University of Oslo
Department of Informatics

Specification and Refinement of Soft Real-time Requirements Using Sequence Diagrams

Atle Refsdal,
Knut Eilif Husa,
Ketil Stølen

Technical report 323

ISBN 82-7368-276-5
ISSN 0806-3036

April 10, 2007

Specification and Refinement of Soft Real-time Requirements Using Sequence Diagrams

Atle Refsdal[1], Knut Eilif Husa[1,2], Ketil Stølen[1,3]

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

Abstract. Soft real-time requirements are often related to communication in distributed systems. Therefore it is interesting to understand how UML sequence diagrams can be used to specify such requirements. We propose a way of integrating soft real-time requirements in sequence diagram specifications by adding probabilities to timed sequence diagrams. Our approach builds on timed STAIRS, which is an approach to the compositional and incremental development of sequence diagrams supporting specification of mandatory as well as potential behavior.

1 Introduction

A soft real-time requirement is a time requirement that needs to be met only by a certain percentage of the relevant behavior. A hard real-time requirement can be seen as a special case of a soft real-time requirement; it is a soft real-time requirement that needs to be met in 100% of the cases. When a delay depends on factors that are hard to measure, highly complex or outside our control, a soft real-time requirement is often more appropriate than a hard constraint.
Time constraints are often related to some kind of communication scenario. Therefore it is important to be able to express soft real-time constraints in sequence diagrams. Sequence diagrams show how a task is performed by sending messages between lifelines.
In this paper we enable specification of soft real-time constraints with sequence diagrams by extending STAIRS presented in [HS03], [HHRS06] and [HHRS05] with the possibility of assigning probabilities. The probabilities are added independently from the time constraints, so our approach supports probabilistic specifications in general.
The rest of this paper is organized as follows: Section 2 introduces a specification to illustrate aspects of probabilistic STAIRS throughout the paper. Section 3 defines events, traces and some basic operators. Timed STAIRS is introduced in section 4, while section 5 discusses the relation between mandatory choice and probabilities. Probabilistic STAIRS is introduced in section 6, and section 7 shows how this enables the addition of a soft real-time requirement to the example specification. In section 8 the refinement relation is defined. Section 9 demonstrates refinement of the example specification. We discuss some related work in section 10 before concluding in section 11.

2 The Automatic Teller Machine Example

We use as example a scenario where a customer withdraws money from an automatic teller machine (atm). This section gives a brief and informal explanation of the example. Figure 1 shows the first version of the specification. It serves two purposes. Firstly, it introduces the basic UML sequence diagram notation. Secondly, it allows us to characterize the need for more expressiveness. We come back to this example in later sections to illustrate our approach. Since our main concern is demonstration of real-time specifications we have omitted some details that would belong in a real-life scenario, such as the entering of a PIN code. The scenario describes the case where the transaction succeeds.

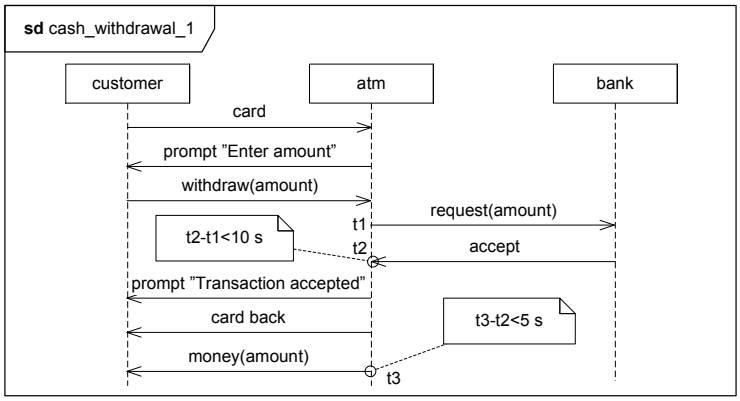

Fig. 1. A cash withdrawal scenario.

It is an interaction between three *lifelines*: the customer, the atm and the bank. Lifelines represent the entities taking part in the interaction. The intuition behind the specification is the following: First the customer inserts her/his card, and the atm displays the text "Enter amount". The customer then enters the desired amount, and the atm sends a request to the bank asking whether the transaction is acceptable. A hard real-time requirement has been placed on the reply from the bank, stating that it should take no more than 10 seconds from the atm sends its request to the reply is received.[4] After the atm receives a positive reply from the bank, it displays the text "Transaction accepted", returns the card, and finally delivers the desired amount of money. A second hard real-time requirement has been put on the delivery of money stating that the time it takes from the atm receives a positive reply from the bank to the money is delivered should be less than five seconds.

UML sequence diagrams describe *traces* representing execution histories, and categorize traces as positive (valid) or negative (invalid). Positive traces represent

[4] We have chosen to use a different notation for real-time requirements than in UML 2.0, since we find our notation more suitable when the requirement crosses an operator boundary, as will happen in later specifications. Graphical (concrete) syntax is not a main issue in this paper.

acceptable executions, while negative traces represent unacceptable executions. All other traces are inconclusive, meaning that the specification does not say whether they are acceptable [OMG04, p. 526]. According to the specification in Figure 1, the positive traces are those where 1) messages are sent in the order shown in the diagram and 2) both real-time requirements are fulfilled. The negative traces are those that fulfill 1) but not 2).
The delay from the request is sent from the atm to a reply is received may depend on several complex factors, so we might want to replace the hard real-time requirement with a soft real-time requirement. Timed STAIRS gives a formal semantics to (a subset of) UML sequence diagrams with hard real-time requirements. Specifying soft real-time constraints, however, is not possible. Enabling the specification of soft real-time requirements within the framework of timed STAIRS is the aim of this paper.

3 Events, Traces and Basic Operators

In this section we define the notions of events and traces. We also introduce a number of useful operators. Most of the definitions and explanations in this section are taken from [HHRS06].
For any set A, A^ω denotes the set of finite as well as infinite sequences of elements of A. $\mathbb{N}$ denotes the set of natural numbers, while $\mathbb{N}_0$ denotes the set of natural numbers including 0. We define the functions

$$\#\_ \in A^\omega \to \mathbb{N}_0 \cup \{\infty\}, \quad \_[\_] \in A^\omega \times \mathbb{N} \to A, \quad \_\frown\_ \in A^\omega \times A^\omega \to A^\omega,$$
$$\_|\_ \in A^\omega \times \mathbb{N}_0 \to A^\omega, \quad \_ⓢ\_ \in \mathbb{P}(A) \times A^\omega \to A^\omega$$

to yield the length of a sequence, the nth element of a sequence, the concatenation of two sequences, truncation of a sequence and the filtering of a sequence. Hence, $\#a$ yields the number of elements in a, and $a[n]$ yields a's nth element if $n \leq \#a$. To concatenate two sequences means to glue them together. Therefore, $a_1 \frown a_2$ denotes a sequence of length $\#a_1 + \#a_2$ that equals a_1 if a_1 is infinite, and is prefixed by a_1 and suffixed by a_2 otherwise. For any $0 \leq i \leq \#a$, $a|_i$ denotes the prefix of a of length i. By $B ⓢ a$ we denote the sequence obtained from the sequence a by removing all elements in a that are not in the set B.
We also need filtering of pairs of sequences. The filtering function

$$\_ⓣ\_ \in \mathbb{P}(A \times B) \times (A^\omega \times B^\omega) \to A^\omega \times B^\omega$$

can be understood as a generalization of ⓢ. For any set of pairs of elements P and pairs of sequences t, $P ⓣ t$ denotes the pair of sequences obtained from t by

- truncating the longest sequence in t at the length of the shortest sequence in t if the two sequences are not of equal length;
- for each $j \in [1...k]$, where k is the length of the shortest sequence in t, selecting or deleting the two elements at index j in the two sequences, depending on whether the pair of these elements is in the set P.

For example, we have that

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

For a formal definition of $\circledcirc$, see [BS01].

3.1 Events

A message is a triple (s, re, tr) of a signal s, a receiver re and a transmitter tr. $\mathcal{M}$ denotes the set of all messages. The receiver and transmitter are lifelines. $\mathcal{L}$ denotes the set of all lifelines.
An event may be of two kinds; a transmission event tagged by "!" or a reception event tagged by "?".[5] Every event occurring in a sequence diagram has a timestamp tag. $\mathcal{T}$ denotes the set of timestamp tags. We use logical formulas with timestamp tags as free variables to impose constraints on the timing of events. By $\mathbb{F}(v)$ we denote the set of logical formulas whose free variables are contained in the set of timestamp tags v.
An event is a triple $(k, m, t) \in \{!, ?\} \times \mathcal{M} \times \mathcal{T}$ of a kind, a message and a timestamp tag. $\mathcal{E}$ denotes the set of all events. We define the functions

$$k\_ \in \mathcal{E} \to \{?, !\},\ m\_ \in \mathcal{E} \to \mathcal{M},\ t\_ \in \mathcal{E} \to \mathcal{T},\ tr\_ \in \mathcal{E} \to \mathcal{L},\ re\_ \in \mathcal{E} \to \mathcal{L}$$

to yield the kind, message, timestamp tag, transmitter and receiver of an event, respectively. Since we are primarily interested in communication scenarios, we do not give a semantic interpretation to events, except that the timestamp tag is assigned a timestamp in form of a real number. $\mathbb{R}$ denotes the set of timestamps. The set $[\![\ \mathcal{E}\ ]\!]$ of event interpretations is therefore defined by

$$[\![\ \mathcal{E}\ ]\!] \stackrel{\mathsf{def}}{=} \{(k, m, t \mapsto r) \mid (k, m, t) \in \mathcal{E} \wedge r \in \mathbb{R}\} \tag{1}$$

$t \mapsto r$ means that timestamp r is assigned to timestamp tag t. We also define the function

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

to yield the timestamp of an event interpretation. In the following, we use "event" and "event interpretation" interchangeably.

3.2 Traces

A trace $h \in [\![\ \mathcal{E}\ ]\!]^\omega$ is a finite or infinite sequence of events. Traces represent executions of the system under specification, and must satisfy a number of well-formedness conditions. Firstly, we require the events of h to be ordered by time:

$$\forall\, i, j \in [1..\#h] : i < j \Rightarrow r.h[i] \leq r.h[j] \tag{2}$$

[5] Note that in timed STAIRS [HHRS06] "?" represents consumption. We have chosen to use "?" for reception since we do not concider consumption events in this paper.

Note that two events may occur at the same time.
Secondly, we allow the same event to occur only once in the same trace:

$$\forall\, i, j \in [1..\#h] : i \neq j \Rightarrow h[i] \neq h[j] \tag{3}$$

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

$$\forall\, l \in \mathcal{L} : (\#(e.l \circledS h) = \infty \Rightarrow \forall\, t \in \mathbb{R} : \exists\, i \in \mathbb{N} : r.(e.l \circledS h)[i] > t) \tag{4}$$

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

$$e.l \stackrel{\mathsf{def}}{=} \{e \in [\![\ \mathcal{E}\ ]\!] \mid (k.e =! \wedge tr.e = l) \vee (k.e =? \wedge re.e = l)\} \tag{5}$$

We also require that for any single message, transmission happens before reception. But we need to take into account that the transmitter or receiver of a certain message might not be included in the sequence diagram. Thus we get the following well-formedness requirement on traces, stating that if at any point in the trace we have a transmission event, up to that point we must have had at least as many transmissions as receptions of that particular message:

$$\forall\, i \in [1..\#h] : k.h[i] =! \Rightarrow \tag{6}$$
$$\#(\{!\} \times \{m.h[i]\} \times U) \circledS h|_i > \#(\{?\} \times \{m.h[i]\} \times U) \circledS h|_i$$

where $U \stackrel{\mathsf{def}}{=} \{t \mapsto r \mid t \in \mathcal{T} \wedge r \in \mathbb{R}\}$.
$\mathcal{H}$ denotes the set of well-formed traces. Traces are written as a sequence of events enclosed by the brackets $\langle \ldots \rangle$, for example $\langle e_1, e_2, e_3 \rangle$.

4 Syntax and Semantics for Timed STAIRS

In the following we explain how a timed sequence diagram can be represented semantically by a *interaction obligation* (p, n) where p is a set of positive traces and n is a set of negative traces. (This is a simplification of timed STAIRS, where a sequence diagram is represented by a set of interaction obligations.) $\mathcal{O}$ denotes the set of interaction obligations. An interaction obligation (p, n) is contradictory if $p \cap n \neq \varnothing$. $[\![\ d\ ]\!]$ denotes the interaction obligation representing sequence diagram d.

4.1 Textual Syntax for Timed Sequence Diagrams

The set of syntactically correct sequence diagrams, $\mathcal{D}$, is defined inductively as the least set such that: [6]

[6] In Timed STAIRS [HHRS06] seq is defined as an n-ary operator instead of binary, and the operators loop, assert and xalt are also included. loop and assert for probabilistic STAIRS are introduced in section A, while the xalt is replaced by palt, which is introduced in Section 6.

– $\mathcal{E} \subset \mathcal{D}$
– $d \in \mathcal{D} \Rightarrow \mathsf{neg}\ d \in \mathcal{D}$
– $d_1, d_2 \in \mathcal{D} \Rightarrow d_1\ \mathsf{par}\ d_2 \in \mathcal{D} \wedge d_1\ \mathsf{seq}\ d_2 \in \mathcal{D} \wedge d_1\ \mathsf{alt}\ d_2 \in \mathcal{D}$
– $d \in \mathcal{D} \wedge C \in \mathbb{F}(tt.d) \Rightarrow d\ \mathsf{tc}\ C \in \mathcal{D}$

where $tt.d$ yields the set of timestamp tags occurring in d. The base case implies that any event is a sequence diagram. Any other sequence diagram is constructed from the basic ones through the application of operations for negation, potential choice (alternative), weak sequencing, parallel execution and time constraint.
Only sequence diagrams that are syntactically correct in UML 2.0 are considered. Also, extra global combined fragments are not handled. This means that for all operators except for seq and par it is assumed that every operand includes only complete message communications, i.e. messages where both the transmission and the reception event is within the same operand. Formally, for every operand d_i of an operator different from seq and par we require:

$$\forall\, m \in msg.d_i : \tag{7}$$
$$\#\{\!\{ e \in ev.d_i \mid k.e =! \wedge m.e = m \}\!\} = \#\{\!\{ e \in ev.d_i \mid k.e =? \wedge m.e = m \}\!\}$$

where $\{\!\{\ \ \}\!\}$ denotes a multiset and $\#$ is overloaded to yield the number of elements in a multiset. The functions

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

yield the messages, events and lifelines of a sequence diagram, respectively.
All single-event diagrams are considered syntactically correct. For all diagrams consisting of more than one event, it is required that a message is complete if both the transmitter and the receiver lifelines are present in the diagram:

$$\forall\, m \in msg.d_i : (\#ev.d > 1 \wedge tr.m \in ll.d \wedge re.m \in ll.d) \Rightarrow \tag{8}$$
$$\#\{\!\{ e \in ev.d \mid k.e =! \wedge m.e = m \}\!\} = \#\{\!\{ e \in ev.d \mid k.e =? \wedge m.e = m \}\!\}$$

4.2 Denotational Semantics for Timed STAIRS

Event The semantics of an event is the interaction obligation whose positive set consists of infinitely many unary positive traces – one for each possible assignment of a timestamp to its timestamp tag. The negative set is empty.

$$[\![\ (k, m, t)\ ]\!] \stackrel{\mathsf{def}}{=} (\{\langle (k, m, t \mapsto r) \rangle \mid r \in \mathbb{R}\}, \varnothing) \quad \text{if } (k, m, t) \in \mathcal{E} \tag{9}$$

Negation Undesired behavior is defined by the use of the neg construct. To negate a specification means to move every positive trace to the negative set. Negative traces remain negative. The empty trace is defined as positive to enable positive traces in a composition. Negation of a specification is defined by

$$[\![\ \mathsf{neg}\ d\ ]\!] \stackrel{\mathsf{def}}{=} \neg\, [\![\ d\ ]\!] \tag{10}$$

where

$$\neg\, (p, n) \stackrel{\mathsf{def}}{=} (\{\langle\rangle\}, n \cup p) \tag{11}$$

Parallel Execution The operator for parallel execution is represented semantically by $\|$. Ignoring for the time being the sets of negative traces, a parallel execution defines the set of traces we get by merging one trace from one (positive) set with one trace from the other (positive) set. Informally, for sets of traces s_1 and s_2, $s_1 \parallel s_2$ is the set of all traces such that:

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

Formally:

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

where π_i is a projection operator returning element number i of a tuple. In this definition we make use of an oracle, the infinite sequence or, to resolve the nondeterminism in the interleaving. It determines the order in which events from traces in s_1 and s_2 are sequenced.
The semantics of parallel execution may then be defined as

$$[\![\ d_1 \ \mathsf{par}\ d_2\ ]\!] \stackrel{\mathsf{def}}{=} [\![\ d_1\ ]\!] \parallel [\![\ d_2\ ]\!] \tag{13}$$

where

$$(p_1, n_1) \parallel (p_2, n_2) \stackrel{\mathsf{def}}{=} (p_1 \parallel p_2, (n_1 \parallel (p_2 \cup n_2)) \cup (p_1 \parallel n_2)) \tag{14}$$

Note that the merging of a negative trace with another (positive or negative) trace always results in a negative trace.

