trivial.
⟨2⟩3. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩2 and definition 12 of $\rightsquigarrow_{rr}$.
⟨1⟩2. Q.E.D.
PROOF: ⇒-rule.

□

Theorem 9. (Monotonicity of $\rightsquigarrow_{rr}$ w.r.t seq.) *Let d_1, d_2, d_1' and d_2' be sequence diagrams without* xalt*. Then*

$$[\![\ d_1\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_1'\ ]\!]^u \wedge [\![\ d_2\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_2'\ ]\!]^u \Rightarrow [\![\ d_1 \ \mathsf{seq}\ d_2\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_1' \ \mathsf{seq}\ d_2'\ ]\!]^u$$

PROOF.

LET: $[\![\ d_1\ ]\!]^u = (p_1, n_1)$
$[\![\ d_1'\ ]\!]^u = (p_1', n_1')$
$[\![\ d_2\ ]\!]^u = (p_2, n_2)$
$[\![\ d_2'\ ]\!]^u = (p_2', n_2')$

⟨1⟩1. ASSUME: 1. $[\![\ d_1\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_1'\ ]\!]^u$,
i.e. $(p_1, n_1) \rightsquigarrow_{rr} (p_1', n_1')$
2. $[\![\ d_2\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_2'\ ]\!]^u$,
i.e. $(p_2, n_2) \rightsquigarrow_{rr} (p_2', n_2')$
PROVE: $[\![\ d_1 \ \mathsf{seq}\ d_2\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_1' \ \mathsf{seq}\ d_2'\ ]\!]^u$,
i.e. $(p_1, n_1) \succsim (p_2, n_2) \rightsquigarrow_{rr} (p_1', n_1') \succsim (p_2', n_2')$ by definition 7
i.e. $(p_1 \succsim p_2, n_1 \succsim p_2 \cup n_1 \succsim n_2 \cup p_1 \succsim n_2)$
$\rightsquigarrow_{rr} (p_1' \succsim p_2', n_1' \succsim p_2' \cup n_1' \succsim n_2' \cup p_1' \succsim n_2')$ by definition 4.

⟨2⟩1. Requirement 1: $(p_1, n_1) \succsim (p_2, n_2) \rightsquigarrow_r (p_1', n_1') \succsim (p_2', n_2')$

⟨3⟩1. $(p_1, n_1) \rightsquigarrow_r (p_1', n_1')$
PROOF: ⟨1⟩1:1 and definition 12 of $\rightsquigarrow_{rr}$.

⟨3⟩2. $(p_2, n_2) \rightsquigarrow_r (p_2', n_2')$
PROOF: ⟨1⟩1:2 and definition 12 of $\rightsquigarrow_{rr}$.

⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩2 and lemma 30 in [HHRS06] (monotonicity of $\rightsquigarrow_r$ w.r.t seq).

⟨2⟩2. Requirement 2: $(p_1' \succsim p_2') \subseteq (p_1 \succsim p_2)$

⟨3⟩1. $p_1' \subseteq p_1$
PROOF: ⟨1⟩1:1 and definition 12 of $\rightsquigarrow_{rr}$.

⟨3⟩2. $p_2' \subseteq p_2$
PROOF: ⟨1⟩1:2 and definition 12 of $\rightsquigarrow_{rr}$.

⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩2 and lemma 1.

⟨2⟩3. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩2 and definition 12 of $\rightsquigarrow_{rr}$.

⟨1⟩2. Q.E.D.
PROOF: ⇒-rule.

□

Theorem 10. (Monotonicity of $\rightsquigarrow_{rr}$ w.r.t par.) *Let d_1, d_2, d_1' and d_2' be sequence diagrams without* xalt*. Then*

$$[\![\ d_1\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_1'\ ]\!]^u \wedge [\![\ d_2\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_2'\ ]\!]^u \Rightarrow [\![\ d_1 \ \mathsf{par}\ d_2\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_1' \ \mathsf{par}\ d_2'\ ]\!]^u$$

PROOF.

LET: $[\![\ d_1\ ]\!]^u = (p_1, n_1)$
$[\![\ d_1'\ ]\!]^u = (p_1', n_1')$
$[\![\ d_2\ ]\!]^u = (p_2, n_2)$
$[\![\ d_2'\ ]\!]^u = (p_2', n_2')$

⟨1⟩1. ASSUME: 1. $[\![\ d_1\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_1'\ ]\!]^u$,
i.e. $(p_1, n_1) \rightsquigarrow_{rr} (p_1', n_1')$
2. $[\![\ d_2\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_2'\ ]\!]^u$,
i.e. $(p_2, n_2)^u \rightsquigarrow_{rr} (p_2', n_2')$
PROVE: $[\![\ d_1 \ \mathsf{par}\ d_2\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_1' \ \mathsf{par}\ d_2'\ ]\!]^u$,
i.e. $(p_1, n_1) \parallel (p_2, n_2) \rightsquigarrow_{rr} (p_1', n_1') \parallel (p_2', n_2')$ by definition 6,
i.e. $(p_1 \parallel p_2, n_1 \parallel p_2 \cup n_1 \parallel n_2 \cup p_1 \parallel n_2)$
$\rightsquigarrow_{rr} (p_1' \parallel p_2', n_1' \parallel p_2' \cup n_1' \parallel n_2' \cup p_1' \parallel n_2')$ by definition 3.

⟨2⟩1. Requirement 1: $(p_1, n_1) \parallel (p_2, n_2) \rightsquigarrow_r (p_1', n_1') \parallel (p_2', n_2')$

⟨3⟩1. $(p_1, n_1) \rightsquigarrow_r (p_1', n_1')$
PROOF: ⟨1⟩1:1 and definition 12 of $\rightsquigarrow_{rr}$.

⟨3⟩2. $(p_2, n_2) \rightsquigarrow_r (p_2', n_2')$
PROOF: ⟨1⟩1:2 and definition 12 of $\rightsquigarrow_{rr}$.

⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩2 and lemma 31 in [HHRS06] (monotonicity of $\rightsquigarrow_r$ w.r.t par).

⟨2⟩2. Requirement 2: $(p_1' \parallel p_2') \subseteq (p_1 \parallel p_2)$

⟨3⟩1. $p_1' \subseteq p_1$
PROOF: ⟨1⟩1:1 and definition 12 of $\rightsquigarrow_{rr}$.

⟨3⟩2. $p_2' \subseteq p_2$
PROOF: ⟨1⟩1:2 and definition 12 of $\rightsquigarrow_{rr}$.

⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩2 and lemma 2.

⟨2⟩3. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩2 and definition 12 of $\rightsquigarrow_{rr}$.

⟨1⟩2. Q.E.D.
PROOF: ⇒-rule.

□

Theorem 11. (Monotonicity of $\rightsquigarrow_r$ w.r.t alt.) *Let d_1, d_2, d_1' and d_2' be sequence diagrams without* xalt*. Then*

$$[\![\ d_1\ ]\!]^u \rightsquigarrow_r [\![\ d_1'\ ]\!]^u \wedge [\![\ d_2\ ]\!]^u \rightsquigarrow_r [\![\ d_2'\ ]\!]^u \Rightarrow [\![\ d_1 \ \mathsf{alt}\ d_2\ ]\!]^u \rightsquigarrow_r [\![\ d_1' \ \mathsf{alt}\ d_2'\ ]\!]^u$$

PROOF.

LET: $[\![\ d_1\ ]\!]^u = (p_1, n_1)$
$[\![\ d_1'\ ]\!]^u = (p_1', n_1')$
$[\![\ d_2\ ]\!]^u = (p_2, n_2)$
$[\![\ d_2'\ ]\!]^u = (p_2', n_2')$

⟨1⟩1. ASSUME: 1. $[\![\ d_1\ ]\!]^u \rightsquigarrow_r [\![\ d_1'\ ]\!]^u$,
i.e. $(p_1, n_1) \rightsquigarrow_r (p_1', n_1')$
2. $[\![\ d_2\ ]\!]^u \rightsquigarrow_r [\![\ d_2'\ ]\!]^u$,
i.e. $(p_2, n_2)^u \rightsquigarrow_r (p_2', n_2')$
PROVE: $[\![\ d_1 \ \mathsf{alt}\ d_2\ ]\!]^u \rightsquigarrow_r [\![\ d_1' \ \mathsf{alt}\ d_2'\ ]\!]^u$,
i.e. $(p_1, n_1) \uplus (p_2, n_2) \rightsquigarrow_r (p_1', n_1') \uplus (p_2', n_2')$ by definition 9,
i.e. $(p_1 \cup p_2, n_1 \cup n_2) \rightsquigarrow_r (p_1' \cup p_2', n_1' \cup n_2')$ by definition 10.
⟨2⟩1. Requirement 1: $(n_1 \cup n_2) \subseteq (n_1' \cup n_2')$
⟨3⟩1. $n_1 \subseteq n_1'$
PROOF: ⟨1⟩1:1 and definition 11 of $\rightsquigarrow_r$.
⟨3⟩2. $n_2 \subseteq n_2'$
PROOF: ⟨1⟩1:2 and definition 11 of $\rightsquigarrow_r$.
⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1 and ⟨3⟩2.
⟨2⟩2. Requirement 2: $(p_1 \cup p_2) \subseteq ((p_1' \cup p_2') \cup (n_1' \cup n_2'))$
⟨3⟩1. $p_1 \subseteq p_1' \cup n_1'$
PROOF: ⟨1⟩1:1 and definition 11 of $\rightsquigarrow_r$.
⟨3⟩2. $p_2 \subseteq p_2' \cup n_2'$
PROOF: ⟨1⟩1:2 and definition 11 of $\rightsquigarrow_r$.
⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩2 and associativity and commutativity of $\cup$.
⟨2⟩3. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩2 and definition 11 of $\rightsquigarrow_r$.
⟨1⟩2. Q.E.D.
PROOF: $\Rightarrow$-rule.