Weak Sequencing Weak sequencing is the implicit composition mechanism combining constructs of a sequence diagram. The operator for weak sequencing is represented semantically by $\succsim$. We again temporarily ignore the sets of negative traces, and let s_1 and s_2 be trace sets. Since lifelines are independent, the constraint for the ordering of events applies to each lifeline; events that occur on different lifelines are interleaved. For $s_1 \succsim s_2$ we therefore have the constraint that events on one lifeline from one trace in s_1 should come before events from one trace in s_2 on the same lifeline:

$$\begin{aligned} s_1 \succsim s_2 \quad \stackrel{\mathsf{def}}{=} \quad & \{h \in \mathcal{H} \mid \exists\, h_1 \in s_1, h_2 \in s_2 : \\ & \forall\, l \in \mathcal{L} : e.l \circledS h = e.l \circledS h_1 \frown e.l \circledS h_2\} \end{aligned} \tag{15}$$

The semantics of weak sequencing may then be defined as

$$[\![\ d_1 \ \mathsf{seq}\ d_2\ ]\!] \stackrel{\mathsf{def}}{=} [\![\ d_1\ ]\!] \succsim [\![\ d_2\ ]\!] \tag{16}$$

where

$$(p_1, n_1) \succsim (p_2, n_2) \stackrel{\mathsf{def}}{=} (p_1 \succsim p_2, (n_1 \succsim (p_2 \cup n_2)) \cup (p_1 \succsim n_2)) \tag{17}$$

Weak sequencing involving at least one negative trace results in a negative trace.

Time Constraint Time requirements are imposed by the use of a time constraint, denoted by $\wr C$, where C is a predicate over timestamp tags. When a time constraint is applied to a trace set all traces not fulfilling the constraint are removed. Formally, time constraint for a trace set s is defined as

$$s \wr C \stackrel{\mathsf{def}}{=} \{h \in s \mid h \models C\} \tag{18}$$

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

$$h = \langle (k_1, m_1, t_1 \mapsto r_1), (k_2, m_2, t_2 \mapsto r_2), (k_3, m_3, t_3 \mapsto r_3) \rangle \text{ and } C = t_3 < t_1 + 5$$

then $h \models C$ if $r_3 < r_1 + 5$.
To apply a time requirement to a specification means to define failure to meet the requirement as negative behavior. Traces of the operand that are positive and do not fulfill the requirement become negative. The semantics of a time constraint is defined as

$$[\![\ d\ \mathsf{tc}\ C\ ]\!] \stackrel{\mathsf{def}}{=} [\![\ d\ ]\!] \wr C \tag{19}$$

where

$$(p, n) \wr C \stackrel{\mathsf{def}}{=} (p \wr C, n \cup (p \wr \neg\ C)) \tag{20}$$

Potential Choice The alt construct is used to express underspecification by grouping together traces that from the specifier's point of view serve the same purpose. This means that they are seen as equally desirable (for positive traces) or undesirable (for negative traces). For two trace sets where both are positive or both are negative, this can be represented semantically simply by taking the union of the sets. Hence, potential choice corresponds to the pairwise union of the positive sets and the negative sets. Formally, the semantics of the alt is defined by

$$[\![\ d_1\ \mathsf{alt}\ d_2\ ]\!] \stackrel{\mathsf{def}}{=} [\![\ d_1\ ]\!] \uplus [\![\ d_2\ ]\!] \tag{21}$$

where

$$(p_1, n_1) \uplus (p_2, n_2) \stackrel{\mathsf{def}}{=} (p_1 \cup p_2, n_1 \cup n_2) \tag{22}$$

5 Mandatory Choice and Probabilities

In STAIRS the alt operator as formally defined above enables underspecification, what we also refer to as potential choice. Underspecification means to leave some freedom of choice to the developers that will eventually implement (or further refine) the specification. This is for example useful when different design alternatives fulfill a function equally well from the specifier's point of view.
STAIRS supports also the specification of mandatory choice. For this purpose the STAIRS specific xalt operator is used. Mandatory choice means that all alternatives must be possible. It is often needed within security, for example in

relation to information flow [Ros95]. When specifying a password generator, for instance, it is vital that all alternatives remain possible in the final implementation – otherwise in the extreme case we might end up with an implementation that always generates the same password.
Mandatory choice is also useful for other purposes. Sometimes non-determinism is employed to model the behavior of the environment of the system under specification. The mandatory choice operator is then used to represent alternative inputs from the environment that the designer has considered. If some of these alternatives are removed from the final specification, the implementation will not be able to handle the relevant input as intended.
Sometimes an application is non-deterministic by nature, for example in games. If we want to specify a dice, we obviously need to ensure that all alternatives, one through six, are possible outcomes in the implementation.
In probabilistic STAIRS we generalize the xalt operator into an operator for the specification of probabilities called palt. We may then also specify with what probability the different alternatives should occur. In the dice example, the probability of every outcome should be exactly $\frac{1}{6}$. Of course, if an alternative has an exact probability greater than zero, then this alternative must be a possible outcome of a valid implementation. For this reason, probabilistic choice can be viewed as a special case of mandatory choice. This view is consistent with the one presented in [MM99].
If an alternative is assigned a *set* of acceptable probabilities, then this set represents underspecification. Such underspecification is usually present in soft real-time requirements. A specification might say that the probability of a certain delay being less than 10 seconds should be 0.8 or more. This amounts to saying that the set of acceptable probabilities is $[0.8, .., 1.0]$. According to this specification, an implementation that gives a probability of 0.9 is certainly valid; the developer only needs to achieve one of the acceptable probabilities.

6 Syntax and Semantics for Probabilistic STAIRS

In the following we explain how a probabilistic sequence diagram can be represented semantically by a set of *probability obligations* (also called *p-obligations*). A p-obligation $((p, n), Q)$ consists of an interaction obligation (p, n) and a set of probabilities Q, with the following interpretation: The traces implementing (p, n) should occur with a probability greater than or equal to a probability in Q. Only traces in $\mathcal{H} \setminus n$ are allowed to implement (p, n). The probability for these traces may be greater than the values in Q only if some or all of the traces are also positive or inconclusive according to some other p-obligation. $\mathcal{P}$ denotes the set of p-obligations. In probabilistic STAIRS we may have underspecification with respect to traces and with respect to probabilities. Underspecification with respect to traces is captured by the fact that we may choose among the non-negative traces within an interaction obligation. Underspecification with respect to probabilities is modeled by the possibility of selecting among the probabilities within a p-obligation.

6.1 Textual Syntax for Probabilistic Sequence Diagrams

The set of syntactically correct sequence diagrams $\mathcal{D}$ is defined simply by adding the following case to the inductive definition in 4.1:

– $d_1, d_2 \in \mathcal{D} \wedge Q_1, Q_2 \subseteq [0...1] \Rightarrow d_1;Q_1 \ \mathsf{palt}\ d_2;Q_2 \in \mathcal{D}$

6.2 Denotational Semantics for Probabilistic STAIRS

Event Probabilities can be assigned only by the use of the palt. The traces specified by a sequence diagram without occurrences of palt must occur with probability 1 in their relevant context. Therefore the set of probabilities associated with an event is $\{1\}$.

$$[\![\ (k, m, t)\ ]\!] \stackrel{\mathsf{def}}{=} \{((\{\langle (k, m, t \mapsto r)\rangle \mid r \in \mathbb{R}\}, \varnothing), \{1\})\} \quad \text{if } (k, m, t) \in \mathcal{E} \tag{23}$$

Negation and Time Constraint Negation and time constraint are not affected by probabilities. They are defined by

$$[\![\ \mathsf{neg}\ d\ ]\!] \stackrel{\mathsf{def}}{=} \{(\neg\ o, Q) \mid (o, Q) \in [\![\ d\ ]\!]\} \tag{24}$$

$$[\![\ d\ \mathsf{tc}\ C\ ]\!] \stackrel{\mathsf{def}}{=} \{(o \wr C, Q) \mid (o, Q) \in [\![\ d\ ]\!]\} \tag{25}$$

Parallel Execution and Weak Sequencing When executing two specifications in parallel or sequentially, we get the set of p-obligations obtained from choosing one p-obligation from the first and one p-obligation from the second and composing them in parallel or sequentially. Choosing the two p-obligations to be composed is seen as two independent probabilistic choices; therefore the sets of probabilities are multiplied. Formally, parallel execution and weak sequencing is defined by

$$[\![\ d_1\ \mathsf{par}\ d_2\ ]\!] \stackrel{\mathsf{def}}{=} \{(o_1 \parallel o_2, Q_1 * Q_2) \mid (o_1, Q_1) \in [\![\ d_1\ ]\!] \wedge (o_2, Q_2) \in [\![\ d_2\ ]\!]\} \tag{26}$$

$$[\![\ d_1\ \mathsf{seq}\ d_2\ ]\!] \stackrel{\mathsf{def}}{=} \{(o_1 \gtrsim o_2, Q_1 * Q_2) \mid (o_1, Q_1) \in [\![\ d_1\ ]\!] \wedge (o_2, Q_2) \in [\![\ d_2\ ]\!]\} \tag{27}$$

where multiplication of probability sets is defined by

$$Q_1 * Q_2 \stackrel{\mathsf{def}}{=} \{q_1 * q_2 \mid q_1 \in Q_1 \wedge q_2 \in Q_2\} \tag{28}$$

Potential Choice The alt construct captures underspecification with respect to traces. Two sets of p-obligations are combined by taking the pairwise combination of p-obligations from each set. As in timed STAIRS, the $\uplus$ operator is used for combining the interaction obligations. The probabilities are multiplied since the two p-obligations are chosen independently from each other.

$$[\![\ d_1\ \mathsf{alt}\ d_2\ ]\!] \stackrel{\mathsf{def}}{=} \{(o_1 \uplus o_2, Q_1 * Q_2) \mid (o_1, Q_1) \in [\![\ d_1\ ]\!] \wedge (o_2, Q_2) \in [\![\ d_2\ ]\!]\} \tag{29}$$

Probabilistic Choice The palt construct expresses probabilistic choice (and therefore mandatory choice). Before defining the semantics of the palt we introduce the notion of probability decoration. Probability decoration is used to assign the probabilities associated with the operands of a palt. It is defined by

$$[\![\ d;Q'\ ]\!] \stackrel{\mathsf{def}}{=} \{(o, Q * Q') \mid (o, Q) \in [\![\ d\ ]\!]\} \tag{30}$$

We also define addition of probability sets:

$$Q_1 + Q_2 \stackrel{\mathsf{def}}{=} \{\mathsf{min}(q_1 + q_2, 1) \mid q_1 \in Q_1 \wedge q_2 \in Q_2\} \tag{31}$$

The palt operator is meant to describe the probabilistic choice between two alternative operands whose joint probability should add up to one. Formally, the palt is defined by

$$\begin{aligned}
&[\![\ d_1;Q_1 \ \mathsf{palt}\ d_2;Q_2\ ]\!] \stackrel{\mathsf{def}}{=} \qquad (32)\\
&\quad [\![\ d_1;Q_1\ ]\!] \cup [\![\ d_2;Q_2\ ]\!] \cup\\
&\quad \{(\oplus\{po_1, po_2\}, \pi_2.po_1 + \pi_2.po_2) \mid po_1 \in [\![\ d_1;Q_1\ ]\!] \wedge po_2 \in [\![\ d_2;Q_2\ ]\!]\} \cup\\
&\quad \{(\oplus([\![\ d_1;Q_1\ ]\!] \cup [\![\ d_2;Q_2\ ]\!]), \{1\} \cap (Q_1 + Q_2))\}
\end{aligned}$$

The single p-obligation in the set in the last line in 32 requires the probabilities of the two operands to add up to one. If it is impossible to choose one probability from Q_1 and one from Q_2 so that the sum is 1, then the probability set will be empty and the specification is not implementable.
$\oplus$ characterizes the traces allowed by the two operands together: A trace t is positive if it is positive according to at least one p-obligation and not inconclusive according to any; t is negative only if it is negative according to all p-obligations; traces that are inconclusive according to at least one p-obligation remain inconclusive. Formally, the operator $\oplus$ for combining the interaction obligations of a set S of p-obligations into a single interaction obligation is therefore defined by

$$\oplus S \stackrel{\mathsf{def}}{=} ((\bigcup_{((p,n),Q)\in S} p) \cap (\bigcap_{((p,n),Q)\in S} p \cup n), \bigcap_{((p,n),Q)\in S} n) \tag{33}$$

Since $\oplus\{((p, n), Q), ((p, n), Q)\} = (p, n)$, the second line on the right-hand side of definition 32 ensures that if a p-obligation $((p, n), Q)$ occurs in both operands of the palt, then the resulting semantics will contain a p-obligation $((p, n), Q + Q)$. The inclusion of the second line on the right-hand side of definition 32 enables us to define the semantics of a specification as a set of p-obligations instead of as a multiset. Other reasons to include this line is related to refinement and will be explained in section 8.

7 Adding a Soft Real-time Requirement to the Atm

We now replace the first hard real-time requirement in the atm example with a soft real-time requirement. Consider the sequence diagram in Figure 2.

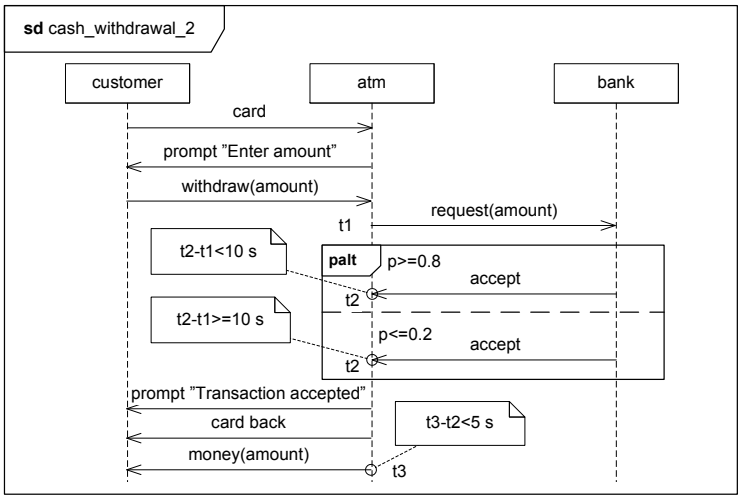

Fig. 2. Cash withdrawal with soft real-time constraint.

This specification is modeled semantically by four p-obligations, we call these po_1, po_2, po_3 and po_4. The result of choosing the first palt operand is modeled semantically by po_1. The positive traces of po_1 are only those in which it takes less than 10 seconds before the reply arrives from the bank and it takes less than five seconds from the reply arrives to the money is delivered. Traces where one or both of these constraints are not met are negative in po_1. The acceptable range of probability for this p-obligation is $[0.8, ..., 1]$.

The result of choosing the second palt operand is modeled semantically by po_2. The positive traces of po_2 are all traces where it takes 10 seconds or more before the reply arrives from the bank and it takes less than five seconds from the reply arrives to the money is delivered. Traces where one or both of these constraints are not met are negative in po_2. The acceptable range of probability for this p-obligation is $[0, ..., 0.2]$.

The third p-obligation, po_3, results from the second line on the right-hand side of definition 32 and models the combination of po_1 and po_2, which means that $po_3 = (\oplus\{po_1, po_2\}, [0.8, ..., 1])$. This means that the positive traces of po_3 are all traces where it takes less than five seconds to get money after the reply is received from the bank, regardless of how long it takes to get the reply. The negative traces are only those where it takes five seconds or more to get the money.

The last p-obligation, po_4, results from the third line on the right-hand side of definition 32 and models the combination of the two operands. Since in this example each operand gives only one p-obligation, this means that the interaction obligation of po_4 is identical to that of po_3, i.e. $po_4 = (\oplus\{po_1, po_2\}, \{1\})$.

Traces where messages are not exchanged between the customer, the atm and the bank as described by Figure 2 (but ignoring the time requirements) are inconclusive according to po_1, po_2, po_3 and po_4.

8 Refinement

Refinement of a specification means to reduce underspecification by adding information so that the specification becomes closer to an implementation. Semantically, in our setting this can be done at the level of p-obligations or at the level of sets of p-obligations. We first define refinement semantically for p-obligations. Then we lift this definition to specifications that are represented semantically by sets of p-obligations.

8.1 Refinement of P-obligations

As in [HHRS05], a interaction obligation is refined by moving positive traces to the set of negative traces or by moving traces from the set of inconclusive traces to either the positive or the negative set. STAIRS [HHRS05] refers to the first option as narrowing and the second option as supplementing. As argued in [HHRS05], narrowing reduces the set of positive traces to capture new design decisions or to match the problem more accurately. Supplementing categorizes (to this point) inconclusive behavior as either positive or negative recognizing that early descriptions normally lack completeness.
A p-obligation is refined by either refining its interaction obligation or reducing its set of probabilities. Formally, a p-obligation $((p', n'), Q')$ is a refinement of a p-obligation $((p, n), Q)$, written $((p, n), Q) \rightsquigarrow ((p', n'), Q')$, iff

$$n \subseteq n' \wedge p \subseteq p' \cup n' \wedge Q' \subseteq Q \tag{34}$$

8.2 Refinement of Specifications

All p-obligations at the given (more abstract) level represent a mandatory alternative. Therefore each p-obligation needs to be represented by a p-obligation also at the refined (more concrete) level. However, if a p-obligation has 0 as an acceptable probability, this means that it does not need to be implemented. Formally, a specification d' is a refinement of a specification d, written $d \rightsquigarrow d'$, iff

$$\forall\, po \in [\![\ d\ ]\!] : 0 \notin \pi_2.po \Rightarrow \exists\, po' \in [\![\ d'\ ]\!] : po \rightsquigarrow po' \tag{35}$$

We now explain further why the second line on the right-hand side of definition 32 is included. Firstly, we want to avoid a situation where two p-obligations $((p_1, n_1), Q_1)$ and $((p_2, n_2), Q_2)$ coming from different operands of a palt are represented only by a single p-obligation at the concrete level. This is ensured since also the p-obligation $(\oplus\{(p_1, n_1), (p_2, n_2)\}, Q_1 + Q_2)$ is included in the semantics and hence needs to be represented at the concrete level.
Secondly, it should be possible to let a single p-obligation at the abstract level be represented by a combination of p-obligations at the concrete level, as long as each of these p-obligations are valid refinements of the original p-obligation with respect to interaction obligations and their probability sets add up to a subset of the original probability set. Since the only way to introduce more p-obligations is to use the palt-operator, the inclusion of the combined p-obligations in the palt semantics makes this possible.

9 Refining the Atm Specification

Figure 3 shows a refinement of the specification in Figure 2.

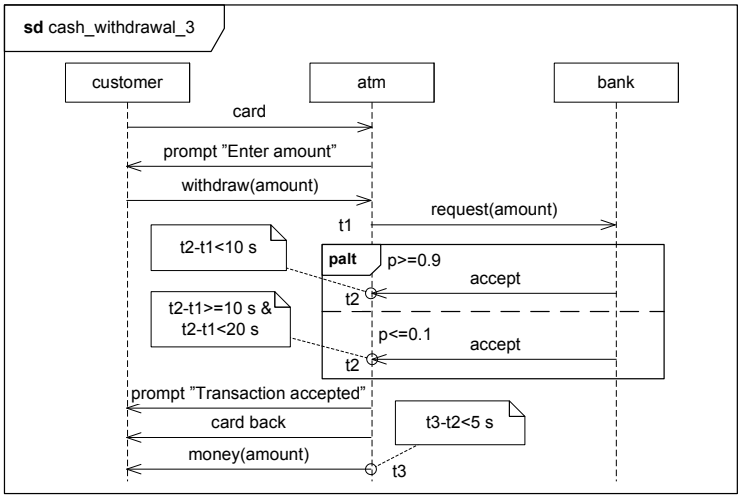

Fig. 3. A refinement of Figure 2.

The change that has been made to "cash_withdrawal_2" is to impose an upper limit to the acceptable response time from the bank also in the second operand, stating that the reply should be received within 20 seconds. In addition we have narrowed the acceptable range of probability for both operands. It is now required that the reply from the bank should be received within 10 seconds in at least 90% of the cases, instead of just 80%.
The specification "cash_withdrawal_3" is modeled semantically by four p-obligations, we call these po_1', po_2', po_3' and po_4'. The p-obligation po_1' represents the result of choosing the first operand of the palt. The positive and negative traces of po_1' are the same as for po_1, while the set of acceptable probabilities for po_1' is $[0.9, ..., 1]$, which is a subset of the probability set of po_1. This means that $po_1 \rightsquigarrow po_1'$.
The result of choosing the second palt operand is modeled semantically by po_2'. The positive and negative traces of po_2' are the same as for po_2, except that traces where it takes more than 20 seconds to get a reply from the bank are positive in po_2 and negative in po_2'. Since the probability set of po_2', $[0, ..., 0.1]$, is a subset of the probability set of po_2, we get $po_2 \rightsquigarrow po_2'$.
The third p-obligation, po_3', models the combination of po_1' and po_2', which means that $po_3' = (\oplus\{po_1', po_2'\}, [0.9, ..., 1])$. According to po_3' the positive traces are all traces where it takes less than 20 seconds to get an answer from the bank and less than five seconds to get money after the reply is received from the bank. The negative traces are those where it takes 20 seconds or more to get a reply or five seconds or more to get the money. Since $[0.9, ..., 1] \subseteq [0.8, ..., 1]$ and the only difference with respect to traces is that the traces where it takes 20 seconds

or more to get a reply from the bank are positive in po_3 and negative in po'_3, we get $po_3 \rightsquigarrow po'_3$.
The last p-obligation, po'_4, models the combination of the two operands. Since in this example each operand gives only one p-obligation, this means that the interaction obligation of po'_4 is identical to that of po'_3, i.e. $po'_4 = (\oplus\{po_1, po_2\}, \{1\})$. Since the probability sets of po_4 and po'_4 are both $\{1\}$ and the only difference with respect to traces is that the traces where it takes 20 seconds or more to get a reply from the bank are positive in po_4 and negative in po'_4, we get $po_4 \rightsquigarrow po'_4$. The above shows that condition 35 is fulfilled, so the specification "cash_withdrawal_3" is a refinement of "cash_withdrawal_2".
We also have that the original specification "cash_withdrawal_1" with its hard real-time constraint is a refinement of "cash_withdrawal_2". To see this, note that the specification "cash_withdrawal_1" is represented semantically by $\{(\pi_1.po_1, \{1\})\}$, and that $(\pi_1.po_1, \{1\})$ is a valid refinement of both po_1, po_3 and po_4. The p-obligation po_2 is not represented in "cash_withdrawal_1", but this is not required, since $0 \in \pi_2.po_2$. Therefore condition 35 is fulfilled, so that "cash_withdrawal_1" is a refinement of "cash_withdrawal_2". A similar argument shows that "cash_withdrawal_1" is also a refinement of "cash_withdrawal_3".

10 Related Work

[Seg95] uses probabilistic automata to address the problem of verification of randomized distributed algorithms. The analysis includes timed systems, so that real-time properties can be investigated in a probabilistic setting. [Jan03] introduces a stochastic extension to statecharts called StoCharts to allow the quantification of the time between events according to a stochastic distribution, and defines a formal semantics that can be analyzed by tools. [JL91] presents a formalism for specifying probabilistic transition systems where transitions have sets of allowed probabilities, and defines two refinement relations on such systems. These formalisms address many of the same issues as we do, but rely on complete specifications of the communicating entities since the models are automata and statecharts.
Various dialects of sequence diagrams have been used informally for several decades. The latest versions of the most known variants are UML 2.0 [OMG04] and MSC-2000 [ITU99].
Live Sequence Charts [DH01], [HM03] is an extension of MSC where (a part of) a chart may be designated as universal (mandatory) or existential (optional). Explicit criteria in the form of pre-charts are given for when a chart applies: Whenever the system exhibits the communication behavior of its pre-chart its own behavior must conform to that prescribed by the chart. Timing constraints are included and alternatives may be assigned exact probabilities.
The UML Profile for Schedulability, Performance and Time [OMG05] extends UML by adding stereotypes and annotations for defining values for performance measures such as response time and CPU demand time. The profile is envisaged to be used with a suitable modeling tool based on for example schedulability

analysis, Petri Nets or stochastic process algebra. The profile enables specification of a wide range of time-related requirements, including soft real-time requirements. However, no formal semantics is defined for the language.
Most closely related to the work presented in this paper is of course timed STAIRS as presented in [HHRS06]. Here the notions of positive and negative behavior, mandatory choice and refinement are formalized in relation to sequence diagrams. Timed STAIRS has a more fine-grained analysis of refinement than presented here. This is partly due to a richer semantical model for events and traces. Events in timed STAIRS can be of three different types: transmit, receive and consume. This enables the distinction between two forms of refinement: glass-box refinement, which take the full semantics into account, and black box refinement, which only considers externally visible changes. The approach presented in this paper can easily be generalized to take this into account. Timed STAIRS does not address probabilities.

11 Conclusion

We have extended the work presented in [HHRS06]. Our contribution is to generalize the approach to handle probabilities. This enables specification of soft real-time constraints as well as probabilistic specifications in general. The resulting approach, which we call probabilistic STAIRS, offers a powerful language for specifying a wide range of communicating systems, underpinned by a formal semantics that allows analysis of functional and non-functional properties, as well as formal definition of incremental development. In the future we intend to explore the relationship between probabilistic STAIRS and state machines with time and probabilities.
This technical report is an extended and revised version of the article [RHS05], and the contents of the report follows this article closely up to and including section 12. For a discussion of changes that have been made, see section C.
In section B we show that the refinement relation is transitive and that the composition operators are monotonic with respect to the refinement relation. This ensures that the approach is incremental and compositional.
In [RHS05] we promised that this report would include a discussion on what probability spaces corresponds to a probabilistic STAIRS specification. However, in order to treat this question properly we have decided that this question is better left to a separate paper.

12 Acknowledgments

The research on which this paper reports has been carried out within the context of the IKT-2010 project SARDAS (15295/431) and the IKT SOS project ENFORCE (164382/V30), both funded by the Research Council of Norway. We thank Rolv Bræk, Øystein Haugen, Birger Møller-Pedersen, Mass Soldal Lund, Judith Rossebø, Ragnhild Kobro Runde, Manfred Broy, Ina Schieferdecker, Thomas Weigert and the anonymous reviewers for helpful feedback.

References

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

[DD02] K. R. Davidson and A. P. Donsig. *Real Analysis with Real Applications*. Prentice Hall, 2002.

[DH01] W. Damm and D. Harel. LSCs: Breathing life into message sequence charts. *Formal Methods in System Design*, 19(1):45–80, 2001.

[HHRS05] Ø. Haugen, K. E. Husa, R. K. Runde, and K. Stølen. STAIRS towards formal design with sequence diagrams. *Journal of Software and Systems Modeling*, 22(4):349–458, 2005.

[HHRS06] Ø. Haugen, K. E. Husa, R. K. Runde, and K. Stølen. Why timed sequence diagrams require three-event semantics. Technical Report 309, Department of Informatics, University of Oslo, December 2006.

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

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

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

[Jan03] D. N. Jansen. *Extensions of Statecharts with Probability, Time, and Stochastic Timing*. PhD thesis, University of Twente, 2003.

[JL91] B. Jonsson and K. G. Larsen. Specification and refinement of probabilistic processes. In *Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science*, pages 266–277, 1991.

[MM99] C. Morgan and A. McIver. pGCL: Formal reasoning for random algorithms. *South African Computer Journal*, 22:14–27, 1999.

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

[OMG05] Object Management Group. *UML Profile for Schedulability, Performance and Time Specification*, version 1.1 formal/05-01-02 edition, 2005.

[RHS05] A. Refsdal, K. E. Husa, and K. Stølen. Specification and refinement of soft real-time requirements using sequence diagrams. In *Proceedings of Formal Modeling and Analysis of Timed Systems: Third International Conference, FORMATS 2005*, volume 3829 of *LNCS*, pages 32–48, 2005.

[Ros95] A. W. Roscoe. CSP and determinism in security modelling. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, pages 114–127, Washington, DC, USA, 1995. IEEE Computer Society.

[Seg95] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.

A Additional operators

A.1 loop

We now introduce a looping construct to the formalism. Syntactically this is done by adding the following case to those given in 4.1 and 6.1:

– $d \in \mathcal{D} \wedge I \subseteq \mathbb{N}_0 \cup \{\infty\} \Rightarrow \mathsf{loop}\ I\ d \in \mathcal{D}$

where $\mathbb{N}_0$ denotes the set of natural numbers including 0.
Before defining the semantics of this construct we need to give some definitions in order to handle the case where the loop may be infinite. These definitions build on the ones given in [HHRS06]. Intuitively, an infinite loop corresponds to infinitely many weak sequencing steps. A *chain* of p-obligations is an infinite sequence of p-obligations such that each element is a sequential composition of the previous p-obligation in the chain and some other appropriate p-obligation. For a set of p-obligations O, its chains is defined as:

$$\begin{aligned} chains(O) \quad \stackrel{\mathsf{def}}{=} \quad & \{\bar{po} \in \mathcal{P}^\infty \mid \bar{po}[1] \in O \wedge \\ & \forall\, j \in \mathbb{N} : \exists\, po \in O : \bar{po}[j+1] = \bar{po}[j] \gtrsim po\} \end{aligned} \tag{36}$$

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

$$\begin{aligned} posCh(\bar{po}) \quad \stackrel{\mathsf{def}}{=} \quad & \{\bar{t} \in \mathcal{H}^\infty \mid \forall\, j \in \mathbb{N} : \bar{t}[j] \in \pi_1.\pi_1.\bar{po}[j] \wedge \\ & \exists\, t \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \gtrsim \{t\}\ \} \end{aligned} \tag{37}$$

For a chain $\bar{po}$ of p-obligations we get a negative chain of traces by selecting the traces such that the first one is negative in some obligation $\bar{po}[i]$ and all the following traces belong to the negative trace sets of the corresponding p-obligations. By starting from some obligation $\bar{po}[i]$ and not just from $\bar{po}[1]$, we take into account that a negative trace may have been positive during a finite number of initial iterations. In the same way as for $posCh(\bar{po})$, each trace in the chain is a weak sequencing extension of the previous trace in the chain. According to definition 17, once we have selected a negative trace, all extensions of this trace with other traces that are positive or negative will also be negative. Hence, we get the following definition for the negative chains of traces:

$$\begin{aligned} negCh(\bar{po}) \quad \stackrel{\mathsf{def}}{=} \quad & \{\bar{t} \in \mathcal{H}^\infty \mid \exists\, i \in \mathbb{N} : \forall\, j \in \mathbb{N} : \\ & \bar{t}[j] \in \pi_2.(\pi_1.\bar{po}[j+i-1]) \wedge \\ & \exists\, t \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \gtrsim \{t\}\ \} \end{aligned} \tag{38}$$