□

Theorem 12. (Monotonicity of $\rightsquigarrow_{rr}$ w.r.t alt.) *Let d_1, d_2, d_1' and d_2' be sequence diagrams without* xalt*. Then*

$$[\![\ d_1\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_1'\ ]\!]^u \wedge [\![\ d_2\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_2'\ ]\!]^u \Rightarrow [\![\ d_1 \ \mathsf{alt}\ d_2\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_1' \ \mathsf{alt}\ d_2'\ ]\!]^u$$

PROOF.

LET: $[\![\ d_1\ ]\!]^u = (p_1, n_1)$
$[\![\ d_1'\ ]\!]^u = (p_1', n_1')$
$[\![\ d_2\ ]\!]^u = (p_2, n_2)$
$[\![\ d_2'\ ]\!]^u = (p_2', n_2')$

⟨1⟩1. ASSUME: 1. $[\![\ d_1\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_1'\ ]\!]^u$,
i.e. $(p_1, n_1) \rightsquigarrow_{rr} (p_1', n_1')$
2. $[\![\ d_2\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_2'\ ]\!]^u$,
i.e. $(p_2, n_2)^u \rightsquigarrow_{rr} (p_2', n_2')$
PROVE: $[\![\ d_1 \ \mathsf{alt}\ d_2\ ]\!]^u \rightsquigarrow_{rr} [\![\ d_1' \ \mathsf{alt}\ d_2'\ ]\!]^u$,
i.e. $(p_1, n_1) \uplus (p_2, n_2) \rightsquigarrow_{rr} (p_1', n_1') \uplus (p_2', n_2')$ by definition 9,
i.e. $(p_1 \cup p_2, n_1 \cup n_2) \rightsquigarrow_r (p_1' \cup p_2', n_1' \cup n_2')$ by definition 10.
⟨2⟩1. Requirement 1: $(p_1, n_1) \uplus (p_2, n_2) \rightsquigarrow_r (p_1', n_1') \uplus (p_2', n_2')$
⟨3⟩1. $(p_1, n_1) \rightsquigarrow_r (p_1', n_1')$
PROOF: ⟨1⟩1:1 and definition 12 of $\rightsquigarrow_{rr}$.
⟨3⟩2. $(p_2, n_2) \rightsquigarrow_r (p_2', n_2')$
PROOF: ⟨1⟩1:2 and definition 12 of $\rightsquigarrow_{rr}$.
⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩2 and theorem 11 (monotonicity of $\rightsquigarrow_r$ w.r.t alt).
⟨2⟩2. Requirement 2: $(p_1' \cup p_2') \subseteq (p_1 \cup p_2)$
⟨3⟩1. $p_1' \subseteq p_1$
PROOF: ⟨1⟩1:1 and definition 12 of $\rightsquigarrow_{rr}$.
⟨3⟩2. $p_2' \subseteq p_2$
PROOF: ⟨1⟩1:2 and definition 12 of $\rightsquigarrow_{rr}$.
⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1 and ⟨3⟩2.
⟨2⟩3. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩2 and definition 12 of $\rightsquigarrow_{rr}$.
⟨1⟩2. Q.E.D.
PROOF: ⇒-rule.

□

B.2 Specifications with inherent nondeterminism

Transitivity

Theorem 13. (Transitivity of $\rightsquigarrow_{rg}$.) *Let d, d' and d'' be sequence diagrams that may contain* xalt. *Then*

$$[\![d]\!]^i \rightsquigarrow_{rg} [\![d']\!]^i \wedge [\![d']\!]^i \rightsquigarrow_{rg} [\![d'']\!]^i \Rightarrow [\![d]\!]^i \rightsquigarrow_{rg} [\![d'']\!]^i$$

Proof.

⟨1⟩1. Assume: 1. $[\![d]\!]^i \rightsquigarrow_{rg} [\![d']\!]^i$
2. $[\![d']\!]^i \rightsquigarrow_{rg} [\![d'']\!]^i$
Prove: $[\![d]\!]^i \rightsquigarrow_{rg} [\![d'']\!]^i$
⟨2⟩1. Choose arbitrary $o \in [\![d]\!]^i$
Proof: $[\![d]\!]^i$ is non-empty for all interactions d.
⟨2⟩2. Choose $o' \in [\![d']\!]^i$ such that $o \rightsquigarrow_{rr} o'$
Proof: ⟨2⟩1, ⟨1⟩1:1 and definition 21 of $\rightsquigarrow_{rg}$.
⟨2⟩3. Choose $o'' \in [\![d'']\!]^i$ such that $o' \rightsquigarrow_{rr} o''$
Proof: ⟨2⟩2, ⟨1⟩1:2 and definition 21 of $\rightsquigarrow_{rg}$.
⟨2⟩4. $o \rightsquigarrow_{rr} o''$
Proof: ⟨2⟩2, ⟨2⟩3 and theorem 5 (transitivity of $\rightsquigarrow_{rr}$).
⟨2⟩5. $\forall o \in [\![d]\!]^i : \exists o'' \in [\![d'']\!]^i : o \rightsquigarrow_{rr} o''$
Proof: ⟨2⟩1, ⟨2⟩3, ⟨2⟩4 and ∀-rule.
⟨2⟩6. Q.E.D.
Proof: ⟨2⟩5 and definition 21 of $\rightsquigarrow_{rg}$.
⟨1⟩2. Q.E.D.
Proof: ⇒-rule.

□

Theorem 14. (Transitivity of $\rightsquigarrow_{rl}$.) *Let d, d' and d'' be sequence diagrams tha may contain* xalt. *Then*

$$[\![d]\!]^i \rightsquigarrow_{rl} [\![d']\!]^i \wedge [\![d']\!]^i \rightsquigarrow_{rl} [\![d'']\!]^i \Rightarrow [\![d]\!]^i \rightsquigarrow_{rl} [\![d'']\!]^i$$

Proof.

⟨1⟩1. Assume: 1. $[\![d]\!]^i \rightsquigarrow_{rl} [\![d']\!]^i$
2. $[\![d']\!]^i \rightsquigarrow_{rl} [\![d'']\!]^i$
Prove: $[\![d]\!]^i \rightsquigarrow_{rl} [\![d'']\!]^i$
⟨2⟩1. $[\![d]\!]^i \rightsquigarrow_{rg} [\![d'']\!]^i$
⟨3⟩1. $[\![d]\!]^i \rightsquigarrow_{rg} [\![d']\!]^i$
Proof: ⟨1⟩1:1 and definition 22 of $\rightsquigarrow_{rl}$.
⟨3⟩2. $[\![d']\!]^i \rightsquigarrow_{rg} [\![d'']\!]^i$
Proof: ⟨1⟩1:2 and definition 22 of $\rightsquigarrow_{rl}$.
⟨3⟩3. Q.E.D.
Proof: ⟨3⟩1, ⟨3⟩2 and theorem 13 (transitivity of $\rightsquigarrow_{rg}$).
⟨2⟩2. $\forall o'' \in [\![d'']\!]^i : \exists o \in [\![d]\!]^i : o \rightsquigarrow_{rr} \ o'$

⟨3⟩1. Choose arbitrary $o'' \in [\![d'']\!]^i$
PROOF: $[\![d]\!]^i$ is non-empty for all interactions d.
⟨3⟩2. Choose $o' \in [\![d']\!]^i$ such that $o' \rightsquigarrow_{rr} o''$
PROOF: ⟨3⟩1, ⟨1⟩1:1 and definition 22 of $\rightsquigarrow_{rl}$.
⟨3⟩3. Choose $o \in [\![d]\!]^i$ such that $o \rightsquigarrow_{rr} o'$
PROOF: ⟨3⟩2, ⟨1⟩1:2 and definition 22 of $\rightsquigarrow_{rl}$.
⟨3⟩4. $o \rightsquigarrow_{rr} o''$
PROOF: ⟨3⟩2, ⟨3⟩3 and theorem 5 (transitivity of $\rightsquigarrow_{rr}$).
⟨3⟩5. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩3, ⟨3⟩4 and ∀-rule.
⟨2⟩3. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩2 and definition 22 of $\rightsquigarrow_{rl}$.
⟨1⟩2. Q.E.D.
PROOF: ⇒-rule.

□

Transitivity between refinement and implementation

Lemma 3. *Let d_1 and d_2 be sequence diagrams that may contain* xalt*, and h a well-formed trace.*

ASSUME: 1. $(p, n) \in [\![d_1]\!]^i$
2. $(p', n') \in [\![d_2]\!]^i$
3. $(p, n) \rightsquigarrow_r (p', n')$
4. $(p', n') \mapsto_r (\{h\}, \mathcal{H}^{ll(d_2)} \setminus \{h\})$
PROVE: $(p, n) \mapsto_r (\{h\}, \mathcal{H}^{ll(d_1)} \setminus \{h\})$

⟨1⟩1. $(p, n) \rightsquigarrow_r (\{h\}, \mathcal{H}^{ll(d_1)} \setminus \{h\})$
⟨2⟩1. Requirement 1: $n \subseteq \mathcal{H}^{ll(d_1)} \setminus \{h\}$
⟨3⟩1. $n \subseteq \mathcal{H}^{ll(d_2)} \setminus \{h\}$
⟨4⟩1. $n \subseteq n'$
PROOF: Assumption 3 and definition 11 of $\rightsquigarrow_r$.
⟨4⟩2. $n' \subseteq \mathcal{H}^{ll(d_2)} \setminus \{h\}$
PROOF: Assumption 4 and definitions 11 and 14 of $\mapsto_r$.
⟨4⟩3. Q.E.D.
PROOF: ⟨4⟩1, ⟨4⟩2 and transitivity of $\subseteq$.
⟨3⟩2. $n \subseteq \mathcal{H}^{ll(d_1)}$
PROOF: Assumption 1, definition of $ll(d)$ and definition of $\mathcal{H}^L$.
⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩2 and $A \subseteq B \wedge A \subseteq C \setminus X \Rightarrow A \subseteq B \setminus X$ for arbitrary sets A, B, C and X.
⟨2⟩2. Requirement 2: $p \subseteq \{h\} \cup (\mathcal{H}^{ll(d_1)} \setminus \{h\})$, i.e. $p \subseteq \{h\} \cup \mathcal{H}^{ll(d_1)}$
PROOF: $p \subseteq \mathcal{H}^{ll(d_1)}$ by assumption 1, definition of $ll(d)$ and definition of $\mathcal{H}^L$.
⟨2⟩3. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩2 and definition 11 of $\rightsquigarrow_r$.