From a chain $\bar{po}$ of p-obligations, we obtain a chain of probabilities by selecting one probability from each p-obligation in the chain $\bar{po}$, and such that each probability in the chain is equal to the previous probability multiplied by an arbitrary probability. For a chain $\bar{po}$ of p-obligations, we define its chains of probabilities as

$$prob(\bar{po}) \stackrel{\mathsf{def}}{=} \{\bar{q} \in [0,1]^\infty \mid \forall\, j \in \mathbb{N} : \bar{q}[j] \in \pi_2.\bar{po}[j] \wedge \exists\, q \in [0,1] : \bar{q}[j+1] \in \bar{q}[j] * q\} \tag{39}$$

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

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

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

$$\sqcup\bar{t} \stackrel{\mathsf{def}}{=} \{h \in \mathcal{H} \mid \forall\, l \in \mathcal{L} : e.l \circledS h = \sqcup_l \bar{t}\} \tag{40}$$

For a chain of p-obligations $\bar{po}$, we then define the p-obligation $\sqcup\bar{po}$ as:

$$\sqcup\bar{po} \stackrel{\mathsf{def}}{=} ((\bigcup_{\bar{t} \in posCh(\bar{po})} \sqcup\bar{t}, \bigcup_{\bar{t} \in negCh(\bar{po})} \sqcup\bar{t}), \{\lim_{j\to\infty} \bar{q} \mid \bar{q} \in prob(\bar{po})\}) \tag{41}$$

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

$$\mu_0\, O \stackrel{\mathsf{def}}{=} \{((\{\langle\rangle\}, \varnothing), \{1\})\} \tag{42}$$

$$\mu_1\, O \stackrel{\mathsf{def}}{=} O \tag{43}$$

$$\mu_n\, O \stackrel{\mathsf{def}}{=} O \gtrsim \mu_{n-1}\, O \text{ for } 1 < n < \infty \tag{44}$$

$$\mu_\infty\, O \stackrel{\mathsf{def}}{=} \{\sqcup\bar{po} \mid \bar{po} \in chains(O)\} \tag{45}$$

The semantics of loop is now defined as:

$$[\![\ \mathsf{loop}\ I\ d\ ]\!] \stackrel{\mathsf{def}}{=} \biguplus_{i \in I} \mu_i [\![\ d\ ]\!] \tag{46}$$

where

$$\biguplus_{i \in I} P_i \stackrel{\mathsf{def}}{=} \{\biguplus_{i \in I}((p_i, n_i), Q_i) \mid \forall\, i \in I : ((p_i, n_i), Q_i) \in P_i\} \tag{47}$$

and

$$\biguplus_{i \in I}((p_i, n_i), Q_i) \stackrel{\mathsf{def}}{=} ((\bigcup_{i \in I} p_i, \bigcup_{i \in I} n_i), \Pi_{i \in I}\, Q_i) \tag{48}$$

A.2 assert

The assert operator is used to define all inconclusive traces as negative. In the definition of syntax the following case is added to those given in 4.1 and 6.1:

– $d \in \mathcal{D} \Rightarrow \mathsf{assert}\ d \in \mathcal{D}$

The semantics for this construct is given by

$$[\![\ \mathsf{assert}\ d\ ]\!] \stackrel{\mathsf{def}}{=} \{((p, n \cup (\mathcal{H} \setminus p)), Q) \mid ((p, n), Q) \in [\![\ d\ ]\!]\} \tag{49}$$

A.3 N-ary palt

For expressing probabilistic choices with more than two alternatives we extend the palt operator so that it can take any finite number of operands. In the definition of syntax the following case is added to those given in 4.1 and 6.1:

– $n \in \mathbb{N} \setminus \{1\} \wedge d_1, ..., d_n \in \mathcal{D} \wedge Q_1, ..., Q_n \subseteq [0...1] \Rightarrow \mathsf{palt}(d_1; Q_1, ..., d_n; Q_n) \in \mathcal{D}$

We do not allow $n = 1$ since this would allow probability decoration in cases with only one operand. In such cases 1 should be the only acceptable probability. The semantics for palt is then given by

$$\begin{aligned}
&[\![\ \mathsf{palt}(d_1; Q_1, ..., d_n; Q_n)\ ]\!] \stackrel{\mathsf{def}}{=} \\
&\quad \{(\oplus \bigcup_{i \in S} \{po_i\}, \sum_{i \in S} \pi_2.po_i) \mid \\
&\qquad S \subseteq \{1, ..., n\} \wedge S \neq \varnothing \wedge \forall\, i \in S : po_i \in [\![\ d_i; Q_i\ ]\!]\} \\
&\quad \cup \{(\oplus \bigcup_{i=1}^{n} [\![\ d_i; Q_i\ ]\!], \{1\} \cap \sum_{i=1}^{n} Q_i)\}
\end{aligned} \tag{50}$$

Note that if $n = 2$ then this defintion gives the same semantics as definition 32.

B Proofs

To simplify notation in the proofs we overload the definitions of the operators $\neg$, $\wr$, $\|$, $\gtrsim$ and $\uplus$ to p-obligations:

$$\neg\,(o, Q) \stackrel{\mathsf{def}}{=} (\neg\, o, Q) \tag{51}$$

$$(o, Q) \wr C \stackrel{\mathsf{def}}{=} (o \wr C, Q) \tag{52}$$

$$(o_1, Q_1) \parallel (o_2, Q_2) \stackrel{\mathsf{def}}{=} (o_1 \parallel o_2, Q_1 * Q_2) \tag{53}$$

$$(o_1, Q_1) \gtrsim (o_2, Q_2) \stackrel{\mathsf{def}}{=} (o_1 \gtrsim o_2, Q_1 * Q_2) \tag{54}$$

$$(o_1, Q_1) \uplus (o_2, Q_2) \stackrel{\mathsf{def}}{=} (o_1 \uplus o_2, Q_1 * Q_2) \tag{55}$$

and further to sets of p-obligations:

$$\neg\, O \stackrel{\mathsf{def}}{=} \{\neg\, po \mid po \in O\} \tag{56}$$

$$O \wr C \stackrel{\mathsf{def}}{=} \{po \wr C \mid po \in O\} \tag{57}$$

$$O_1 \parallel O_2 \stackrel{\mathsf{def}}{=} \{po_1 \parallel po_2 \mid po_1 \in O_1 \wedge po_2 \in O_2\} \tag{58}$$

$$O_1 \gtrsim O_2 \stackrel{\mathsf{def}}{=} \{po_1 \gtrsim po_2 \mid po_1 \in O_1 \wedge po_2 \in O_2\} \tag{59}$$

$$O_1 \uplus O_2 \stackrel{\mathsf{def}}{=} \{po_1 \uplus po_2 \mid po_1 \in O_1 \wedge po_2 \in O_2\} \tag{60}$$

Multiplication of a p-obligation or a set of p-obligations with a probability set Q' is defined as follows:

$$(o, Q) * Q' \stackrel{\mathsf{def}}{=} (o, Q * Q') \tag{61}$$

$$O * Q' \stackrel{\mathsf{def}}{=} \{po * Q' \mid po \in O\} \tag{62}$$

For p-obligations we introduce the functions *pos* and *neg* that return the set of positive and negative traces, resepctively:

$$pos.((p, n), Q) \stackrel{\mathsf{def}}{=} p \tag{63}$$

$$neg.((p, n), Q) \stackrel{\mathsf{def}}{=} n \tag{64}$$

The definition 35 of refinement is extended so that it applies to all sets of p-obligations (and not just to sequence diagrams as in definition 35). If O and O' are sets of p-obligations then $O \rightsquigarrow O'$ iff

$$\forall\, po \in O : 0 \notin \pi_2.po \Rightarrow \exists\, po' \in O' : po \rightsquigarrow po' \tag{65}$$

We also define $\rightsquigarrow$ for interaction obligations: $(p, n) \rightsquigarrow (p', n')$ iff

$$n \subseteq n' \wedge p \subseteq p' \cup n' \tag{66}$$

Note that these definitions ensure that

$$d \rightsquigarrow d' \quad \Leftrightarrow \quad [\![\ d\ ]\!] \rightsquigarrow [\![\ d'\ ]\!] \tag{67}$$

$$o \rightsquigarrow o' \wedge Q' \subseteq Q \quad \Rightarrow \quad (o, Q) \rightsquigarrow (o', Q') \tag{68}$$

In the proofs we use po and po_i as shorthand notation for $((p, n), Q)$ and $((p_i, n_i), Q_i)$, respectively.

B.1 Transitivity of refinement

The refinement relation needs to be transitive in order to allow a stepwise development from an initial specification with a high level of abstraction to a detailed specification suitable for implementation.

Lemma 1. *Transitivity of $\rightsquigarrow$ for p-obligations*
ASSUME: 1. $((p, n), Q) \rightsquigarrow ((p', n'), Q')$
2. $((p', n'), Q') \rightsquigarrow ((p'', n''), Q'')$
PROVE: $((p, n), Q) \rightsquigarrow ((p'', n''), Q'')$

$\langle 1 \rangle 1.$ $n \subseteq n''$
  $\langle 2 \rangle 1.$ $n \subseteq n'$
    PROOF: By assumtion 1.
  $\langle 2 \rangle 2.$ $n' \subseteq n''$
    PROOF: By assumtion 2.
  $\langle 2 \rangle 3.$ Q.E.D.
    PROOF: By $\langle 2 \rangle 1$, $\langle 2 \rangle 2$ and transitivity of $\subseteq$.
$\langle 1 \rangle 2.$ $p \subseteq p'' \cup n''$
  $\langle 2 \rangle 1.$ $p \subseteq p' \cup n'$
    PROOF: By assumtion 1
  $\langle 2 \rangle 2.$ $p' \subseteq p'' \cup n''$
    PROOF: By assumtion 2
  $\langle 2 \rangle 3.$ $p \subseteq (p'' \cup n'') \cup n'$
    PROOF: By $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$
  $\langle 2 \rangle 4.$ $n' \subseteq n''$
    PROOF: By assumtion 2
  $\langle 2 \rangle 5.$ $(p'' \cup n'') \cup n' = p'' \cup n''$
    PROOF: By $\langle 2 \rangle 4$
  $\langle 2 \rangle 6.$ Q.E.D.
    PROOF: By $\langle 2 \rangle 3$ and $\langle 2 \rangle 5$
$\langle 1 \rangle 3.$ $Q'' \subseteq Q$
  $\langle 2 \rangle 1.$ $Q' \subseteq Q$
    PROOF: By assumption 1
  $\langle 2 \rangle 2.$ $Q'' \subseteq Q'$

PROOF: By assumption 2
⟨2⟩3. Q.E.D.
PROOF: By ⟨2⟩1, ⟨2⟩2 and transitivity of $\subseteq$
⟨1⟩4. Q.E.D.
PROOF: By ⟨1⟩1, ⟨1⟩2, ⟨1⟩3 and definition 34

□

Theorem 1. *Transitivity of $\rightsquigarrow$ for specifications*
ASSUME: 1. $d \rightsquigarrow d'$
2. $d' \rightsquigarrow d''$
PROVE: $d \rightsquigarrow d''$
LET: $[\![d]\!] = O \wedge [\![d']\!] = O' \wedge [\![d'']\!] = O''$
⟨1⟩1. $\forall po \in O : 0 \notin \pi_2.po \Rightarrow \exists po'' \in O'' : po \rightsquigarrow po''$
⟨2⟩1. ASSUME: $po \in O$
PROVE: $0 \notin \pi_2.po \Rightarrow \exists po'' \in O'' : po \rightsquigarrow po''$
⟨3⟩1. ASSUME: $0 \notin \pi_2.po$
PROVE: $\exists po'' \in O'' : po \rightsquigarrow po''$
⟨4⟩1. LET: $po' \in O'$ s.t. $po \rightsquigarrow po'$
PROOF: By assumption 1 and assumption ⟨3⟩1
⟨4⟩2. $\exists po'' \in O'' : po' \rightsquigarrow po''$
⟨5⟩1. $0 \notin \pi_2.po'$
PROOF: By assumption ⟨3⟩1 and ⟨4⟩1
⟨5⟩2. Q.E.D.
PROOF: By assumption 2 and ⟨5⟩1
⟨4⟩3. LET: $po'' \in O''$ s.t. $po' \rightsquigarrow po''$
PROOF: By ⟨4⟩2
⟨4⟩4. $po \rightsquigarrow po''$
PROOF: By ⟨4⟩1, ⟨4⟩3 and Lemma 1
⟨4⟩5. Q.E.D.
PROOF: By ⟨4⟩3 ($po'' \in O''$) and ⟨4⟩4
⟨3⟩2. Q.E.D.
PROOF: $\Rightarrow$-rule
⟨2⟩2. Q.E.D.
PROOF: $\forall$-rule
⟨1⟩2. Q.E.D.
PROOF: By definition 35

□

B.2 Monotonicity

When dealing with complex specifications it is very important to be able to work with one part at a time. Monotonicity the composition operators w.r.t. refinement ensures that different parts of a specification can be refined independently from each other. In this sense the approach is compositional.

Monotonicity of neg w.r.t. $\rightsquigarrow$

Lemma 2. *Monotonicity of $\neg$ for single p-obligations w.r.t. $\rightsquigarrow$*
ASSUME: $((p, n), Q) \rightsquigarrow ((p', n'), Q')$
PROVE: $\neg\,((p, n), Q) \rightsquigarrow \neg\,((p', n'), Q')$, i.e.
$((\{\langle\rangle\}, p \cup n), Q) \rightsquigarrow ((\{\langle\rangle\}, p' \cup n'), Q')$

$\langle 1\rangle 1$. $n \subseteq n' \wedge p \subseteq p' \cup n'$
PROOF: By definition 34 and the assumption.
$\langle 1\rangle 2$. $Q' \subseteq Q$
PROOF: By definition 34 and the assumption.
$\langle 1\rangle 3$. $\{\langle\rangle\} \subseteq \{\langle\rangle\} \cup p' \cup n'$
PROOF: By basic set theory.
$\langle 1\rangle 4$. $p \cup n \subseteq p' \cup n'$
PROOF: By $\langle 1\rangle 1$ and basic set theory.
$\langle 1\rangle 5$. $p \cup n \subseteq p' \cup n' \wedge \{\langle\rangle\} \subseteq \{\langle\rangle\} \cup p' \cup n' \wedge Q' \subseteq Q$
PROOF: $\wedge$-introduction from $\langle 1\rangle 4$, $\langle 1\rangle 3$ and $\langle 1\rangle 2$.
$\langle 1\rangle 6$. Q.E.D.
PROOF: By $\langle 1\rangle 5$ and definition 34.

□

Theorem 2. *Monotonicity of $\neg$ w.r.t. $\rightsquigarrow$*
ASSUME: $O \rightsquigarrow O'$
PROVE: $\neg\, O \rightsquigarrow \neg\, O'$

$\langle 1\rangle 1$. $\forall\, po \in \neg\, O : 0 \notin \pi_2.po \Rightarrow \exists\, po' \in \neg\, O' : po \rightsquigarrow po'$
$\langle 2\rangle 1$. ASSUME: $po \in \neg\, O$
PROVE: $0 \notin \pi_2.po \Rightarrow \exists\, po' \in \neg\, O' : po \rightsquigarrow po'$
$\langle 3\rangle 1$. ASSUME: $0 \notin \pi_2.po$
PROVE: $\exists\, po' \in \neg\, O' : po \rightsquigarrow po'$
$\langle 4\rangle 1$. LET: $\overline{po} \in O$ s.t. $\neg\, \overline{po} = po$
PROOF: By definition 56 and assumption $\langle 2\rangle 1$
$\langle 4\rangle 2$. $0 \notin \pi_2.\overline{po}$
PROOF: By $\langle 4\rangle 1$ and assumption $\langle 3\rangle 1$
$\langle 4\rangle 3$. LET: $\overline{po'} \in O'$ s.t. $\overline{po} \rightsquigarrow \overline{po'}$
PROOF: By $\langle 4\rangle 1$, $\langle 4\rangle 2$, the main assumption and definition 35
$\langle 4\rangle 4$. $\neg\, \overline{po} \rightsquigarrow \neg\, \overline{po'}$
PROOF: By $\langle 4\rangle 3$ and Lemma 2
$\langle 4\rangle 5$. $\neg\, \overline{po'} \in \neg\, O'$
PROOF: By $\langle 4\rangle 3$ and definition 56
$\langle 4\rangle 6$. Q.E.D.
PROOF: $\exists$ -rule with $\langle 4\rangle 4$ and $\langle 4\rangle 5$; $\neg\, \overline{po'}$ is the po' we are looking for.
$\langle 3\rangle 2$. Q.E.D.
PROOF: $\Rightarrow$ -rule
$\langle 2\rangle 2$. Q.E.D.
PROOF: $\forall$ -rule
$\langle 1\rangle 2$. Q.E.D.

PROOF: By ⟨1⟩1 and definition 65

□

Monotonicity of neg w.r.t. $\rightsquigarrow$ follows immediately from Theorem 2.

Monotonicity of seq w.r.t. $\rightsquigarrow$

Lemma 3. *Monotonicity of $\succsim$ w.r.t. $\rightsquigarrow$ for p-obligations*
ASSUME: 1. $((p, n), Q) = ((p_1, n_1), Q_1) \succsim ((p_2, n_2), Q_2)$
2. $((p', n'), Q') = ((p'_1, n'_1), Q'_1) \succsim ((p'_2, n'_2), Q'_2)$
3. $((p_1, n_1), Q_1) \rightsquigarrow ((p'_1, n'_1), Q'_1)$
4. $((p_2, n_2), Q_2) \rightsquigarrow ((p'_2, n'_2), Q'_2)$
PROVE: $((p, n), Q) \rightsquigarrow ((p', n'), Q')$

⟨1⟩1. $n \subseteq n' \wedge p \subseteq p' \cup n'$
PROOF: See proof of Lemma 30 in [HHRS06].
⟨1⟩2. $Q' \subseteq Q$
⟨2⟩1. $Q'_1 \subseteq Q_1$
PROOF: By assumption 3
⟨2⟩2. $Q'_2 \subseteq Q_2$
PROOF: By assumption 4
⟨2⟩3. $Q = Q_1 * Q_2$
PROOF: By assumption 1 and definition 54
⟨2⟩4. $Q' = Q'_1 * Q'_2$
PROOF: By assumption 2 and definition 54
⟨2⟩5. Q.E.D.
PROOF: By ⟨2⟩1, ⟨2⟩2, ⟨2⟩3, ⟨2⟩4 and definition 28
⟨1⟩3. Q.E.D.
PROOF: By ⟨1⟩1, ⟨1⟩2 and definition 34

□

Theorem 3. *Monotonicity of $\succsim$ w.r.t. $\rightsquigarrow$*
ASSUME: 1. $O_1 \rightsquigarrow O'_1$
2. $O_2 \rightsquigarrow O'_2$
PROVE: $O_1 \succsim O_2 \rightsquigarrow O'_1 \succsim O'_2$

⟨1⟩1. $\forall\, po \in O_1 \succsim O_2 : 0 \notin \pi_2.po \Rightarrow \exists\, po' \in O'_1 \succsim O'_2 : po \rightsquigarrow po'$
⟨2⟩1. ASSUME: $po \in O_1 \succsim O_2$
PROVE: $0 \notin \pi_2.po \Rightarrow \exists\, po' \in O'_1 \succsim O'_2 : po \rightsquigarrow po'$
⟨3⟩1. ASSUME: $0 \notin \pi_2.po$
PROVE: $\exists\, po' \in O'_1 \succsim O'_2 : po \rightsquigarrow po'$
⟨4⟩1. LET: $po_1 \in O_1, po_2 \in O_2$ s.t. $po = po_1 \succsim po_2$
PROOF: By assumption ⟨2⟩1
⟨4⟩2. $0 \notin \pi_2.po_1 \wedge 0 \notin \pi_2.po_2$
PROOF: By assumption ⟨3⟩1
⟨4⟩3. LET: $po'_1 \in O'_1$ s.t $po_1 \rightsquigarrow po'_1$
PROOF: By ⟨4⟩2, ⟨4⟩1 and assumption 1

⟨4⟩4. LET: $po_2' \in O_2'$ s.t. $po_2 \rightsquigarrow po_2'$
PROOF: By ⟨4⟩2, ⟨4⟩1 and assumption 2
⟨4⟩5. $po_1 \succsim po_2 \rightsquigarrow po_1' \succsim po_2'$
PROOF: By ⟨4⟩3, ⟨4⟩4 and Lemma 3
⟨4⟩6. $po_1' \succsim po_2' \in O_1' \succsim O_2'$
PROOF: By ⟨4⟩3 and ⟨4⟩4
⟨4⟩7. Q.E.D.
PROOF: By ⟨4⟩5 and ⟨4⟩6; $po_1' \succsim po_2'$ is the po' we are looking for
⟨3⟩2. Q.E.D.
PROOF: ⇒-rule
⟨2⟩2. Q.E.D.
PROOF: ∀-rule
⟨1⟩2. Q.E.D.
PROOF: By definition 65

□

Monotonicity of seq w.r.t. $\rightsquigarrow$ follows immediately from Theorem 3

Monotonicity of par w.r.t. $\rightsquigarrow$

Lemma 4. *Monotonicity of $\|$ w.r.t. $\rightsquigarrow$ for p-obligations*
ASSUME: 1. $((p, n), Q) = ((p_1, n_1), Q_1) \parallel ((p_2, n_2), Q_2)$
2. $((p', n'), Q') = ((p_1', n_1'), Q_1') \parallel ((p_2', n_2'), Q_2')$
3. $((p_1, n_1), Q_1) \rightsquigarrow ((p_1', n_1'), Q_1')$
4. $((p_2, n_2), Q_2) \rightsquigarrow ((p_2', n_2'), Q_2')$
PROVE: $((p, n), Q) \rightsquigarrow ((p', n'), Q')$

PROOF: The proof is similar to the proof for Lemma 3; just replace $\succsim$ with $\|$, and refer to Lemma 31 in [HHRS06] instead of Lemma 30 in [HHRS06].

□

Theorem 4. *Monotonicity of $\|$ w.r.t. $\rightsquigarrow$*
ASSUME: 1. $O_1 \rightsquigarrow O_1'$
2. $O_2 \rightsquigarrow O_2'$
PROVE: $O_1 \parallel O_2 \rightsquigarrow O_1' \parallel O_2'$

PROOF: The proof is similar to the proof for $\succsim$; just replace $\succsim$ with $\|$ and refer to Lemma 4 instead of Lemma 3.

□

Monotonicity of par w.r.t. $\rightsquigarrow$ follows immediately from Theorem 4.

Monotonicity of alt w.r.t. $\rightsquigarrow$

Lemma 5. *Monotonicity of $\uplus$ w.r.t. $\rightsquigarrow$ for p-obligations*
ASSUME: 1. $((p_1, n_1), Q_1) \rightsquigarrow ((p_1', n_1'), Q_1')$
2. $((p_2, n_2), Q_2) \rightsquigarrow ((p_2', n_2'), Q_2')$
PROVE: $((p_1, n_1), Q_1) \uplus ((p_2, n_2), Q_2) \rightsquigarrow ((p_1', n_1'), Q_1') \uplus ((p_2', n_2'), Q_2')$

⟨1⟩1. $((p_1 \cup p_2, n_1 \cup n_2), Q_1 * Q_2) \rightsquigarrow ((p_1' \cup p_2', n_1' \cup n_2'), Q_1' * Q_2')$
  ⟨2⟩1. $n_1 \cup n_2 \subseteq n_1' \cup n_2' \wedge p_1 \cup p_2 \subseteq p_1' \cup p_2' \cup n_1' \cup n_2'$
    ⟨3⟩1. $n_1 \cup n_2 \subseteq n_1' \cup n_2'$
      ⟨4⟩1. $n_1 \subseteq n_1'$
        PROOF: By assumption 1
      ⟨4⟩2. $n_2 \subseteq n_2'$
        PROOF: By assumption 2
      ⟨4⟩3. Q.E.D.
        PROOF: By ⟨4⟩1, ⟨4⟩2 and basic set theory
    ⟨3⟩2. $p_1 \cup p_2 \subseteq p_1' \cup p_2' \cup n_1' \cup n_2'$
      ⟨4⟩1. $p_1 \subseteq p_1' \cup n_1'$
        PROOF: By assumption 1
      ⟨4⟩2. $p_2 \subseteq p_2' \cup n_2'$
        PROOF: By assumption 2
      ⟨4⟩3. Q.E.D.
        PROOF: By ⟨4⟩1, ⟨4⟩2 and basic set theory
    ⟨3⟩3. Q.E.D.
      PROOF: $\wedge$-introduction: ⟨3⟩1 and ⟨3⟩2
  ⟨2⟩2. $Q_1' * Q_2' \subseteq Q_1 * Q_2$
    ⟨3⟩1. $Q_1' \subseteq Q_1 \wedge Q_2' \subseteq Q_2$
      PROOF: By assumption 1 and assumption 2
    ⟨3⟩2. Q.E.D.
      PROOF: By ⟨3⟩1 and definition 28
  ⟨2⟩3. Q.E.D.
    PROOF: By ⟨2⟩1, ⟨2⟩2 and definition 34
⟨1⟩2. Q.E.D.
  PROOF: By ⟨1⟩1 and definition 55

□

Theorem 5. *Monotonicity of $\uplus$ w.r.t. $\rightsquigarrow$*
ASSUME: 1. $O_1 \rightsquigarrow O_1'$
2. $O_2 \rightsquigarrow O_2'$
PROVE: $O_1 \uplus O_2 \rightsquigarrow O_1' \uplus O_2'$

⟨1⟩1. $\forall\, po \in O_1 \uplus O_2 : 0 \notin \pi_2.po \Rightarrow \exists\, po' \in O_1' \uplus O_2' : po \rightsquigarrow po'$
  ⟨2⟩1. ASSUME: $po \in O_1 \uplus O_2$
    PROVE: $0 \notin \pi_2.po \Rightarrow \exists\, po' \in O_1' \uplus O_2' : po \rightsquigarrow po'$
    ⟨3⟩1. ASSUME: $0 \notin \pi_2.po$
      PROVE: $\exists\, po' \in O_1' \uplus O_2' : po \rightsquigarrow po'$
      ⟨4⟩1. LET: $po_1 \in O_1, po_2 \in O_2$ s.t. $po = po_1 \uplus po_2$
        PROOF: By assumption ⟨2⟩1
      ⟨4⟩2. $0 \notin \pi_2.po_1 \wedge 0 \notin \pi_2.po_2$
        PROOF: By assumption ⟨3⟩1 and ⟨4⟩1
      ⟨4⟩3. LET: $po_1' \in O_1'$ s.t. $po_1 \rightsquigarrow po_1'$
        PROOF: By assumption 1, ⟨4⟩1 and ⟨4⟩2
      ⟨4⟩4. LET: $po_2' \in O_2'$ s.t. : $po_2 \rightsquigarrow po_2'$

PROOF: By assumption 2, ⟨4⟩1 and ⟨4⟩2

⟨4⟩5. $po_1 \uplus po_2 \rightsquigarrow po_1' \uplus po_2'$

PROOF: By ⟨4⟩3, ⟨4⟩4 and Lemma 5

⟨4⟩6. $po_1' \uplus po_2' \in O_1' \uplus O_2'$

PROOF: By ⟨4⟩3 and ⟨4⟩4

⟨4⟩7. Q.E.D.

PROOF: By ⟨4⟩5 and ⟨4⟩6; $po_1' \uplus po_2'$ is the po' we are looking for

⟨3⟩2. Q.E.D.

PROOF: ⇒-rule

⟨2⟩2. Q.E.D.

PROOF: ∀-rule

⟨1⟩2. Q.E.D.

PROOF: By definition 65

□

Monotonicity of **alt** w.r.t. $\rightsquigarrow$ follows immediately from Theorem 5.

Monotonicity of palt w.r.t. $\rightsquigarrow$

Lemma 6. *Refinement of combinations of sets of p-obligations*

ASSUME: 1. $\oplus O_1 \rightsquigarrow \oplus O_1'$

2. $\oplus O_2 \rightsquigarrow \oplus O_2'$

PROVE: $\oplus(O_1 \cup O_2) \rightsquigarrow \oplus(O_1' \cup O_2')$

⟨1⟩1. $\bigcap_{po \in O_1 \cup O_2} n \subseteq \bigcap_{po' \in O_1' \cup O_2'} n'$

⟨2⟩1. $\bigcap_{po_1 \in O_1} n_1 \cap \bigcap_{po_2 \in O_2} n_2 \subseteq \bigcap_{po_1' \in O_1'} n_1' \cap \bigcap_{po_2' \in O_2'} n_2'$

⟨3⟩1. $\bigcap_{po_1 \in O_1} n_1 \subseteq \bigcap_{po_1' \in O_1'} n_1'$

PROOF: By assumption 1

⟨3⟩2. $\bigcap_{po_2 \in O_2} n_2 \subseteq \bigcap_{po_2' \in O_2'} n_2'$

PROOF: By assumption 2

⟨3⟩3. Q.E.D.

PROOF: By ⟨3⟩1, ⟨3⟩2 and basic set theory

⟨2⟩2. $\bigcap_{po_1 \in O_1} n_1 \cap \bigcap_{po_2 \in O_1} n_2 = \bigcap_{po \in O_1 \cup O_2} n \wedge \bigcap_{po_1' \in O_1'} n_1' \cap \bigcap_{po_2' \in O_2'} n_2' = \bigcap_{po' \in O_1' \cup O_2'} n'$

PROOF: By basic set theory

⟨2⟩3. Q.E.D.

PROOF: By ⟨2⟩1 and ⟨2⟩2

⟨1⟩2. $\bigcup_{po \in O_1 \cup O_2} p \cap \bigcap_{po \in O_1 \cup O_2} p \cup n \subseteq (\bigcup_{po' \in O_1' \cup O_2'} p' \cap \bigcap_{po' \in O_1' \cup O_2'} p' \cup n') \cup \bigcap_{po' \in O_1' \cup O_2'} n'$

⟨2⟩1. ASSUME: $t \in \bigcup_{po \in O_1 \cup O_2} p \cap \bigcap_{po \in O_1 \cup O_2} p \cup n$

PROVE: $t \in (\bigcup_{po' \in O_1' \cup O_2'} p' \cap \bigcap_{po' \in O_1' \cup O_2'} p' \cup n') \cup \bigcap_{po' \in O_1' \cup O_2'} n'$

⟨3⟩1. $t \in \bigcup_{po_1 \in O_1} p_1 \vee t \in \bigcup_{po_2 \in O_2} p_2$

PROOF: By assumption ⟨2⟩1

⟨3⟩2. CASE: $t \in \bigcup_{po_1 \in O_1} p_1$

⟨4⟩1. $t \in (\bigcup_{po_1' \in O_1'} p_1' \cap \bigcap_{po_1' \in O_1'} p_1' \cup n_1') \cup \bigcap_{po_1' \in O_1'} n_1'$

⟨5⟩1. $t \in \bigcup_{po_1 \in O_1} p_1 \cap \bigcap_{po_1 \in O_1} p_1 \cup n_1$

⟨6⟩1. $t \in \bigcap_{po_1 \in O_1} p_1 \cup n_1$

PROOF: By assumption ⟨2⟩1

⟨6⟩2. Q.E.D.

PROOF: By ⟨6⟩1 and assumption ⟨3⟩2

⟨5⟩2. Q.E.D.

PROOF: By ⟨5⟩1 and assumption 1

⟨4⟩2. ASSUME: $t \notin (\bigcup_{po' \in O_1' \cup O_2'} p' \cap \bigcap_{po' \in O_1' \cup O_2'} p' \cup n') \cup \bigcap_{po' \in O_1' \cup O_2'} n'$

PROVE: $\bot$

⟨5⟩1. $t \notin (\bigcup_{po_2' \in O_2'} p_2' \cap \bigcap_{po_2' \in O_2'} p_2' \cup n_2') \cup \bigcap_{po_2' \in O_2'} n_2'$

⟨6⟩1. $t \notin \bigcap_{po_2' \in O_2'} p_2' \cup n_2'$

⟨7⟩1. ASSUME: $t \in \bigcap_{po_2' \in O_2'} p_2' \cup n_2'$

PROVE: $\bot$

⟨8⟩1. $t \in \bigcap_{po_1' \in O_1'} p_1' \cup n_1'$

PROOF: By ⟨4⟩1

⟨8⟩2. $t \in \bigcap_{po' \in O_1' \cup O_2'} p' \cup n'$

PROOF: By ⟨8⟩1 and assumption ⟨7⟩1

⟨8⟩3. Q.E.D.