⟨1⟩2. Q.E.D.
PROOF: Definition 14 of $\mapsto_r$.

□

Lemma 4. *Let d_1 and d_2 be sequence diagrams that may contain* xalt*, and h a well-formed trace.*

ASSUME: 1. $(p, n) \in [\![d_1]\!]^i$
2. $(p', n') \in [\![d_2]\!]^i$
3. $(p, n) \rightsquigarrow_{rr} (p', n')$
4. $(p', n') \mapsto_{rr} (\{h\}, \mathcal{H}^{ll(d_2)} \setminus \{h\})$
PROVE: $(p, n) \mapsto_{rr} (\{h\}, \mathcal{H}^{ll(d_1)} \setminus \{h\})$

⟨1⟩1. $(p, n) \rightsquigarrow_r (\{h\}, \mathcal{H}^{ll(d_1)} \setminus \{h\})$
PROOF: Assumptions 1–4 and lemma 3.
⟨1⟩2. $p \cap \{h\} \neq \emptyset$
⟨2⟩1. $p' \subseteq p$
PROOF: Assumption 3 and definition 12 of $\rightsquigarrow_{rr}$.
⟨2⟩2. $p' \cap \{h\} \neq \emptyset$
PROOF: Assumption 4 and definition 15 of $\mapsto_{rr}$.
⟨2⟩3. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩2 and $A \subseteq B \wedge A \cap X \neq \emptyset \Rightarrow B \cap X \neq \emptyset$ for arbitrary sets A, B and X.
⟨1⟩3. Q.E.D.
PROOF: ⟨1⟩1, ⟨1⟩2 and definitions 14 and 15 of $\mapsto_{rr}$.

□

Theorem 15. (Transitivity between refinement and implementation for $\rightsquigarrow_{(r)g}$.) *Let d_1 and d_2 be sequence diagrams that may contain* xalt*. Then*

$$[\![d_1]\!]^i \rightsquigarrow_{(r)g} [\![d_2]\!]^i \wedge [\![d_2]\!]^i \mapsto_{(r)g} \langle I \rangle^i_{d_2} \Rightarrow [\![d_1]\!]^i \mapsto_{(r)g} \langle I \rangle^i_{d_1}$$

PROOF.

⟨1⟩1. $\langle I \rangle^i_{d_1} = \{(\{h\}, \mathcal{H}^{ll(d_1)} \setminus \{h\}) \mid h \in traces(I)\}$
PROOF: Definition 23 of $\langle I \rangle^i_d$.
⟨1⟩2. $\langle I \rangle^i_{d_2} = \{(\{h\}, \mathcal{H}^{ll(d_2)} \setminus \{h\}) \mid h \in traces(I)\}$
PROOF: Definition 23 of $\langle I \rangle^i_d$.
⟨1⟩3. ASSUME: 1. $[\![d_1]\!]^i \rightsquigarrow_{(r)g} [\![d_2]\!]^i$
2. $[\![d_2]\!]^i \mapsto_{(r)g} \langle I \rangle^i_{d_2}$,
i.e. $\forall o \in [\![d_2]\!]^i : \exists o' \in \langle I \rangle^i_{d_2} : o \mapsto_{(r)r} o'$ by definition 24.
PROVE: $[\![d_1]\!]^i \mapsto_{(r)g} \langle I \rangle^i_{d_1}$,
i.e. $\forall o \in [\![d_1]\!]^i : \exists o' \in \langle I \rangle^i_{d_1} : o \mapsto_{(r)r} o'$ by definition 24.
⟨2⟩1. Choose arbitrary $(p, n) \in [\![d_1]\!]^i$
PROOF: $[\![d]\!]^i$ is non-empty for all interactions d.
⟨2⟩2. Choose $(p', n') \in [\![d_2]\!]^i$ such that $(p, n) \rightsquigarrow_{(r)r} (p', n')$
PROOF: ⟨2⟩1, ⟨1⟩3:1 and definition 21 of $\rightsquigarrow_{(r)g}$.

⟨2⟩3. Choose $h \in \mathit{traces}(I)$ such that $(p', n') \mapsto_{(r)r} (\{h\}, \mathcal{H}^{ll(d_2)} \setminus \{h\})$
PROOF: ⟨2⟩2, ⟨1⟩3:2 and ⟨1⟩2.
⟨2⟩4. $(p, n) \mapsto_{(r)r} (\{h\}, \mathcal{H}^{ll(d_1)} \setminus \{h\})$
PROOF: ⟨2⟩1, ⟨2⟩2, ⟨2⟩3 and lemma 3 (4).
⟨2⟩5. $(\{h\}, \mathcal{H}^{ll(d_1)} \setminus \{h\}) \in \langle I \rangle^i_{d_1}$
PROOF: ⟨1⟩1 and ⟨2⟩3.
⟨2⟩6. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩4, ⟨2⟩5 and ∀-rule.
⟨1⟩4. Q.E.D.
PROOF: ⟨1⟩3 and ⇒-rule.

□

Theorem 16. (Transitivity between refinement and implementation for $\rightsquigarrow_{(r)l}$.) *Let d_1 and d_2 be sequence diagrams that may contain* xalt. *Then*

$$[\![d_1]\!]^i \rightsquigarrow_{(r)l} [\![d_2]\!]^i \wedge [\![d_2]\!]^i \mapsto_{(r)l} \langle I \rangle^i_{d_2} \Rightarrow [\![d_1]\!]^i \mapsto_{(r)l} \langle I \rangle^i_{d_1}$$

PROOF.

⟨1⟩1. $\langle I \rangle^i_{d_1} = \{(\{h\}, \mathcal{H}^{ll(d_1)} \setminus \{h\}) \mid h \in \mathit{traces}(I)\}$
PROOF: Definition 23 of $\langle I \rangle^i_d$.
⟨1⟩2. $\langle I \rangle^i_{d_2} = \{(\{h\}, \mathcal{H}^{ll(d_2)} \setminus \{h\}) \mid h \in \mathit{traces}(I)\}$
PROOF: Definition 23 of $\langle I \rangle^i_d$.
⟨1⟩3. ASSUME: 1. $[\![d_1]\!]^i \rightsquigarrow_{(r)l} [\![d_2]\!]^i$
2. $[\![d_2]\!]^i \mapsto_{(r)l} \langle I \rangle^i_{d_2}$,
i.e. $[\![d_2]\!]^i \mapsto_{(r)g} \langle I \rangle^i_{d_2} \wedge \forall o' \in \langle I \rangle^i_{d_2} : \exists o \in [\![d_2]\!]^i : o \mapsto_{(r)r} o'$
by definition 25.
PROVE: $[\![d_1]\!]^i \mapsto_{(r)l} \langle I \rangle^i_{d_1}$
⟨2⟩1. $[\![d_1]\!]^i \mapsto_{(r)g} \langle I \rangle^i_{d_1}$
⟨3⟩1. $[\![d_1]\!]^i \rightsquigarrow_{(r)g} [\![d_2]\!]^i$
PROOF: ⟨1⟩3:1 and definition 22 of $\rightsquigarrow_{(r)l}$.
⟨3⟩2. $[\![d_2]\!]^i \mapsto_{(r)g} \langle I \rangle^i_{d_2}$
PROOF: ⟨1⟩3:2.
⟨3⟩3. Q.E.D.
PROOF: ⟨3⟩1, ⟨3⟩2 and theorem 15 (transitivity between refinement and implementation for $\rightsquigarrow_{(r)g}$).
⟨2⟩2. $\forall o' \in \langle I \rangle^i_{d_1} : \exists o \in [\![d_1]\!]^i : o \mapsto_{(r)r} o'$
⟨3⟩1. Choose arbitrary $o' \in \langle I \rangle^i_{d_1}$
PROOF: $\langle I \rangle^i_{d_1}$ is non-empty for all real systems I.
⟨3⟩2. Choose $h \in \mathit{traces}(I)$ such that $o' = (\{h\}, \mathcal{H}^{ll(d_1)} \setminus \{h\})$
PROOF: ⟨3⟩1 and ⟨1⟩1.
⟨3⟩3. $(\{h\}, \mathcal{H}^{ll(d_2)} \setminus \{h\}) \in \langle I \rangle^i_{d_2}$
PROOF: ⟨3⟩2 and ⟨1⟩2.
⟨3⟩4. Choose $(p', n') \in [\![d_2]\!]^i$ such that $(p', n') \mapsto_{(r)r} (\{h\}, \mathcal{H}^{ll(d_2)} \setminus \{h\})$
PROOF: ⟨3⟩3 and ⟨1⟩3:2.
⟨3⟩5. Choose $(p, n) \in [\![d_1]\!]^i$ such that $(p, n) \rightsquigarrow_{(r)r} (p', n')$

PROOF: ⟨3⟩4, ⟨1⟩3:1 and definition 22 of $\rightsquigarrow_{(r)l}$.

⟨3⟩6. $(p, n) \mapsto_{(r)r} (\{h\}, \mathcal{H}^{ll(d_1)} \setminus \{h\})$

PROOF: ⟨3⟩5, ⟨3⟩4 and lemma 3 (4).

⟨3⟩7. Q.E.D.

PROOF: ⟨3⟩2, ⟨3⟩5, ⟨3⟩6 and ∀-rule.

⟨2⟩3. Q.E.D.

PROOF: ⟨2⟩1, ⟨2⟩2 and definition 25 of $\mapsto_{(r)l}$.

⟨1⟩4. Q.E.D.

PROOF: ⟨1⟩3 and ⇒-rule.

□

Monotonicity

Theorem 17. (Monotonicity of $\rightsquigarrow_{rg}$ w.r.t refuse.) *Let d be a sequence diagram that may contain* xalt*. Then*

$$[\![\ d\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'\ ]\!]^i \Rightarrow [\![\ \mathsf{refuse}\ d\ ]\!]^i \rightsquigarrow_{rg} [\![\ \mathsf{refuse}\ d'\ ]\!]^i$$

PROOF.