PROOF: By ⟨8⟩2 and assumption ⟨4⟩2

⟨7⟩2. Q.E.D.

PROOF: $\bot$-rule

⟨6⟩2. Q.E.D.

PROOF: By ⟨6⟩1

⟨5⟩2. $t \in \bigcap_{po_2' \in O_2'} n_2'$

⟨6⟩1. $t \notin \bigcup_{po_2 \in O_2} p_2 \cap \bigcap_{po_2 \in O_2} p_2 \cup n_2$

⟨7⟩1. $\bigcup_{po_2 \in O_2} p_2 \cap \bigcap_{po_2 \in O_2} p_2 \cup n_2 \subseteq (\bigcup_{po_2' \in O_2'} p_2' \cap \bigcap_{po_2' \in O_2'} p_2' \cup n_2') \cup \bigcap_{po_2' \in O_2'} n_2$

PROOF: By assumption 2

⟨7⟩2. Q.E.D.

PROOF: By $\langle 5\rangle 1$ and $\langle 7\rangle 1$

$\langle 6\rangle 2$. $t \in \bigcap_{po_2 \in O_2} p_2 \cup n_2$

PROOF: By assumption $\langle 2\rangle 1$

$\langle 6\rangle 3$. $t \notin \bigcup_{po_2 \in O_2} p_2$

PROOF: By $\langle 6\rangle 1$ and $\langle 6\rangle 2$

$\langle 6\rangle 4$. $t \in \bigcap_{po_2 \in O_2} n_2$

PROOF: By $\langle 6\rangle 2$ and $\langle 6\rangle 3$

$\langle 6\rangle 5$. $\bigcap_{po_2 \in O_2} n_2 \subseteq \bigcap_{po'_2 \in O'_2} n'_2$

PROOF: By assumption 2

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

PROOF: By $\langle 6\rangle 4$ and $\langle 6\rangle 5$

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

PROOF: By $\langle 5\rangle 1$ and $\langle 5\rangle 2$

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

PROOF: $\bot$-rule

$\langle 3\rangle 3$. CASE: $t \in \bigcup_{po_2 \in O_2} p_2$

PROOF: Similar proof as case $\langle 3\rangle 2$

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

PROOF: By $\langle 3\rangle 1$ the cases $\langle 3\rangle 2$ and $\langle 3\rangle 3$ are exhaustive

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

PROOF: $\subseteq$-rule

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

PROOF: By $\langle 1\rangle 1$, $\langle 1\rangle 2$ and definition 66

□

Lemma 7. *Refinement of the combination of a set of p-obligations*

ASSUME: $S \subseteq \mathbb{N} \wedge S \neq \varnothing \wedge \forall\, i \in S : po_i \rightsquigarrow po'_i$

PROVE: $(\oplus \bigcup_{i \in S} \{po_i\}, \sum_{i \in S} \pi_2.po_i) \rightsquigarrow (\oplus \bigcup_{i \in S} \{po'_i\}, \sum_{i \in S} \pi_2.po'_i)$

$\langle 1\rangle 1$. CASE: $|S| = 1$ (Induction basis)

$\langle 2\rangle 1$. $\exists j \in \mathbb{N} : S = \{j\}$

PROOF: By assumption $\langle 1\rangle 1$ and the main assumption

$\langle 2\rangle 2$. $(\oplus \bigcup_{i \in \{j\}} \{po_i\}, \sum_{i \in \{j\}} \pi_2.po_i) \rightsquigarrow (\oplus \bigcup_{i \in \{j\}} \{po'_i\}, \sum_{i \in \{j\}} \pi_2.po'_i)$

PROOF: By the main assumption, since for any j, $(\oplus \bigcup_{i \in \{j\}} \{po_i\}, \sum_{i \in \{j\}} \pi_2.po_i) = po_j$

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

PROOF: By $\langle 2\rangle 2$

$\langle 1\rangle 2$. CASE: $|S| > 1$ (Induction step)

$\langle 2\rangle 1$. ASSUME: $|S| \leq k \Rightarrow (\oplus \bigcup_{i \in S} \{po_i\}, \sum_{i \in S} \pi_2.po_i) \rightsquigarrow (\oplus \bigcup_{i \in S} \{po'_i\}, \sum_{i \in S} \pi_2.po'_i)$ (ind. hyp.)

PROVE: $|S| = k+1 \Rightarrow$
$(\oplus \bigcup_{i \in S} \{po_i\}, \sum_{i \in S} \pi_2.po_i) \rightsquigarrow (\oplus \bigcup_{i \in S} \{po'_i\}, \sum_{i \in S} \pi_2.po'_i)$

$\langle 3 \rangle 1$. ASSUME: $|S| = k+1$
PROVE: $(\oplus \bigcup_{i \in S} \{po_i\}, \sum_{i \in S} \pi_2.po_i) \rightsquigarrow (\oplus \bigcup_{i \in S} \{po'_i\}, \sum_{i \in S} \pi_2.po'_i)$

$\langle 4 \rangle 1$. $\exists j \in S, S' \subset S : S = S' \cup \{j\}$
PROOF: By assumption $\langle 3 \rangle 1$

$\langle 4 \rangle 2$. $(\oplus \bigcup_{i \in S'} \{po_i\}, \sum_{i \in S'} \pi_2.po_i) \rightsquigarrow (\oplus \bigcup_{i \in S'} \{po'_i\}, \sum_{i \in S'} \pi_2.po'_i)$
PROOF: By assumption $\langle 2 \rangle 1$ and $\langle 4 \rangle 1$, since $|S'| = k$

$\langle 4 \rangle 3$. $\oplus \bigcup_{i \in S} \{po_i\} \rightsquigarrow \oplus \bigcup_{i \in S} \{po'_i\}$

$\langle 5 \rangle 1$. $\oplus \bigcup_{i \in S'} \{po_i\} \rightsquigarrow \oplus \bigcup_{i \in S'} \{po'_i\}$
PROOF: By $\langle 4 \rangle 2$

$\langle 5 \rangle 2$. $\oplus\{po_j\} \rightsquigarrow \oplus\{po'_j\}$
PROOF: By the overall assumption, since $\oplus\{po\} = \pi_1.po$ for any po

$\langle 5 \rangle 3$. $\oplus \bigcup_{i \in S' \cup \{j\}} \{po_i\} \rightsquigarrow \oplus \bigcup_{i \in S' \cup \{j\}} \{po'_i\}$
PROOF: By $\langle 5 \rangle 1$, $\langle 5 \rangle 2$ and Lemma 6

$\langle 5 \rangle 4$. Q.E.D.
PROOF: By $\langle 5 \rangle 3$

$\langle 4 \rangle 4$. $\sum_{i \in S} \pi_2.po'_i \subseteq \sum_{i \in S} \pi_2.po_i$

$\langle 5 \rangle 1$. $\sum_{i \in S'} \pi_2.po' \subseteq \sum_{i \in S'} \pi_2.po$
PROOF: By $\langle 4 \rangle 2$

$\langle 5 \rangle 2$. $\sum_{i \in \{j\}} \pi_2.po' \subseteq \sum_{i \in \{j\}} \pi_2.po$
PROOF: By the overall assumption

$\langle 5 \rangle 3$. $\sum_{i \in S \cup \{j\}} \pi_2.po' \subseteq \sum_{i \in S \cup \{j\}} \pi_2.po$
PROOF: By $\langle 5 \rangle 1$, $\langle 5 \rangle 2$, definition 31 and basic set theory

$\langle 5 \rangle 4$. Q.E.D.
PROOF: By $\langle 5 \rangle 3$

$\langle 4 \rangle 5$. Q.E.D.
PROOF: By $\langle 4 \rangle 3$ and $\langle 4 \rangle 4$

$\langle 3 \rangle 2$. Q.E.D.
PROOF: $\Rightarrow$-rule

$\langle 2 \rangle 2$. Q.E.D.
PROOF: Induction step

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

□

Theorem 6. *Restricted monotonicity of* palt *w.r.t.* $\rightsquigarrow$
ASSUME: 1. $\forall\, i \leq n : [\![\ d_i\ ]\!] \rightsquigarrow [\![\ d'_i\ ]\!]$
2. $\forall\, i \leq n : Q'_i \subseteq Q_i$
3. $\forall\, i \leq n : \oplus[\![\ d_i\ ]\!] \rightsquigarrow \oplus[\![\ d'_i\ ]\!]$

LET: 1. $O = [\![\ \mathsf{palt}(d_1;Q_1,...,d_n;Q_n)\ ]\!]$
2. $O' = [\![\ \mathsf{palt}(d'_1;Q'_1,...,d'_n;Q'_n)\ ]\!]$

PROVE: $O \rightsquigarrow O'$

$\langle 1\rangle 1.\ \forall\, po \in O : 0 \notin \pi_2.po \Rightarrow \exists\, po' \subseteq O' : po \rightsquigarrow po'$
$\langle 2\rangle 1.$ ASSUME: $po \in O$
PROVE: $0 \notin \pi_2.po \Rightarrow \exists\, po' \in O' : po \rightsquigarrow po'$
$\langle 3\rangle 1.$ ASSUME: $0 \notin \pi_2.po$
PROVE: $\exists\, po' \in O' : po \rightsquigarrow po'$
$\langle 4\rangle 1.$ CASE: $po \in \{(\oplus \bigcup_{i\in S} \{po_i\}, \sum_{i\in S} \pi_2.po_i) \mid S \subseteq \{1,...,n\} \wedge S \neq \varnothing \wedge \forall\, i \in S : po_i \in [\![\ d_i;Q_i\ ]\!]\}$
$\langle 5\rangle 1.$ LET: $S \subseteq \{1,...n\}$ s. t. $S \neq \varnothing \wedge \forall\, i \in S : \exists\, po_i \in [\![\ d_i;Q_i\ ]\!] : po = (\oplus \bigcup_{i\in S} \{po_i\}, \sum_{i\in S} \pi_2.po_i)$
PROOF: By assumption $\langle 4\rangle 1$
$\langle 5\rangle 2.\ \forall\, i \in S : \exists\, po'_i \in [\![\ d'_i;Q'_i\ ]\!] : po_i \rightsquigarrow po'_i$
PROOF: By assumptions 1 and 2
$\langle 5\rangle 3.$ LET: po'_i be such that $po_i \rightsquigarrow po'_i$ for each $i \in S$
PROOF: By $\langle 5\rangle 2$
$\langle 5\rangle 4.\ (\oplus \bigcup_{i\in S} \{po_i\}, \sum_{i\in S} \pi_2.po_i) \rightsquigarrow (\oplus \bigcup_{i\in S} \{po'_i\}, \sum_{i\in S} \pi_2.po'_i)$
PROOF: By $\langle 5\rangle 3$, $\langle 5\rangle 1$ and Lemma 7
$\langle 5\rangle 5.\ (\oplus \bigcup_{i\in S} \{po'_i\}, \sum_{i\in S} \pi_2.po'_i) \in [\![\ \mathsf{palt}(d'_1;Q'_n,...,d'_n;Q'_n)\ ]\!]$
PROOF: By definition 50 and $\langle 5\rangle 2$
$\langle 5\rangle 6.$ Q.E.D.
PROOF: By $\langle 5\rangle 4$ and $\langle 5\rangle 5$; $(\oplus \bigcup_{i\in S} \{po'_i\}, \sum_{i\in S} \pi_2.po'_i)$ is the po' we are looking for.
$\langle 4\rangle 2.$ CASE: $po = (\oplus \bigcup_{i=1}^{n} [\![\ d_i;Q_i\ ]\!], \{1\} \cap \sum_{i=1}^{n} Q_i)$
$\langle 5\rangle 1.\ \{1\} \cap \sum_{i=1}^{n} Q'_i \subseteq \{1\} \cap \sum_{i=1}^{n} Q_i$
PROOF: By assumption 2
$\langle 5\rangle 2.\ \oplus \bigcup_{i=1}^{n} [\![\ d_i;Q_i\ ]\!] \rightsquigarrow \oplus \bigcup_{i=1}^{n} [\![\ d'_i;Q'_i\ ]\!]$
$\langle 6\rangle 1.$ CASE: $n = 2$ (induction basis)
$\langle 7\rangle 1.\ \oplus [\![\ d_1;Q_1\ ]\!] \rightsquigarrow \oplus [\![\ d'_1;Q'_1\ ]\!]$
PROOF: By assumption 3
$\langle 7\rangle 2.\ \oplus [\![\ d_2;Q_2\ ]\!] \rightsquigarrow \oplus [\![\ d'_2;Q'_2\ ]\!]$
PROOF: By assumption 3
$\langle 7\rangle 3.$ Q.E.D.
PROOF: By $\langle 7\rangle 1$, $\langle 7\rangle 2$ and Lemma 6
$\langle 6\rangle 2.$ CASE: $n > 2$ (induction step)
$\langle 7\rangle 1.$ ASSUME: $n \leq k \Rightarrow \oplus \bigcup_{i=1}^{n} [\![\ d_i;Q_i\ ]\!] \rightsquigarrow \oplus \bigcup_{i=1}^{n} [\![\ d'_i;Q'_i\ ]\!]$ (induction hypothesis)

PROVE: $\oplus \bigcup_{i=1}^{k+1} [\![\ d_i;Q_i\ ]\!] \rightsquigarrow \oplus \bigcup_{i=1}^{k+1} [\![\ d'_i;Q'_i\ ]\!]$

$\langle 8\rangle 1.\ \oplus \bigcup_{i=1}^{k} [\![\ d_i;Q_i\ ]\!] \rightsquigarrow \oplus \bigcup_{i=1}^{k} [\![\ d'_i;Q'_i\ ]\!]$

PROOF: By assumption $\langle 7\rangle 1$

$\langle 8\rangle 2.\ \oplus [\![\ d_{k+1};Q_{k+1}\ ]\!] \rightsquigarrow \oplus [\![\ d'_{k+1};Q'_{k+1}\ ]\!]$

PROOF: By assumption 3

$\langle 8\rangle 3.\ \oplus(\bigcup_{i=1}^{k} [\![\ d_k;Q_k\ ]\!] \cup [\![\ d_{k+1};Q_{k+1}\ ]\!]) \rightsquigarrow$
$\oplus(\bigcup_{i=1}^{k} [\![\ d'_k;Q'_k\ ]\!] \cup [\![\ d'_{k+1};Q'_{k+1}\ ]\!])$

PROOF: By $\langle 8\rangle 1$, $\langle 8\rangle 2$ and Lemma 6

$\langle 8\rangle 4.$ Q.E.D.

PROOF: By $\langle 8\rangle 3$

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

PROOF: Induction step

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

PROOF: Induction: $\langle 6\rangle 1$ and $\langle 6\rangle 2$ (Note that according to definition 50, $n \geq 2$.)

$\langle 5\rangle 3.\ (\oplus \bigcup_{i=1}^{n} [\![\ d_i;Q_i\ ]\!], \{1\} \cap \sum_{i=1}^{n} Q_i) \rightsquigarrow (\oplus \bigcup_{i=1}^{n} [\![\ d'_i;Q'_i\ ]\!], \{1\} \cap \sum_{i=1}^{n} Q'_i)$

PROOF: By $\langle 5\rangle 1$ and $\langle 5\rangle 2$

$\langle 5\rangle 4.\ (\oplus \bigcup_{i=1}^{n} [\![\ d'_i;Q'_i\ ]\!], \{1\} \cap \sum_{i=1}^{n} Q'_i) \in O'$

PROOF: By definition 50

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

PROOF: By $\langle 5\rangle 3$ and $\langle 5\rangle 4$; $(\oplus \bigcup_{i=1}^{n} [\![\ d'_i;Q'_i\ ]\!], \{1\} \cap \sum_{i=1}^{n} Q'_i)$ is the po' we are looking for

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

PROOF: By definition 50, the cases $\langle 4\rangle 1$ and $\langle 4\rangle 2$ are exhaustive.