⟨1⟩1. ASSUME: $[\![\ d\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'\ ]\!]^i$
PROVE: $[\![\ \mathsf{refuse}\ d\ ]\!]^i \rightsquigarrow_{rg} [\![\ \mathsf{refuse}\ d'\ ]\!]^i$
⟨2⟩1. Choose arbitrary $o = (p, n) \in [\![\ \mathsf{refuse}\ d\ ]\!]^i$
PROOF: $[\![\ d\ ]\!]^i$ is non-empty for all interactions d.
⟨2⟩2. Choose $(p_1, n_1) \in [\![\ d\ ]\!]$ such that $p = \emptyset$ and $n = p_1 \cup n_1$
PROOF: ⟨2⟩1 and definitions 5 and 8 of refuse.
⟨2⟩3. Choose $(p'_1, n'_1) \in [\![\ d'\ ]\!]$ such that $(p_1, n_1) \rightsquigarrow_{rr} (p'_1, n'_1)$
PROOF: ⟨2⟩2, ⟨1⟩1 and definition 21 of $\rightsquigarrow_{rg}$.
⟨2⟩4. $o' = (p', n') = (\emptyset, p'_1 \cup n'_1) \in [\![\ \mathsf{refuse}\ d'\ ]\!]$
PROOF: ⟨2⟩3 and definitions 5 and 8 of refuse.
⟨2⟩5. $(p, n) \rightsquigarrow_{rr} (p', n')$
PROOF: ⟨2⟩2, ⟨2⟩3, ⟨2⟩4 and theorem 8 (monotonicity of $\rightsquigarrow_{rr}$ w.r.t refuse).
⟨2⟩6. $\forall o \in [\![\ \mathsf{refuse}\ d\ ]\!]^i : \exists o' \in [\![\ \mathsf{refuse}\ d'\ ]\!]^i : o \rightsquigarrow_{rr} o'$
PROOF: ⟨2⟩1, ⟨2⟩4, ⟨2⟩5 and ∀-rule.
⟨2⟩7. Q.E.D.
PROOF: ⟨2⟩6 and definition 21 of $\rightsquigarrow_{rg}$.
⟨1⟩2. Q.E.D.
PROOF: ⇒-rule.

□

Theorem 18. (Monotonicity of $\rightsquigarrow_{rg}$ w.r.t seq.) *Let d_1, d_2, d'_1 and d'_2 be sequence diagrams that may contain* xalt*. Then*

$$[\![\ d_1\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'_1\ ]\!]^i \wedge [\![\ d_2\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'_2\ ]\!]^i \Rightarrow [\![\ d_1\ \mathsf{seq}\ d_2\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'_1\ \mathsf{seq}\ d'_2\ ]\!]^i$$

PROOF.

⟨1⟩1. ASSUME: 1. $[\![\ d_1\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'_1\ ]\!]^i$
2. $[\![\ d_2\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'_2\ ]\!]^i$
PROVE: $[\![\ d_1\ \mathsf{seq}\ d_2\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'_1\ \mathsf{seq}\ d'_2\ ]\!]^i$
⟨2⟩1. Choose arbitrary $o = (p, n) \in [\![\ d_1\ \mathsf{seq}\ d_2\ ]\!]^i$
PROOF: $[\![\ d\ ]\!]^i$ is non-empty for all interactions d.
⟨2⟩2. Choose $(p_1, n_1) \in [\![\ d_1\ ]\!]^i$ and $(p_2, n_2) \in [\![\ d_2\ ]\!]^i$ such that $p = p_1 \gtrsim p_2$ and $n = n_1 \gtrsim p_2 \cup n_1 \gtrsim n_2 \cup p_1 \gtrsim n_2$
PROOF: ⟨2⟩1 and definitions 4, 7 and 19 of seq.
⟨2⟩3. Choose $(p'_1, n'_1) \in [\![\ d'_1\ ]\!]^i$ such that $(p_1, n_1) \rightsquigarrow_{rr} (p'_1, n'_1)$
PROOF: ⟨2⟩2, ⟨1⟩1:1 and definition 21 of $\rightsquigarrow_{rg}$.
⟨2⟩4. Choose $(p'_2, n'_2) \in [\![\ d'_2\ ]\!]^i$ such that $(p_2, n_2) \rightsquigarrow_{rr} (p'_2, n'_2)$
PROOF: ⟨2⟩2, ⟨1⟩1:2 and definition 21 of $\rightsquigarrow_{rg}$.

⟨2⟩5. $o' = (p', n') = (p'_1 \succsim p'_2, n'_1 \succsim p'_2 \cup n'_1 \succsim n'_2 \cup p'_1 \succsim n'_2) \in [\![\ d'_1\ \mathsf{seq}\ d'_2\ ]\!]^i$
PROOF: ⟨2⟩3, ⟨2⟩4 and definitions 4, 7 and 19 of seq.
⟨2⟩6. $(p, n) \rightsquigarrow_{rr} (p', n')$
PROOF: ⟨2⟩2, ⟨2⟩3, ⟨2⟩4, ⟨2⟩5 and theorem 9 (monotonicity of $\rightsquigarrow_{rr}$ w.r.t. seq).
⟨2⟩7. $\forall o \in [\![\ d_1\ \mathsf{seq}\ d_2\ ]\!]^i : \exists o' \in [\![\ d'_1\ \mathsf{seq}\ d'_2\ ]\!]^i : o \rightsquigarrow_{rr} o'$
PROOF: ⟨2⟩1, ⟨2⟩5, ⟨2⟩6 and ∀-rule.
⟨2⟩8. Q.E.D.
PROOF: ⟨2⟩7 and definition 21 of $\rightsquigarrow_{rg}$.
⟨1⟩2. Q.E.D.
PROOF: ⇒-rule.

□

Theorem 19. (Monotonicity of $\rightsquigarrow_{rg}$ w.r.t par.) *Let d_1, d_2, d'_1 and d'_2 be sequence diagrams that may contain* xalt. *Then*

$$[\![\ d_1\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'_1\ ]\!]^i \wedge [\![\ d_2\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'_2\ ]\!]^i \Rightarrow [\![\ d_1\ \mathsf{par}\ d_2\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'_1\ \mathsf{par}\ d'_2\ ]\!]^i$$

PROOF.

⟨1⟩1. ASSUME: 1. $[\![\ d_1\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'_1\ ]\!]^i$
2. $[\![\ d_2\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'_2\ ]\!]^i$
PROVE: $[\![\ d_1\ \mathsf{par}\ d_2\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'_1\ \mathsf{par}\ d'_2\ ]\!]^i$
⟨2⟩1. Choose arbitrary $o = (p, n) \in [\![\ d_1\ \mathsf{par}\ d_2\ ]\!]^i$
PROOF: $[\![\ d\ ]\!]^i$ is non-empty for all interactions d.
⟨2⟩2. Choose $(p_1, n_1) \in [\![\ d_1\ ]\!]^i$ and $(p_2, n_2) \in [\![\ d_2\ ]\!]^i$ such that $p = p_1 \parallel p_2$ and $n = n_1 \parallel p_2 \cup n_1 \parallel n_2 \cup p_1 \parallel n_2$
PROOF: ⟨2⟩1 and definitions 3, 6 and 19 of par.
⟨2⟩3. Choose $(p'_1, n'_1) \in [\![\ d'_1\ ]\!]^i$ such that $(p_1, n_1) \rightsquigarrow_{rr} (p'_1, n'_1)$
PROOF: ⟨2⟩2, ⟨1⟩1:1 and definition 21 of $\rightsquigarrow_{rg}$.
⟨2⟩4. Choose $(p'_2, n'_2) \in [\![\ d'_2\ ]\!]^i$ such that $(p_2, n_2) \rightsquigarrow_{rr} (p'_2, n'_2)$
PROOF: ⟨2⟩2, ⟨1⟩1:2 and definition 21 of $\rightsquigarrow_{rg}$.
⟨2⟩5. $o' = (p', n') = (p'_1 \parallel p'_2, n'_1 \parallel p'_2 \cup n'_1 \parallel n'_2 \cup p'_1 \parallel n'_2) \in [\![\ d'_1\ \mathsf{par}\ d'_2\ ]\!]^i$
PROOF: ⟨2⟩3, ⟨2⟩4 and definitions 3, 6 and 19 of par.
⟨2⟩6. $(p, n) \rightsquigarrow_{rr} (p', n')$
PROOF: ⟨2⟩2, ⟨2⟩3, ⟨2⟩4, ⟨2⟩5 and theorem 10 (monotonicity of $\rightsquigarrow_{rr}$ w.r.t. par).
⟨2⟩7. $\forall o \in [\![\ d_1\ \mathsf{par}\ d_2\ ]\!]^i : \exists o' \in [\![\ d'_1\ \mathsf{par}\ d'_2\ ]\!]^i : o \rightsquigarrow_{rr} o'$
PROOF: ⟨2⟩1, ⟨2⟩5, ⟨2⟩6 and ∀-rule.
⟨2⟩8. Q.E.D.
PROOF: ⟨2⟩7 and definition 21 of $\rightsquigarrow_{rg}$.
⟨1⟩2. Q.E.D.
PROOF: ⇒-rule.

□

Theorem 20. (Monotonicity of $\rightsquigarrow_{rg}$ w.r.t alt.) *Let d_1, d_2, d_1' and d_2' be sequence diagrams that may contain* xalt*. Then*

$$[\![d_1]\!]^i \rightsquigarrow_{rg} [\![d_1']\!]^i \wedge [\![d_2]\!]^i \rightsquigarrow_{rg} [\![d_2']\!]^i \Rightarrow [\![d_1 \ \mathsf{alt}\ d_2]\!]^i \rightsquigarrow_{rg} [\![d_1' \ \mathsf{alt}\ d_2']\!]^i$$

Proof.

⟨1⟩1. Assume: 1. $[\![d_1]\!]^i \rightsquigarrow_{rg} [\![d_1']\!]^i$
2. $[\![d_2]\!]^i \rightsquigarrow_{rg} [\![d_2']\!]^i$
Prove: $[\![d_1 \ \mathsf{alt}\ d_2]\!]^i \rightsquigarrow_{rg} [\![d_1' \ \mathsf{alt}\ d_2']\!]^i$
⟨2⟩1. Choose arbitrary $o = (p, n) \in [\![d_1 \ \mathsf{alt}\ d_2]\!]^i$
Proof: $[\![d]\!]^i$ is non-empty for all interactions d.
⟨2⟩2. Choose $(p_1, n_1) \in [\![d_1]\!]^i$ and $(p_2, n_2) \in [\![d_2]\!]^i$ such that $p = p_1 \cup p_2$ and $n = n_1 \cup n_2$
Proof: ⟨2⟩1 and definitions 9, 10 and 19 of alt.
⟨2⟩3. Choose $(p_1', n_1') \in [\![d_1']\!]^i$ such that $(p_1, n_1) \rightsquigarrow_{rr} (p_1', n_1')$
Proof: ⟨2⟩2, ⟨1⟩1:1 and definition 21 of $\rightsquigarrow_{rg}$.
⟨2⟩4. Choose $(p_2', n_2') \in [\![d_2']\!]^i$ such that $(p_2, n_2) \rightsquigarrow_{rr} (p_2', n_2')$
Proof: ⟨2⟩2, ⟨1⟩1:2 and definition 21 of $\rightsquigarrow_{rg}$.
⟨2⟩5. $o' = (p', n') = (p_1' \cup p_2', n_1' \cup n_2') \in [\![d_1' \ \mathsf{alt}\ d_2']\!]^i$
Proof: ⟨2⟩3, ⟨2⟩4 and definitions 9, 10 and 19 of alt.
⟨2⟩6. $(p, n) \rightsquigarrow_{rr} (p', n')$
Proof: ⟨2⟩2, ⟨2⟩3, ⟨2⟩4, ⟨2⟩5 and theorem 12 (monotonicity of $\rightsquigarrow_{rr}$ w.r.t. alt).
⟨2⟩7. $\forall o \in [\![d_1 \ \mathsf{alt}\ d_2]\!]^i : \exists o' \in [\![d_1' \ \mathsf{alt}\ d_2']\!]^i : o \rightsquigarrow_{rr} o'$
Proof: ⟨2⟩1, ⟨2⟩5, ⟨2⟩6 and ∀-rule.
⟨2⟩8. Q.E.D.
Proof: ⟨2⟩7 and definition 21 of $\rightsquigarrow_{rg}$.
⟨1⟩2. Q.E.D.
Proof: ⇒-rule.

□

Theorem 21. (Monotonicity of $\rightsquigarrow_{rg}$ w.r.t xalt.) *Let d_1, d_2, d_1' and d_2' be sequence diagrams that may contain* xalt*. Then*

$$[\![d_1]\!]^i \rightsquigarrow_{rg} [\![d_1']\!]^i \wedge [\![d_2]\!]^i \rightsquigarrow_{rg} [\![d_2']\!]^i \Rightarrow [\![d_1 \ \mathsf{xalt}\ d_2]\!]^i \rightsquigarrow_{rg} [\![d_1' \ \mathsf{xalt}\ d_2']\!]^i$$

Proof.

⟨1⟩1. Assume: 1. $[\![d_1]\!]^i \rightsquigarrow_{rg} [\![d_1']\!]^i$
2. $[\![d_2]\!]^i \rightsquigarrow_{rg} [\![d_2']\!]^i$
Prove: $[\![d_1 \ \mathsf{xalt}\ d_2]\!]^i \rightsquigarrow_{rg} [\![d_1' \ \mathsf{xalt}\ d_2']\!]^i$
⟨2⟩1. Choose arbitrary $o \in [\![d_1 \ \mathsf{xalt}\ d_2]\!]^i$
Proof: $[\![d]\!]^i$ is non-empty for all interactions d.
⟨2⟩2. $\exists o' \in [\![d_1' \ \mathsf{xalt}\ d_2']\!]^i : o \rightsquigarrow_{rr} o'$
⟨3⟩1. Case: $o \in [\![d_1]\!]^i$
⟨4⟩1. Choose $o' \in [\![d_1']\!]^i$ such that $o \rightsquigarrow_{rr} o'$
Proof: ⟨2⟩1, ⟨1⟩1:1 and definition 21 of $\rightsquigarrow_{rg}$.
⟨4⟩2. $o' \in [\![d_1' \ \mathsf{xalt}\ d_2']\!]^i$

PROOF: ⟨4⟩1 and definition 18 of xalt.

⟨4⟩3. Q.E.D.

⟨3⟩2. CASE: $o \in [\![\ d_2\ ]\!]^i$

⟨4⟩1. Choose $o' \in [\![\ d_2'\ ]\!]^i$ such that $o \rightsquigarrow_{rr} o'$

PROOF: ⟨2⟩1, ⟨1⟩1:2 and definition 21 of $\rightsquigarrow_{rg}$.

⟨4⟩2. $o' \in [\![\ d_1' \ \mathsf{xalt}\ d_2'\ ]\!]^i$

PROOF: ⟨4⟩1 and definition 18 of xalt.

⟨4⟩3. Q.E.D.

⟨3⟩3. Q.E.D.

PROOF: The cases are exhaustive by ⟨2⟩1 and definition 18 of xalt.

⟨2⟩3. $\forall o \in [\![\ d_1 \ \mathsf{xalt}\ d_2\ ]\!]^i : \exists o' \in [\![\ d_1' \ \mathsf{xalt}\ d_2'\ ]\!]^i : o \rightsquigarrow_{rr} o'$

PROOF: ⟨2⟩1, ⟨2⟩2 and ∀-rule.

⟨2⟩4. Q.E.D.

PROOF: ⟨2⟩3 and definition 21 of $\rightsquigarrow_{rg}$.

⟨1⟩2. Q.E.D.

PROOF: ⇒-rule.

□

Theorem 22. (Monotonicity of $\rightsquigarrow_{rl}$ w.r.t refuse.) *Let d be a sequence diagram that may contain* xalt. *Then*

$$[\![\ d\ ]\!]^i \rightsquigarrow_{rl} [\![\ d'\ ]\!]^i \Rightarrow [\![\ \mathsf{refuse}\ d\ ]\!]^i \rightsquigarrow_{rl} [\![\ \mathsf{refuse}\ d'\ ]\!]^i$$

PROOF.

⟨1⟩1. ASSUME: $[\![\ d\ ]\!]^i \rightsquigarrow_{rl} [\![\ d'\ ]\!]^i$

PROVE: $[\![\ \mathsf{refuse}\ d\ ]\!]^i \rightsquigarrow_{rl} [\![\ \mathsf{refuse}\ d'\ ]\!]^i$

⟨2⟩1. $[\![\ \mathsf{refuse}\ d\ ]\!]^i \rightsquigarrow_{rg} [\![\ \mathsf{refuse}\ d'\ ]\!]^i$

PROOF: ⟨1⟩1, definition 22 of $\rightsquigarrow_{rl}$ and theorem 17 (monotonicity of $\rightsquigarrow_{rg}$ with respect to refuse).

⟨2⟩2. Choose arbitrary $o' = (p', n') \in [\![\ \mathsf{refuse}\ d'\ ]\!]^i$

PROOF: $[\![\ d\ ]\!]^i$ is non-empty for all interaction d.

⟨2⟩3. Choose $(p_1', n_1') \in [\![\ d'\ ]\!]^i$ such that $p' = \emptyset$ and $n' = p_1' \cup n_1'$

PROOF: ⟨2⟩2 and definitions 8 and 20 of refuse.

⟨2⟩4. Choose $(p_1, n_1) \in [\![\ d\ ]\!]^i$ such that $(p_1, n_1) \rightsquigarrow_{rr} (p_1', n_1')$

PROOF: ⟨2⟩3, ⟨1⟩1 and definition 22 of $\rightsquigarrow_{rl}$.

⟨2⟩5. $o = (p, n) = (\emptyset, p_1 \cup n_1) \in [\![\ \mathsf{refuse}\ d\ ]\!]^i$

PROOF: ⟨2⟩4 and definitions 8 and 20 of refuse.

⟨2⟩6. $(p, n) \rightsquigarrow_{rr} (p', n')$

PROOF: ⟨2⟩3, ⟨2⟩4, ⟨2⟩5 and theorem 8 (monotonicity of $\rightsquigarrow_{rr}$ w.r.t. refuse).

⟨2⟩7. $\forall o' \in [\![\ \mathsf{refuse}\ d'\ ]\!]^i : \exists o \in [\![\ \mathsf{refuse}\ d\ ]\!]^i : o \rightsquigarrow_{rr} o'$

PROOF: ⟨2⟩2, ⟨2⟩5, ⟨2⟩6 and ∀-rule.

⟨2⟩8. Q.E.D.

PROOF: ⟨2⟩1, ⟨2⟩7 and definition 22 of $\rightsquigarrow_{rl}$.

⟨1⟩2. Q.E.D.

PROOF: ⇒-rule.

□

Theorem 23. (Monotonicity of $\rightsquigarrow_{rl}$ w.r.t seq.) *Let d_1, d_2, d_1' and d_2' be sequence diagrams that may contain* xalt. *Then*

$$[\![d_1]\!]^i \rightsquigarrow_{rl} [\![d_1']\!]^i \wedge [\![d_2]\!]^i \rightsquigarrow_{rl} [\![d_2']\!]^i \Rightarrow [\![d_1 \ \mathsf{seq} \ d_2]\!]^i \rightsquigarrow_{rl} [\![d_1' \ \mathsf{seq} \ d_2']\!]^i$$

Proof.