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

PROOF: $\Rightarrow$-rule

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

PROOF: $\forall$-rule

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

PROOF: By $\langle 1\rangle 1$

□

To see why the last assumption in Theorem 6 is necessary consider the following example: For $n = 1, 2, 3$ let

$$[\![\ d_n\ ]\!] = \{(o_n, \{1\})\}$$

and let

$$d'_1 \quad = \quad d_1;\{1\} \ \mathsf{palt}\ d_2;\{0\}$$

$$
\begin{aligned}
d_A &= d_1;[0.4, 0.6] \text{ palt } d_3;[0.4, 0.6] \\
d'_A &= d'_1;[0.4, 0.6] \text{ palt } d_3;[0.4, 0.6]
\end{aligned}
$$

This means that

$$
\begin{aligned}
[\![\ d'_1\ ]\!] &= \{(o_1, \{1\}), (o_2, \{0\}), (\oplus\{o_1, o_2\}, \{1\})\} \\
[\![\ d_A\ ]\!] &= \{(o_1, [0.4, 0.6]), (o_3, [0.4, 0.6]), (\oplus\{o_1, o_3\}, [0.8, 1]), (\oplus\{o_1, o_3\}, \{1\})\} \\
[\![\ d'_A\ ]\!] &= \{(o_1, [0.4, 0.6]), (o_2, \{0\}), (\oplus\{o_1, o_2\}, [0.4, 0.6]), (o_3, [0.4, 0.6]), \\
&\quad (\oplus\{o_1, o_3\}, [0.8, 1]), (\oplus\{o_2, o_3\}, [0.4, 0.6]), (\oplus\{o_1, o_2, o_3\}, [0.8, 1]), \\
&\quad (\oplus\{o_1, o_2, o_3\}, \{1\})\}
\end{aligned}
$$

We now have $d_1 \rightsquigarrow d'_1$ and $d_3 \rightsquigarrow d_3$. But $d_A \rightsquigarrow d'_A$ does not hold, since there is no $po' \in [\![\ d'_A\ ]\!]$ such that $(\oplus\{o_1, o_3\}, \{1\}) \rightsquigarrow po'$. This is because the only p-obligation in $[\![\ d'_A\ ]\!]$ with interaction obligation $\oplus\{o_1, o_3\}$ has probability set $[0.8, 1]$, which is not a subset of $\{1\}$.

Monotonicity of tc w.r.t. $\rightsquigarrow$

Lemma 8. *Monotonicity of $\wr C$ w.r.t. $\rightsquigarrow$ for single p-obligations*
ASSUME: $((p, n), Q) \rightsquigarrow ((p', n'), Q')$
PROVE: $((p, n), Q) \wr C \rightsquigarrow ((p', n'), Q') \wr C$, i.e.
$((p \wr C, n \cup (p \wr \neg C)), Q) \rightsquigarrow ((p' \wr C, n' \cup (p' \wr \neg C)), Q')$

$\langle 1 \rangle 1$. $n \subseteq n \cup (p \wr \neg C)$
PROOF: By basic set theory
$\langle 1 \rangle 2$. $p \subseteq p \wr C \cup (n \cup (p \wr \neg C))$
$\langle 2 \rangle 1$. $p = p \wr C \cup p \wr \neg C$
PROOF: By definition 18
$\langle 2 \rangle 2$. Q.E.D.
PROOF: By $\langle 2 \rangle 1$ and basic set theory
$\langle 1 \rangle 3$. $Q' \subseteq Q$
PROOF: By the assumption
$\langle 1 \rangle 4$. Q.E.D.
PROOF: By $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, $\langle 1 \rangle 3$ and definition 34

□

Theorem 7. *Monotonicity of $\wr C$ w.r.t. $\rightsquigarrow$*
ASSUME: $O \rightsquigarrow O'$
PROVE: $O \wr C \rightsquigarrow O' \wr C$

$\langle 1 \rangle 1$. $\forall\, po \in O \wr C : 0 \notin \pi_2.po \Rightarrow \exists\, po' \in O' \wr C : po \rightsquigarrow po'$
$\langle 2 \rangle 1$. ASSUME: $po \in O \wr C$
PROVE: $0 \notin \pi_2.po \Rightarrow \exists\, po' \in O' \wr C : po \rightsquigarrow po'$
$\langle 3 \rangle 1$. ASSUME: $0 \notin \pi_2.po$

PROVE: $\exists\, po' \in O' \wr C : po \rightsquigarrow po'$

$\langle 4\rangle 1$. LET: $\overline{po} \in O$ s.t. $\overline{po} \wr C = po$

PROOF: By definition 57 and assumption $\langle 2\rangle 1$

$\langle 4\rangle 2$. $0 \notin \pi_2.\overline{po}$

PROOF: By assumption $\langle 3\rangle 1$ and $\langle 4\rangle 1$

$\langle 4\rangle 3$. LET: $\overline{po'} \subseteq O'$ s.t. $\overline{po} \rightsquigarrow \overline{po'}$

PROOF: By $\langle 4\rangle 1$, $\langle 4\rangle 2$ and the assumption

$\langle 4\rangle 4$. $(\overline{po}) \wr C \rightsquigarrow (\overline{po'}) \wr C$

PROOF: By $\langle 4\rangle 3$ and Lemma 8

$\langle 4\rangle 5$. $\overline{po'} \wr C \in O' \wr C$

PROOF: By $\langle 4\rangle 3$ and definition 57

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

PROOF: $\exists$ -rule with $\langle 4\rangle 4$ and $\langle 4\rangle 5$; $\overline{po'} \wr C$ is the po' we are looking for

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

PROOF: $\Rightarrow$ -rule

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

PROOF: $\forall$-rule

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

PROOF: By $\langle 1\rangle 1$

□

Monotonicity of tc w.r.t. $\rightsquigarrow$ follows immediately from Theorem 7.

Non-monotonicity of assert w.r.t. $\rightsquigarrow$ The assert operator is not monotonic w.r.t. $\rightsquigarrow$. This is shown by the following example: Let

$$\begin{aligned} [\![\ d\ ]\!] &= \{((\{t_1\}, \varnothing), \{1\})\} \\ [\![\ d'\ ]\!] &= \{((\{t_1, t_2\}, \varnothing), \{1\})\} \end{aligned}$$

We then have $d \rightsquigarrow d'$. However, (assert d) $\rightsquigarrow$ (assert d') does not hold. To see this, observe that t_2 will be negative in the only p-obligation in assert d but positive in the only p-obligation in assert d'.

Monotonicity of loop w.r.t. $\rightsquigarrow$

Lemma 9. *Monotonicity of μ_i w.r.t. $\rightsquigarrow$*

This proof is partly based on the proof of Lemma 38 in [HHRS06].

ASSUME: $O \rightsquigarrow O' \wedge i \in \mathbb{N}_0 \cup \{\infty\}$

PROVE: $\mu_i O \rightsquigarrow \mu_i O'$

$\langle 1\rangle 1$. CASE: $i = 0$

$\langle 2\rangle 1$. $\mu_i O = \mu_i O' = \{((\{\langle\rangle\}, \varnothing), \{1\})\}$

PROOF: By definition 42 and assumption $\langle 1\rangle 1$

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

PROOF: By $\langle 2\rangle 1$, since $((\{\langle\rangle\}, \varnothing), \{1\}) \rightsquigarrow ((\{\langle\rangle\}, \varnothing), \{1\})$

$\langle 1\rangle 2$. CASE: $i = 1$

⟨2⟩1. $\mu_i O = O \wedge \mu_i O' = O'$
PROOF: By assumption ⟨1⟩2 and definition 43
⟨2⟩2. Q.E.D.
PROOF: By ⟨2⟩1 and the overall assumption
⟨1⟩3. CASE: $1 < i < \infty$
⟨2⟩1. $\mu_2 O \rightsquigarrow \mu_2 O'$ (Induction basis)
⟨3⟩1. $\mu_2 O = O \succsim O \wedge \mu_2 O' = O' \succsim O'$
PROOF: By definitions 43 and 44
⟨3⟩2. Q.E.D.
PROOF: By the overall assumption, ⟨3⟩1 and Theorem 3
⟨2⟩2. ASSUME: $\mu_k O \rightsquigarrow \mu_k O'$ for $1 < k < \infty$ (Induction hypothesis)
PROVE: $\mu_{k+1} O \rightsquigarrow \mu_{k+1} O'$
⟨3⟩1. $\mu_{k+1} O = O \succsim \mu_k O \wedge \mu_{k+1} O' = O' \succsim \mu_k O'$
PROOF: By definition 44
⟨3⟩2. Q.E.D.
PROOF: By the overall assumption, assumption ⟨2⟩2 and Theorem 3
⟨2⟩3. Q.E.D.
PROOF: By induction with ⟨2⟩1 as basis and ⟨2⟩2 as induction step
⟨1⟩4. CASE: $i = \infty$
⟨2⟩1. $\forall\, po \in \{\sqcup \bar{po} \mid \bar{po} \in chains(O)\} : 0 \notin \pi_2.po \Rightarrow$
$\exists\, po' \in \{\sqcup \bar{po}' \mid \bar{po}' \in chains(O')\} : po \rightsquigarrow po'$, i.e.
$\forall\, \bar{po} \in chains(O) : 0 \notin \pi_2. \sqcup \bar{po} \Rightarrow \exists\, \bar{po}' \in chains(O') : \sqcup \bar{po} \rightsquigarrow \sqcup \bar{po}'$
⟨3⟩1. ASSUME: $\bar{po} \in chains(O)$
PROVE: $0 \notin \pi_2. \sqcup \bar{po} \Rightarrow \exists\, \bar{po}' \in chains(O') : \sqcup \bar{po} \rightsquigarrow \sqcup \bar{po}'$
⟨4⟩1. ASSUME: $0 \notin \pi_2. \sqcup \bar{po}$
PROVE: $\exists\, \bar{po}' \in chains(O') : \sqcup \bar{po} \rightsquigarrow \sqcup \bar{po}'$
⟨5⟩1. $\exists\, \bar{po}_1 \in chains(O') : \forall\, j \in \mathbb{N} : \bar{po}[j] \rightsquigarrow \bar{po}_1[j]$
⟨6⟩1. LET: $po^1 \in O$ s.t. $\bar{po}[1] = po^1$
$po^{j+1} \in O$ s.t. $\bar{po}[j+1] = \bar{po}[j] \succsim po^{j+1}$ for all $j \in \mathbb{N}$
PROOF: By assumption ⟨3⟩1 and definition 36
⟨6⟩2. LET: $po_2^j \in O'$ s.t. $po^j \rightsquigarrow po_2^j$ for all $j \in \mathbb{N}$
⟨7⟩1. $\forall\, j \in \mathbb{N} : 0 \notin \pi_2.po^j$
⟨8⟩1. $0 \notin \{\lim_{j\to\infty} \bar{q} \mid \bar{q} \in prob(\bar{po})\}$
PROOF: By assumption ⟨4⟩1
⟨8⟩2. Q.E.D.
PROOF: By ⟨8⟩1 and definition 39
⟨7⟩2. Q.E.D.
PROOF: By ⟨7⟩1, the overall assumption, and the fact that $po^j \in O$ for all $j \in \mathbb{N}$
⟨6⟩3. LET: $\bar{po}_2 \in chains(O')$ s.t. $\bar{po}_2[1] = po_2^1 \wedge$
$\forall\, j \in \mathbb{N} : \bar{po}_2[j+1] = \bar{po}_2[j] \succsim po_2^{j+1}$
PROOF: By ⟨6⟩2 and definition 36
⟨6⟩4. $\forall\, j \in \mathbb{N} : \bar{po}[j] \rightsquigarrow \bar{po}_2[j]$
⟨7⟩1. $\bar{po}[1] \rightsquigarrow \bar{po}_2[1]$ (induction basis)
PROOF: By ⟨6⟩3, ⟨6⟩2 and ⟨6⟩1

⟨7⟩2. ASSUME: $\bar{po}[k] \rightsquigarrow \bar{po}_2[k]$ (induction hypothesis)
PROVE: $\bar{po}[k+1] \rightsquigarrow \bar{po}_2[k+1]$
⟨8⟩1. $\bar{po}[k+1] = \bar{po}[k] \succsim po^{k+1}$
PROOF: By ⟨6⟩1
⟨8⟩2. $\bar{po}_2[k+1] = \bar{po}_2[k] \succsim po_2^{k+1}$
PROOF: By ⟨6⟩3
⟨8⟩3. $po^{k+1} \rightsquigarrow po_2^{k+1}$
PROOF: By ⟨6⟩2
⟨8⟩4. Q.E.D.
PROOF: By ⟨8⟩1, ⟨8⟩2, ⟨8⟩3, assumption ⟨7⟩2 and Theorem 3
⟨7⟩3. Q.E.D.
PROOF: By induction with ⟨7⟩1 as induction basis and ⟨7⟩2 as induction step
⟨6⟩5. Q.E.D.
PROOF: By ⟨6⟩3 and ⟨6⟩4; $\bar{po}_2$ is the $\bar{po}_1$ we are looking for
⟨5⟩2. LET: $\bar{po}_1 \in chains(O')$ s.t. $\forall j \in \mathbb{N} : \bar{po}[j] \rightsquigarrow \bar{po}_1[j]$
PROOF: By ⟨5⟩1
⟨5⟩3. $\sqcup\bar{po} \rightsquigarrow \sqcup\bar{po}_1$
⟨6⟩1. $neg.\sqcup\bar{po} \subseteq neg.\sqcup\bar{po}_1$
⟨7⟩1. $\bigcup_{\bar{t}\in negCh(\bar{po})} \sqcup\bar{t} \subseteq \bigcup_{\bar{t}\in negCh(\bar{po}_1)} \sqcup\bar{t}$
⟨8⟩1. $negCh(\bar{po}) \subseteq negCh(\bar{po}_1)$
⟨9⟩1. ASSUME: $\bar{t} \in negCh(\bar{po})$
PROVE: $\bar{t} \in negCh(\bar{po}_1)$
⟨10⟩1. LET: $i \in \mathbb{N}$ s.t. $\forall j \in \mathbb{N} : \bar{t}[j] \in neg.\bar{po}[j+i-1]$
PROOF: By assumption ⟨9⟩1 and definition 38
⟨10⟩2. $\forall j \in \mathbb{N} : \bar{t}[j] \in neg.\bar{po}_1[j+i-1]$
⟨11⟩1. $\bar{po}[j+i-1] \rightsquigarrow \bar{po}_1[j+i-1]$
PROOF: By ⟨5⟩2
⟨11⟩2. Q.E.D.
PROOF: By ⟨11⟩1 and definition 34
⟨10⟩3. $\forall j \in \mathbb{N} : \exists t \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \succsim \{t\}$
PROOF: By assumption ⟨9⟩1 and definition 38
⟨10⟩4. Q.E.D.
PROOF: By ⟨10⟩2, ⟨10⟩3 and definition 38
⟨9⟩2. Q.E.D.
PROOF: $\subseteq$-rule
⟨8⟩2. Q.E.D.
PROOF: By ⟨8⟩1
⟨7⟩2. Q.E.D.
PROOF: By ⟨7⟩1 and definition 41
⟨6⟩2. $pos.\sqcup\bar{po} \subseteq pos.\sqcup\bar{po}_1 \cup neg.\sqcup\bar{po}_1$
⟨7⟩1. $\bigcup_{\bar{t}\in posCh(\bar{po})} \sqcup\bar{t} \subseteq (\bigcup_{\bar{t}\in posCh(\bar{po}_1)} \sqcup\bar{t}) \cup (\bigcup_{\bar{t}\in negCh(\bar{po}_1)} \sqcup\bar{t})$

⟨8⟩1. $\forall\, \bar{t} \in posCh(\bar{po})$:
$\sqcup\, \bar{t} \subseteq (\bigcup_{\bar{t}' \in posCh(p\bar{o}_1)} \sqcup \bar{t}') \cup (\bigcup_{\bar{t}' \in negCh(p\bar{o}_1)} \sqcup \bar{t}')$