⟨1⟩1. Assume: 1. $[\![d_1]\!]^i \rightsquigarrow_{rl} [\![d_1']\!]^i$
2. $[\![d_2]\!]^i \rightsquigarrow_{rl} [\![d_2']\!]^i$
Prove: $[\![d_1 \ \mathsf{seq} \ d_2]\!]^i \rightsquigarrow_{rl} [\![d_1' \ \mathsf{seq} \ d_2']\!]^i$
⟨2⟩1. $[\![d_1 \ \mathsf{seq} \ d_2]\!]^i \rightsquigarrow_{rg} [\![d_1' \ \mathsf{seq} \ d_2']\!]^i$
Proof: ⟨1⟩1:1, ⟨1⟩1:2, definition 22 of $\rightsquigarrow_{rl}$ and theorem 18 (monotonicity of $\rightsquigarrow_{rg}$ with respect to seq).
⟨2⟩2. Choose arbitrary $o' = (p', n') \in [\![d_1' \ \mathsf{seq} \ d_2']\!]^i$
Proof: $[\![d]\!]^i$ is non-empty for all interactions d.
⟨2⟩3. Choose $(p_1', n_1') \in [\![d_1']\!]^i$ and $(p_2', n_2') \in [\![d_2']\!]^i$ such that $p' = p_1' \gtrsim p_2'$ and $n' = n_1' \gtrsim p_2' \cup n_1' \gtrsim n_2' \cup p_1' \gtrsim n_2'$
Proof: ⟨2⟩2 and definitions 4, 7 and 19 of seq.
⟨2⟩4. Choose $(p_1, n_1) \in [\![d_1]\!]^i$ such that $(p_1, n_1) \rightsquigarrow_{rr} (p_1', n_1')$
Proof: ⟨2⟩3, ⟨1⟩1:1 and definition 22 of $\rightsquigarrow_{rl}$.
⟨2⟩5. Choose $(p_2, n_2) \in [\![d_2]\!]^i$ such that $(p_2, n_2) \rightsquigarrow_{rr} (p_2', n_2')$
Proof: ⟨2⟩3, ⟨1⟩1:2 and definition 22 of $\rightsquigarrow_{rl}$.
⟨2⟩6. $o = (p, n) = (p_1 \gtrsim p_2, n_1 \gtrsim p_2 \cup n_1 \gtrsim n_2 \cup p_1 \gtrsim n_2) \in [\![d_1 \ \mathsf{seq} \ d_2]\!]^i$
Proof: ⟨2⟩4, ⟨2⟩5 and definitions 4, 7 and 19 of seq.
⟨2⟩7. $(p, n) \rightsquigarrow_{rr} (p', n')$
Proof: ⟨2⟩3, ⟨2⟩4, ⟨2⟩5, ⟨2⟩6 and theorem 9 (monotonicity of $\rightsquigarrow_{rr}$ w.r.t. seq).
⟨2⟩8. $\forall o' \in [\![d_1' \ \mathsf{seq} \ d_2']\!]^i : \exists o \in [\![d_1 \ \mathsf{seq} \ d_2]\!]^i : o \rightsquigarrow_{rr} o'$
Proof: ⟨2⟩2, ⟨2⟩6, ⟨2⟩7 and ∀-rule.
⟨2⟩9. Q.E.D.
Proof: ⟨2⟩1, ⟨2⟩8 and definition 22 of $\rightsquigarrow_{rl}$.
⟨1⟩2. Q.E.D.
Proof: ⇒-rule.

□

Theorem 24. (Monotonicity of $\rightsquigarrow_{(r)l}$ w.r.t par.) *Let d_1, d_2, d_1' and d_2' be sequence diagrams that may contain* xalt. *Then*

$$[\![d_1]\!]^i \rightsquigarrow_{(r)l} [\![d_1']\!]^i \wedge [\![d_2]\!]^i \rightsquigarrow_{(r)l} [\![d_2']\!]^i \Rightarrow [\![d_1 \ \mathsf{par} \ d_2]\!]^i \rightsquigarrow_{(r)l} [\![d_1' \ \mathsf{par} \ d_2']\!]^i$$

Proof.

⟨1⟩1. Assume: 1. $[\![d_1]\!]^i \rightsquigarrow_{(r)l} [\![d_1']\!]^i$
2. $[\![d_2]\!]^i \rightsquigarrow_{(r)l} [\![d_2']\!]^i$
Prove: $[\![d_1 \ \mathsf{par} \ d_2]\!]^i \rightsquigarrow_{(r)l} [\![d_1' \ \mathsf{par} \ d_2']\!]^i$
⟨2⟩1. $[\![d_1 \ \mathsf{par} \ d_2]\!]^i \rightsquigarrow_{(r)g} [\![d_1' \ \mathsf{par} \ d_2']\!]^i$
Proof: ⟨1⟩1:1, ⟨1⟩1:2, definition 22 of $\rightsquigarrow_{(r)l}$ and theorem 14 in [HHRS06]/theorem 19 (monotonicity of $\rightsquigarrow_{(r)g}$ with respect to par).
⟨2⟩2. Choose arbitrary $o' = (p', n') \in [\![d_1' \ \mathsf{par} \ d_2']\!]^i$

PROOF: $[\![\ d\ ]\!]^i$ is non-empty for all interactions d.

⟨2⟩3. Choose $(p'_1, n'_1) \in [\![\ d'_1\ ]\!]^i$ and $(p'_2, n'_2) \in [\![\ d'_2\ ]\!]^i$ such that $p' = p'_1 \parallel p'_2$ and $n' = n'_1 \parallel p'_2 \cup n'_1 \parallel n'_2 \cup p'_1 \parallel n'_2$

PROOF: ⟨2⟩2 and definitions 3, 6 and 19 of par.

⟨2⟩4. Choose $(p_1, n_1) \in [\![\ d_1\ ]\!]^i$ such that $(p_1, n_1) \rightsquigarrow_{(r)r} (p'_1, n'_1)$

PROOF: ⟨2⟩3, ⟨1⟩1:1 and definition 22 of $\rightsquigarrow_{(r)l}$.

⟨2⟩5. Choose $(p_2, n_2) \in [\![\ d_2\ ]\!]^i$ such that $(p_2, n_2) \rightsquigarrow_{(r)r} (p'_2, n'_2)$

PROOF: ⟨2⟩3, ⟨1⟩1:2 and definition 22 of $\rightsquigarrow_{(r)l}$.

⟨2⟩6. $o = (p, n) = (p_1 \parallel p_2, n_1 \parallel p_2 \cup n_1 \parallel n_2 \cup p_1 \parallel n_2) \in [\![\ d_1\ \mathsf{par}\ d_2\ ]\!]^i$

PROOF: ⟨2⟩4, ⟨2⟩5 and definitions 3, 6 and 19 of par.

⟨2⟩7. $(p, n) \rightsquigarrow_{(r)r} (p', n')$

PROOF: ⟨2⟩3, ⟨2⟩4, ⟨2⟩5, ⟨2⟩6 and lemma 31 in [HHRS06]/theorem 10 (monotonicity of $\rightsquigarrow_{(r)r}$ w.r.t. par).

⟨2⟩8. $\forall o' \in [\![\ d'_1\ \mathsf{par}\ d'_2\ ]\!]^i : \exists o \in [\![\ d_1\ \mathsf{par}\ d_2\ ]\!]^i : o \rightsquigarrow_{(r)r} o'$

PROOF: ⟨2⟩2, ⟨2⟩6, ⟨2⟩7 and ∀-rule.

⟨2⟩9. Q.E.D.

PROOF: ⟨2⟩1, ⟨2⟩8 and definition 22 of $\rightsquigarrow_{(r)l}$.

⟨1⟩2. Q.E.D.

PROOF: ⇒-rule.

□

Theorem 25. (Monotonicity of $\rightsquigarrow_{rl}$ w.r.t alt.) *Let d_1, d_2, d'_1 and d'_2 be sequence diagrams that may contain* xalt. *Then*

$$[\![\ d_1\ ]\!]^i \rightsquigarrow_{rl} [\![\ d'_1\ ]\!]^i \wedge [\![\ d_2\ ]\!]^i \rightsquigarrow_{rl} [\![\ d'_2\ ]\!]^i \Rightarrow [\![\ d_1\ \mathsf{alt}\ d_2\ ]\!]^i \rightsquigarrow_{rl} [\![\ d'_1\ \mathsf{alt}\ d'_2\ ]\!]^i$$

PROOF.

⟨1⟩1. ASSUME: 1. $[\![\ d_1\ ]\!]^i \rightsquigarrow_{rl} [\![\ d'_1\ ]\!]^i$
2. $[\![\ d_2\ ]\!]^i \rightsquigarrow_{rl} [\![\ d'_2\ ]\!]^i$

PROVE: $[\![\ d_1\ \mathsf{alt}\ d_2\ ]\!]^i \rightsquigarrow_{rl} [\![\ d'_1\ \mathsf{alt}\ d'_2\ ]\!]^i$

⟨2⟩1. $[\![\ d_1\ \mathsf{alt}\ d_2\ ]\!]^i \rightsquigarrow_{rg} [\![\ d'_1\ \mathsf{alt}\ d'_2\ ]\!]^i$

PROOF: ⟨1⟩1:1, ⟨1⟩1:2, definition 22 of $\rightsquigarrow_{rl}$ and theorem 20 (monotonicity of $\rightsquigarrow_{rg}$ with respect to alt).

⟨2⟩2. Choose arbitrary $o' = (p', n') \in [\![\ d'_1\ \mathsf{alt}\ d'_2\ ]\!]^i$

PROOF: $[\![\ d\ ]\!]^i$ is non-empty for all interactions d.

⟨2⟩3. Choose $(p'_1, n'_1) \in [\![\ d'_1\ ]\!]^i$ and $(p'_2, n'_2) \in [\![\ d'_2\ ]\!]^i$ such that $p' = p'_1 \cup p'_2$ and $n' = n'_1 \cup n'_2$

PROOF: ⟨2⟩2 and definitions 9, 10 and 19 of alt.

⟨2⟩4. Choose $(p_1, n_1) \in [\![\ d_1\ ]\!]^i$ such that $(p_1, n_1) \rightsquigarrow_{rr} (p'_1, n'_1)$

PROOF: ⟨2⟩3, ⟨1⟩1:1 and definition 22 of $\rightsquigarrow_{rl}$.

⟨2⟩5. Choose $(p_2, n_2) \in [\![\ d_2\ ]\!]^i$ such that $(p_2, n_2) \rightsquigarrow_{rr} (p'_2, n'_2)$

PROOF: ⟨2⟩3, ⟨1⟩1:2 and definition 22 of $\rightsquigarrow_{rl}$.

⟨2⟩6. $o = (p, n) = (p_1 \cup p_2, n_1 \cup n_2) \in [\![\ d_1\ \mathsf{alt}\ d_2\ ]\!]^i$

PROOF: ⟨2⟩4, ⟨2⟩5 and definitions 9, 10 and 19 of alt.

⟨2⟩7. $(p, n) \rightsquigarrow_{rr} (p', n')$

PROOF: ⟨2⟩3, ⟨2⟩4, ⟨2⟩5, ⟨2⟩6 and theorem 12 (monotonicity of $\rightsquigarrow_{rr}$ w.r.t. alt).

⟨2⟩8. $\forall o' \in [\![d'_1 \ \mathsf{alt}\ d'_2]\!]^i : \exists o \in [\![d_1 \ \mathsf{alt}\ d_2]\!]^i : o \rightsquigarrow_{rr} o'$
PROOF: ⟨2⟩2, ⟨2⟩6, ⟨2⟩7 and ∀-rule.
⟨2⟩9. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩8 and definition 22 of $\rightsquigarrow_{rl}$.
⟨1⟩2. Q.E.D.
PROOF: ⇒-rule.

□

Theorem 26. (Monotonicity of $\rightsquigarrow_{rl}$ w.r.t xalt.) *Let d_1, d_2, d'_1 and d'_2 be sequence diagrams that may contain* xalt. *Then*

$$[\![d_1]\!]^i \rightsquigarrow_{rl} [\![d'_1]\!]^i \wedge [\![d_2]\!]^i \rightsquigarrow_{rl} [\![d'_2]\!]^i \Rightarrow [\![d_1 \ \mathsf{xalt}\ d_2]\!]^i \rightsquigarrow_{rl} [\![d'_1 \ \mathsf{xalt}\ d'_2]\!]^i$$

PROOF.

⟨1⟩1. ASSUME: 1. $[\![d_1]\!]^i \rightsquigarrow_{rl} [\![d'_1]\!]^i$
2. $[\![d_2]\!]^i \rightsquigarrow_{rl} [\![d'_2]\!]^i$
PROVE: $[\![d_1 \ \mathsf{xalt}\ d_2]\!]^i \rightsquigarrow_{rl} [\![d'_1 \ \mathsf{xalt}\ d'_2]\!]^i$
⟨2⟩1. $[\![d_1 \ \mathsf{xalt}\ d_2]\!]^i \rightsquigarrow_{rg} [\![d'_1 \ \mathsf{xalt}\ d'_2]\!]^i$
PROOF: ⟨1⟩1:1, ⟨1⟩1:2, definition 22 of $\rightsquigarrow_{rl}$ and theorem 21 (monotonicity of $\rightsquigarrow_{rg}$ with respect to xalt).
⟨2⟩2. Choose arbitrary $o' \in [\![d'_1 \ \mathsf{xalt}\ d'_2]\!]^i$
PROOF: $[\![d]\!]^i$ is non-empty for all interactions d.
⟨2⟩3. $\exists o \in [\![d_1 \ \mathsf{xalt}\ d_2]\!]^i : o \rightsquigarrow_{rr} o'$
⟨3⟩1. CASE: $o' \in [\![d'_1]\!]^i$
⟨4⟩1. Choose $o \in [\![d_1]\!]^i$ such that $o \rightsquigarrow_{rr} o'$
PROOF: ⟨2⟩2, ⟨1⟩1:1 and definition 22 of $\rightsquigarrow_{rl}$.
⟨4⟩2. $o \in [\![d_1 \ \mathsf{xalt}\ d_2]\!]^i$
PROOF: ⟨4⟩1 and definition 18 of xalt.
⟨4⟩3. Q.E.D.
⟨3⟩2. CASE: $o' \in [\![d'_2]\!]^i$
⟨4⟩1. Choose $o \in [\![d_2]\!]^i$ such that $o \rightsquigarrow_{rr} o'$
PROOF: ⟨2⟩2, ⟨1⟩1:2 and definition 22 of $\rightsquigarrow_{rl}$.
⟨4⟩2. $o \in [\![d_1 \ \mathsf{xalt}\ d_2]\!]^i$
PROOF: ⟨4⟩1 and definition 18 of xalt.
⟨4⟩3. Q.E.D.
⟨3⟩3. Q.E.D.
PROOF: The cases are exhaustive by ⟨2⟩2 and definition 18 of xalt.
⟨2⟩4. $\forall o' \in [\![d'_1 \ \mathsf{xalt}\ d'_2]\!]^i : \exists o \in [\![d_1 \ \mathsf{xalt}\ d_2]\!]^i : o \rightsquigarrow_{rr} o'$
PROOF: ⟨2⟩2, ⟨2⟩3 and ∀-rule.
⟨2⟩5. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩4 and definition 22 of $\rightsquigarrow_{rl}$.
⟨1⟩2. Q.E.D.
PROOF: ⇒-rule.

□

B.3 Correspondence

Lemma 5. *Let (p, n) be an interaction obligation for the sequence diagram d, I be a system and h a well-formed trace.*

ASSUME: 1. $(p, n) \rightsquigarrow_r (traces(I), \mathcal{H}^{ll(d)} \setminus traces(I))$
2. $h \in traces(I)$
PROVE: $(p, n) \mapsto_r (\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\})$,
i.e. $(p, n) \rightsquigarrow_r (\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\})$ by definition 14.
PROOF.

⟨1⟩1. Requirement 1: $n \subseteq \mathcal{H}^{ll(d)} \setminus \{h\}$
⟨2⟩1. $n \subseteq \mathcal{H}^{ll(d)} \setminus traces(I)$
PROOF: Assumption 1 and definition 11 of $\rightsquigarrow_r$.
⟨2⟩2. $h \in traces(I)$
PROOF: Assumption 2.
⟨2⟩3. Q.E.D.
PROOF: ⟨2⟩1, ⟨2⟩2 and $A \subseteq B \setminus X \wedge x \in X \Rightarrow A \subseteq B \setminus \{x\}$ for arbitrary sets A, B and X.
⟨1⟩2. Requirement 2: $p \subseteq \{h\} \cup (\mathcal{H}^{ll(d)} \setminus \{h\})$, i.e. $p \subseteq \{h\} \cup \mathcal{H}^{ll(d)}$
PROOF: $p \subseteq \mathcal{H}^{ll(d)}$ by definition of $\mathcal{H}^{ll(d)}$, as (p, n) is an interaction obligation for d.
⟨1⟩3. Q.E.D.
PROOF: ⟨1⟩1, ⟨1⟩2 and definition 11 of $\rightsquigarrow_r$.

□

Lemma 6. *Let (p, n) be an interaction obligation for the sequence diagram d, I be a system and h a well-formed trace.*

ASSUME: $\forall h \in traces(I) : (p, n) \rightsquigarrow_r (\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\})$
PROVE: $(p, n) \mapsto_r (traces(I), \mathcal{H}^{ll(d)} \setminus traces(I))$,
i.e. $(p, n) \rightsquigarrow_r (traces(I), \mathcal{H}^{ll(d)} \setminus traces(I))$ by definition 14.
PROOF.

⟨1⟩1. Requirement 1: $n \subseteq \mathcal{H}^{ll(d)} \setminus traces(I)$
⟨2⟩1. $\forall h \in traces(I) : n \subseteq \mathcal{H}^{ll(d)} \setminus \{h\}$
PROOF: The assumption and definition 11 of $\rightsquigarrow_r$.
⟨2⟩2. $\forall h \in traces(I) : \{h\} \cap n = \emptyset$
PROOF: ⟨2⟩1 and definition of $\subseteq$.
⟨2⟩3. $traces(I) \cap n = \emptyset$
PROOF: ⟨2⟩2 and $X \cap A = \emptyset \wedge Y \cap A = \emptyset \Rightarrow (X \cup Y) \cap A = \emptyset$ for arbitrary sets A, X and Y.
⟨2⟩4. $n \subseteq \mathcal{H}^{ll(d)}$
PROOF: Definition of $\mathcal{H}^{ll(d)}$, as (p, n) is an interaction obligation for d.
⟨2⟩5. Q.E.D.
PROOF: ⟨2⟩3, ⟨2⟩4 and $A \subseteq B \wedge A \cap X = \emptyset \Rightarrow A \subseteq B \setminus X$ for arbitrary sets A, B and X.
⟨1⟩2. Requirement 2: $p \subseteq traces(I) \cup (\mathcal{H}^{ll(d)} \setminus traces(I))$, i.e. $p \subseteq traces(I) \cup \mathcal{H}^{ll(d)}$

PROOF: $p \subseteq \mathcal{H}^{ll(d)}$ by definition of $\mathcal{H}^{ll(d)}$, as (p, n) is an interaction obligation for d.

⟨1⟩3. Q.E.D.
PROOF: ⟨1⟩1, ⟨1⟩2 and definition 11 of $\rightsquigarrow_r$.

□

Theorem 27. (Correspondence.) *Let d be a a sequence diagram with no* xalt *operator. Then*

$$[\![d]\!]^u \mapsto_r \langle I \rangle_d^u \Rightarrow [\![d]\!]^i \mapsto_g \langle I \rangle_d^i$$

PROOF.

LET: $[\![d]\!]^u = (p, n)$,
i.e. $[\![d]\!]^i = \{(p, n)\}$

⟨1⟩1. ASSUME: $[\![d]\!]^u \mapsto_r \langle I \rangle_d^u$
PROVE: $[\![d]\!]^i \mapsto_g \langle I \rangle_d^i$
⟨2⟩1. $(p, n) \mapsto_r (traces(I), \mathcal{H}^{ll(d)} \setminus traces(I))$
PROOF: ⟨1⟩1, $[\![d]\!]^u = (p, n)$ and definition 13 of $\langle I \rangle_d^u$.
⟨2⟩2. $(p, n) \rightsquigarrow_r (traces(I), \mathcal{H}^{ll(d)} \setminus traces(I))$
PROOF: ⟨2⟩1 and definition 14 of $\mapsto_r$
⟨2⟩3. Choose arbitrary $h \in traces(I)$
PROOF: $traces(I)$ is non-empty for all real systems I.
⟨2⟩4. $(p, n) \mapsto_r (\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\})$
PROOF: ⟨2⟩2, ⟨2⟩3 and lemma 5.
⟨2⟩5. $(\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\}) \in \langle I \rangle_d^i$
PROOF: ⟨2⟩3 and definition 23 of $\langle I \rangle_d^i$.
⟨2⟩6. $\forall o \in [\![d]\!]^i : \exists o' \in \langle I \rangle_d^i : o \mapsto_r o'$
PROOF: $[\![d]\!]^i = \{(p, n)\}$, ⟨2⟩4, ⟨2⟩5 and ∀-rule.
⟨2⟩7. Q.E.D.
PROOF: ⟨2⟩6 and definition 24 of $\mapsto_g$.
⟨1⟩2. Q.E.D.
PROOF: ⇒-rule.

□

Theorem 28. (Correspondence.) *Let d be a a sequence diagram with no* xalt *operator. Then*

$$[\![d]\!]^u \mapsto_{rr} \langle I \rangle_d^u \Rightarrow [\![d]\!]^i \mapsto_{rg} \langle I \rangle_d^i$$

PROOF.

LET: $[\![d]\!]^u = (p, n)$,
i.e. $[\![d]\!]^i = \{(p, n)\}$

⟨1⟩1. ASSUME: $[\![d]\!]^u \mapsto_{rr} \langle I \rangle_d^u$
PROVE: $[\![d]\!]^i \mapsto_{rg} \langle I \rangle_d^i$
⟨2⟩1. $(p, n) \mapsto_{rr} (traces(I), \mathcal{H}^{ll(d)} \setminus traces(I))$

PROOF: $\langle 1\rangle 1$, $[\![d]\!]^u = (p, n)$ and definition 13 of $\langle I\rangle_d^u$.

$\langle 2\rangle 2$. $(p, n) \rightsquigarrow_r (traces(I), \mathcal{H}^{ll(d)} \setminus traces(I))$

PROOF: $\langle 2\rangle 1$ and definitions 14 and 15 of $\mapsto_{rr}$.

$\langle 2\rangle 3$. $p \cap traces(I) \neq \emptyset$

PROOF: $\langle 2\rangle 1$ and definition 15 of $\mapsto_{rr}$.

$\langle 2\rangle 4$. Choose $h \in traces(I)$ such that $p \cap \{h\} \neq \emptyset$

PROOF: $\langle 2\rangle 3$.

$\langle 2\rangle 5$. $(p, n) \mapsto_r (\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\})$

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

$\langle 2\rangle 6$. $(\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\}) \in \langle I\rangle_d^i$

PROOF: $\langle 2\rangle 4$ and definition 23 of $\langle I\rangle_d^i$.

$\langle 2\rangle 7$. $\forall o \in [\![d]\!]^i : \exists o' \in \langle I\rangle_d^i : o \mapsto_{rr} o'$

PROOF: $[\![d]\!]^i = \{(p, n)\}$, $\langle 2\rangle 4$, $\langle 2\rangle 5$, $\langle 2\rangle 6$ and $\forall$-rule.

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

PROOF: $\langle 2\rangle 7$ and definition 24 of $\mapsto_{rg}$.

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

PROOF: $\Rightarrow$-rule.

□

Theorem 29. (Correspondence.) *Let d be a a sequence diagram with no* xalt *operator. Then*

$$[\![d]\!]^u \mapsto_r \langle I\rangle_d^u \Rightarrow [\![d]\!]^i \mapsto_l \langle I\rangle_d^i$$

PROOF.

LET: $[\![d]\!]^u = (p, n)$,
i.e. $[\![d]\!]^i = \{(p, n)\}$

$\langle 1\rangle 1$. ASSUME: $[\![d]\!]^u \mapsto_r \langle I\rangle_d^u$
PROVE: $[\![d]\!]^i \mapsto_l \langle I\rangle_d^i$

$\langle 2\rangle 1$. $[\![d]\!]^i \mapsto_g \langle I\rangle_d^i$

PROOF: $\langle 1\rangle 1$ and theorem 27 (correspondence between $\mapsto_r$ and $\mapsto_g$).

$\langle 2\rangle 2$. $(p, n) \mapsto_r (traces(I), \mathcal{H}^{ll(d)} \setminus traces(I))$

PROOF: $\langle 1\rangle 1$, $[\![d]\!]^u = (p, n)$ and definition 13 of $\langle I\rangle_d^u$.

$\langle 2\rangle 3$. $(p, n) \rightsquigarrow_r (traces(I), \mathcal{H}^{ll(d)} \setminus traces(I))$

PROOF: $\langle 2\rangle 2$ and definition 14 of $\mapsto_r$

$\langle 2\rangle 4$. Choose arbitrary $o' \in \langle I\rangle_d^i$

PROOF: $\langle I\rangle_d^i$ is non-empty for all real systems I.

$\langle 2\rangle 5$. Choose $h \in traces(I)$ such that $o' = (\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\})$

PROOF: $\langle 2\rangle 4$ and definition 23 of $\langle I\rangle_d^i$.

$\langle 2\rangle 6$. $(p, n) \mapsto_r (\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\})$

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

$\langle 2\rangle 7$. $\forall o' \in \langle I\rangle_d^i : \exists o \in [\![d]\!]^i : o \mapsto_r o'$

PROOF: $[\![d]\!]^i = \{(p, n)\}$, $\langle 2\rangle 4$, $\langle 2\rangle 5$, $\langle 2\rangle 6$ and $\forall$-rule.

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

PROOF: $\langle 2\rangle 1$, $\langle 2\rangle 7$ and definition 25 of $\mapsto_l$.

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

PROOF: ⇒-rule.

□

Theorem 30. (Correspondence.) *Let d be a a sequence diagram with no* xalt *operator. Then*

$$[\![\ d\ ]\!]^u \mapsto_r \langle I \rangle_d^u \Leftarrow [\![\ d\ ]\!]^i \mapsto_l \langle I \rangle_d^i$$

PROOF.

LET: $[\![\ d\ ]\!]^i = \{(p,n)\}$,
i.e.$[\![\ d\ ]\!]^u = (p,n)$
⟨1⟩1. ASSUME: $[\![\ d\ ]\!]^i \mapsto_l \langle I \rangle_d^i$
PROVE: $[\![\ d\ ]\!]^u \mapsto_r \langle I \rangle_d^u$
⟨2⟩1. $\{(p,n)\} \mapsto_l \{(\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\}) \mid h \in traces(I)\}$
PROOF: ⟨1⟩1, $[\![\ d\ ]\!]^i = \{(p,n)\}$ and definition 23 of $\langle I \rangle_d^i$.
⟨2⟩2. $\forall h \in traces(I) : (p,n) \mapsto_r (\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\})$
PROOF: ⟨2⟩1 and definition 25 of $\mapsto_l$.
⟨2⟩3. $\forall h \in traces(I) : (p,n) \rightsquigarrow_r (\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\})$
PROOF: ⟨2⟩2 and definition 14 of $\mapsto_r$.
⟨2⟩4. $(p,n) \mapsto_r (traces(I), \mathcal{H}^{ll(d)} \setminus traces(I))$
PROOF: ⟨2⟩3 and lemma 6.
⟨2⟩5. Q.E.D.
PROOF: ⟨2⟩4, $[\![\ d\ ]\!]^u = (p,n)$ and definition 13 of $\langle I \rangle_d^u$.
⟨1⟩2. Q.E.D.
PROOF: ⇐-rule.

□

Theorem 31. (Correspondence.) *Let d be a a sequence diagram with no* xalt *operator. Then*

$$[\![\ d\ ]\!]^u \mapsto_{rr} \langle I \rangle_d^u \Leftarrow [\![\ d\ ]\!]^i \mapsto_{rl} \langle I \rangle_d^i$$

PROOF.

LET: $[\![\ d\ ]\!]^i = \{(p,n)\}$,
i.e.$[\![\ d\ ]\!]^u = (p,n)$
⟨1⟩1. ASSUME: $[\![\ d\ ]\!]^i \mapsto_{rl} \langle I \rangle_d^i$
PROVE: $[\![\ d\ ]\!]^u \mapsto_{rr} \langle I \rangle_d^u$
⟨2⟩1. $\{(p,n)\} \mapsto_{rl} \{(\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\}) \mid h \in traces(I)\}$
PROOF: ⟨1⟩1, $[\![\ d\ ]\!]^i = \{(p,n)\}$ and definition 23 of $\langle I \rangle_d^i$.
⟨2⟩2. $\forall h \in traces(I) : (p,n) \mapsto_{rr} (\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\})$
PROOF: ⟨2⟩1 and definition 25 of $\mapsto_{rl}$.
⟨2⟩3. $\forall h \in traces(I) : (p,n) \rightsquigarrow_r (\{h\}, \mathcal{H}^{ll(d)} \setminus \{h\})$
PROOF: ⟨2⟩2 and definitions 14 and 15 of $\mapsto_{rr}$.
⟨2⟩4. $(p,n) \mapsto_r (traces(I), \mathcal{H}^{ll(d)} \setminus traces(I))$
PROOF: ⟨2⟩3 and lemma 6.
⟨2⟩5. $\forall h \in traces(I) : p \cap \{h\} \neq \emptyset$
PROOF: ⟨2⟩2 and definition 15 of $\mapsto_{rr}$.

⟨2⟩6. $p \cap traces(I) \neq \emptyset$
PROOF: ⟨2⟩5
⟨2⟩7. $(p, n) \mapsto_{rr} (traces(I), \mathcal{H}^{ll(d)} \setminus traces(I))$
PROOF: ⟨2⟩4, ⟨2⟩6 and definition 15 of $\mapsto_{rr}$.
⟨2⟩8. Q.E.D.
PROOF: ⟨2⟩7, $[\![d]\!]^u = (p, n)$ and definition 13 of $\langle I \rangle_d^u$.
⟨1⟩2. Q.E.D.
PROOF: $\Leftarrow$-rule.

□