⟨9⟩1. ASSUME: $\bar{t} \in posCh(\bar{po})$
PROVE: $\sqcup \bar{t} \subseteq (\bigcup_{\bar{t}' \in posCh(p\bar{o}_1)} \sqcup \bar{t}') \cup (\bigcup_{\bar{t}' \in negCh(p\bar{o}_1)} \sqcup \bar{t}')$

⟨10⟩1. $\forall\, j \in \mathbb{N} : \bar{t}[j] \in pos.p\bar{o}_1[j] \cup neg.p\bar{o}_1[j]$

⟨11⟩1. $\forall\, j \in \mathbb{N} : \bar{t}[j] \in pos.\bar{po}[j]$
PROOF: By assumption ⟨9⟩1

⟨11⟩2. Q.E.D.
PROOF: By ⟨11⟩1 ⟨5⟩2 and definition 34

⟨10⟩2. CASE: $\forall\, j \in \mathbb{N} : \bar{t}[j] \in pos.p\bar{o}_1[j]$

⟨11⟩1. $\sqcup \bar{t} \subseteq \bigcup_{\bar{t}' \in posCh(p\bar{o}_1)} \sqcup \bar{t}'$

⟨12⟩1. $\forall\, j \in \mathbb{N} : \exists\, t \in \mathcal{H} : \bar{t}[j+1] \in \{\bar{t}[j]\} \succsim \{t\}$
PROOF: By assumption ⟨9⟩1 and definition 37

⟨12⟩2. $\bar{t} \in posCh(p\bar{o}_1)$
PROOF: By assumption ⟨10⟩2, ⟨12⟩1 and definition 37

⟨12⟩3. Q.E.D.
PROOF: By ⟨12⟩2

⟨11⟩2. Q.E.D.
PROOF: By ⟨11⟩1

⟨10⟩3. CASE: $\exists\, j \in \mathbb{N} : \bar{t}[j] \in neg.p\bar{o}_1[j]$

⟨11⟩1. LET: $j \in \mathbb{N}$ s.t. $\bar{t}[j] \in neg.p\bar{o}_1[j]$
PROOF: By assumption ⟨10⟩3

⟨11⟩2. $\sqcup \bar{t} \subseteq \bigcup_{\bar{t}' \in negCh(p\bar{o}_1)} \sqcup \bar{t}'$

⟨12⟩1. $\bar{t} \in negCh(p\bar{o}_1)$

⟨13⟩1. $\forall\, g \in \mathbb{N} : \exists\, t \in \mathcal{H} : \bar{t}[g+1] \in \{\bar{t}[g]\} \succsim \{t\}$
PROOF: By assumption ⟨9⟩1 and definition 37

⟨13⟩2. $\exists\, i \in \mathbb{N} : \forall\, g \in \mathbb{N} : \bar{t}[g] \in neg.p\bar{o}_1[g+i-1]$

⟨14⟩1. $\forall\, g \in \mathbb{N} : \bar{t}[g] \in neg.p\bar{o}_1[g+j-1]$

⟨15⟩1. $\bar{t}[1+j-1] \in neg.p\bar{o}_1[j]$ (base case)
PROOF: By ⟨11⟩1

⟨15⟩2. ASSUME: $\bar{t}[k+j-1] \in neg.p\bar{o}_1[k+j-1]$ (ind. hyp.)
PROVE: $\bar{t}[k+j] \in neg.p\bar{o}_1[k+j]$

⟨16⟩1. $\bar{t}[k+j] \in pos.\bar{po}[k+j]$
PROOF: By assumption ⟨9⟩1 and definition 37

⟨16⟩2. $\bar{t}[k+j] \in pos.p\bar{o}_1[k+j] \cup neg.p\bar{o}_1[k+j]$
PROOF: By ⟨5⟩2, ⟨16⟩1 and definition 34

⟨16⟩3. CASE: $\bar{t}[k+j] \in neg.p\bar{o}_1[k+j]$

⟨17⟩1. Q.E.D.
PROOF: By assumption ⟨16⟩3

⟨16⟩4. CASE: $\bar{t}[k+j] \in pos.p\bar{o}_1[k+j]$

⟨17⟩1. LET: $s = \{\bar{t}[k+j-1]\} \gtrsim \mathcal{H}$

⟨17⟩2. $\bar{t}[k+j] \in s$

PROOF: By assumption ⟨9⟩1, definition 37 and ⟨17⟩1

⟨17⟩3. $s \subseteq neg.p\bar{o}_1[k+j] \cup (\mathcal{H} \setminus (pos.p\bar{o}_1[k+j] \cup neg.p\bar{o}_1[k+j]))$

PROOF: By assumption ⟨15⟩2, ⟨5⟩2 ($p\bar{o}_1 \in chains(O')$) and definition 36

(Step ⟨17⟩3 states that all traces in s are either negative or inconlusive in $p\bar{o}_1[k+j]$. Intuitively, the reason for this is that the trace $\bar{t}[k+j-1]$ is negative in $p\bar{o}_1[k+j-1]$. Note that a trace may be both positive and negative in $p\bar{o}_1[k+j]$, so ⟨17⟩3 does not contradict assumption ⟨16⟩4.)

⟨17⟩4. $\bar{t}[k+j] \in neg.p\bar{o}_1[k+j] \cup (\mathcal{H} \setminus (pos.p\bar{o}_1[k+j] \cup neg.p\bar{o}_1[k+j]))$

PROOF: By ⟨17⟩2 and ⟨17⟩3

⟨17⟩5. $\bar{t}[k+j] \notin \mathcal{H} \setminus (pos.p\bar{o}_1[k+j] \cup neg.p\bar{o}_1[k+j])$

PROOF: By assumption ⟨16⟩4

⟨17⟩6. Q.E.D.

PROOF: By ⟨17⟩4 and ⟨17⟩5

⟨16⟩5. Q.E.D.

PROOF: By ⟨16⟩2 the cases ⟨16⟩3 and ⟨16⟩4 are exhaustive

⟨15⟩3. Q.E.D.

PROOF: Induction with ⟨15⟩1 as base case and ⟨15⟩2 as induction step

⟨14⟩2. Q.E.D.

PROOF: By ⟨14⟩1; j is the i we are looking for

⟨13⟩3. Q.E.D.

PROOF: By ⟨13⟩1, ⟨13⟩2 and definition 38

⟨12⟩2. Q.E.D.

PROOF: By ⟨12⟩1

⟨11⟩3. Q.E.D.

PROOF: By ⟨11⟩2

⟨10⟩4. Q.E.D.

PROOF: By ⟨10⟩1, the cases ⟨10⟩2 and ⟨10⟩3 are exhaustive

⟨9⟩2. Q.E.D.

PROOF: ∀-rule

⟨8⟩2. Q.E.D.

PROOF: By ⟨8⟩1

⟨7⟩2. Q.E.D.

PROOF: By ⟨7⟩1 and definition 41

⟨6⟩3. $\pi_2.\sqcup \bar{po}_1 \subseteq \pi_2.\sqcup \bar{po}$

⟨7⟩1. $\{\lim_{j\to\infty} \bar{q}_1 \mid \bar{q}_1 \in prob(\bar{po}_1)\} \subseteq \{\lim_{j\to\infty} \bar{q} \mid \bar{q} \in prob(\bar{po})\}$

⟨8⟩1. ASSUME: $q \in \{\lim_{j\to\infty} \bar{q}_1 \mid \bar{q}_1 \in prob(\bar{po}_1)\}$

PROVE: $q \in \{\lim_{j\to\infty} \bar{q} \mid \bar{q} \in prob(\bar{po})\}$

⟨9⟩1. LET: $\bar{q}_1 \in prob(\bar{po}_1)$ s.t. $q = \lim_{j\to\infty} \bar{q}_1$

PROOF: By assumption ⟨8⟩1

⟨9⟩2. $\bar{q}_1 \in prob(\bar{po})$

⟨10⟩1. $\forall j \in \mathbb{N} : \bar{q}_1[j] \in \pi_2.\bar{po}_1[j] \wedge \exists q \in [0..1] : \bar{q}_1[j+1] \in \bar{q}_1[j] * q$

PROOF: By ⟨9⟩1 and definition 39

⟨10⟩2. $\forall j \in \mathbb{N} : \pi_2.\bar{po}_1[j] \subseteq \pi_2.\bar{po}[j]$

PROOF: By ⟨5⟩2

⟨10⟩3. $\forall j \in \mathbb{N} : \bar{q}_1[j] \in \pi_2.\bar{po}[j] \wedge \exists q \in [0..1] : \bar{q}_1[j+1] \in \bar{q}_1[j] * q$

PROOF: By ⟨10⟩1 and ⟨10⟩2

⟨10⟩4. Q.E.D.

PROOF: By ⟨10⟩3 and definition 39

⟨9⟩3. Q.E.D.

PROOF: By ⟨9⟩1 and ⟨9⟩2

⟨8⟩2. Q.E.D.

PROOF: $\subseteq$-rule

⟨7⟩2. Q.E.D.

PROOF: By ⟨7⟩1 and definition 41

⟨6⟩4. Q.E.D.

PROOF: By ⟨6⟩1, ⟨6⟩2, ⟨6⟩3 and definition 34

⟨5⟩4. Q.E.D.

PROOF: By ⟨5⟩2 and ⟨5⟩3; $\bar{po}_1$ is the $\bar{po}'$ we are looking for.

⟨4⟩2. Q.E.D.

PROOF: $\Rightarrow$-rule

⟨3⟩2. Q.E.D.

PROOF: $\forall$-rule

⟨2⟩2. Q.E.D.

PROOF: By ⟨2⟩1, assumption ⟨1⟩4, definition 45 and definition 65

⟨1⟩5. Q.E.D.

PROOF: By the overall assumption, the cases ⟨1⟩1, ⟨1⟩2, ⟨1⟩3 and ⟨1⟩4 are exhaustive.

□

Theorem 8. *Monotonicity of* $\biguplus_{i\in I} \mu_i$ *w.r.t.* $\rightsquigarrow$

LET: $[\![d]\!] = O, [\![d']\!] = O'$

ASSUME: 1. $O \rightsquigarrow O'$

2. $I \subseteq \mathbb{N}_0 \cup \{\infty\} \wedge I \neq \varnothing$

PROVE: $\biguplus_{i\in I} \mu_i O \rightsquigarrow \biguplus_{i\in I} \mu_i O'$

⟨1⟩1. $\biguplus_{i\in I\setminus\{\infty\}} \mu_i O \rightsquigarrow \biguplus_{i\in I\setminus\{\infty\}} \mu_i O'$

⟨2⟩1. LET: $\bar{I}$ be an ordering of the elements in $I \setminus \{\infty\}$

PROOF: $I \setminus \{\infty\} \subseteq \mathbb{N}_0$

⟨2⟩2. $\biguplus_{i\in\{1\}} \mu_{\bar{I}[i]} O \rightsquigarrow \biguplus_{i\in\{1\}} \mu_{\bar{I}[i]} O'$ (induction basis)

PROOF: By Lemma 9, since $\biguplus_{i\in\{n\}} \mu_{\bar{I}[i]} O = \mu_{\bar{I}[n]} O$ according to definition 47

⟨2⟩3. ASSUME: $\biguplus_{i\in\{1,\ldots,k\}} \mu_{\bar{I}[i]} O \rightsquigarrow \biguplus_{i\in\{1,\ldots,k\}} \mu_{\bar{I}[i]} O'$ (induction hypothesis)

PROVE: $\biguplus_{i\in\{1,\ldots,k+1\}} \mu_{\bar{I}[i]} O \rightsquigarrow \biguplus_{i\in\{1,\ldots,k+1\}} \mu_{\bar{I}[i]} O'$

⟨3⟩1. $\mu_{\bar{I}[k+1]} O \rightsquigarrow \mu_{\bar{I}[k+1]} O'$

PROOF: By Lemma 9

⟨3⟩2. Q.E.D.

PROOF: By assumption ⟨2⟩3, ⟨3⟩1 and Theorem 5

⟨2⟩4. Q.E.D.

PROOF: Induction with ⟨2⟩2 as basis and ⟨2⟩3 as induction step

⟨1⟩2. CASE: $\infty \notin I$

⟨2⟩1. Q.E.D.

PROOF: By ⟨1⟩1

⟨1⟩3. CASE: $\infty \in I$

⟨2⟩1. $\mu_\infty O \rightsquigarrow \mu_\infty O'$

PROOF: By Lemma 9

⟨2⟩2. Q.E.D.

PROOF: By ⟨1⟩1, ⟨2⟩1 and Theorem 5

⟨1⟩4. Q.E.D.

PROOF: The cases ⟨1⟩2 and ⟨1⟩3 are exhaustive

□

Monotonicity of loop w.r.t. $\rightsquigarrow$ follows immediately from Theorem 8.

B.3 Convergence of probabilities for loop

In this section we show that if O is a set of p-obligations with non-empty probability sets and $\bar{po} \in \mathit{chains}(O)$, then the p-obligation $\sqcup\bar{po}$ (see definition 41) has a non-empty probability set. From definition 45 we see that this ensures that probability set of specifications with infinite loop will not in general be empty.

Lemma 10. *Every set of positive real numbers with an upper bound in* $\mathbb{R}$ *has a least upper bound in* $\mathbb{R}$*.*

ASSUME: S is a non-empty set of positive real numbers with an upper bound in $\mathbb{R}$, i.e.

1. $S \subseteq \mathbb{R}^+$
2. $S \neq \varnothing$

3. $\exists\, r \in \mathbb{R} : \forall\, s \in S : r \geq s$

PROVE: $\exists\, q \in \mathbb{R} : \forall\, s \in S : q \geq s \wedge \forall\, r \in \mathbb{R} : r < q \Rightarrow \exists\, s' \in S : s' > r$
i.e. S has a least upper bound q in $\mathbb{R}$

PROOF: This is a well known result. A proof can be found in most books on real analysis, for example in [DD02], p. 46-47.

□

Theorem 9. *Convergence of probabilities.*
ASSUME: $\bar{s} \in \mathit{prob}(\bar{po})$ where $\bar{po} \in \mathit{chains}(O)$ and O is a set of p-obligations with non-empty probability sets.
PROVE: $\exists\, k \in [0...1] : \lim_{j \to \infty} \bar{s}[j] = k$

$\langle 1 \rangle 1$. $\exists\, k \in [0...1] : \forall\, \epsilon > 0 : \exists\, i \in \mathbb{N} : \forall\, j > i : \bar{s}[j] - k \leq \epsilon$
  $\langle 2 \rangle 1$. LET: $U = \{r \in [0...1] \mid \forall\, j \in \mathbb{N} : r \leq \bar{s}[j]\}$
  $\langle 2 \rangle 2$. $\exists\, \mathit{lub} \in [0...1] : \forall\, u \in U : \mathit{lub} \geq u \wedge$
    $\forall\, r \in \mathbb{R} : (r < \mathit{lub} \Rightarrow \exists\, u' \in U : u' > r)$
    i.e. U has a least upper bound lub in $[0...1]$.
    $\langle 3 \rangle 1$. $\exists\, \mathit{lub}' \in \mathbb{R} : \forall\, u \in U : \mathit{lub}' \geq u \wedge$
      $\forall\, r \in \mathbb{R} : (r < \mathit{lub}' \Rightarrow \exists\, u' \in U : u' > r)$,
      i.e. U has a least upper bound lub' in $\mathbb{R}$.
      $\langle 4 \rangle 1$. $\exists\, x \in \mathbb{R} : \forall\, u \in U : x \geq u$,
        i.e. U has an upper bound in $\mathbb{R}$
        PROOF: By $\langle 2 \rangle 1$, every $\bar{s}[j]$ is an upper bound for U.
      $\langle 4 \rangle 2$. Q.E.D.
        PROOF: By $\langle 4 \rangle 1$ and Lemma 10.
    $\langle 3 \rangle 2$. LET: $\mathit{lub}' \in \mathbb{R}$ s.t. $\forall\, u \in U : \mathit{lub}' \geq u \wedge$
      $\forall\, r \in \mathbb{R} : (r < \mathit{lub}' \Rightarrow \exists\, u' \in U : u' > r)$
      PROOF: By $\langle 3 \rangle 1$
    $\langle 3 \rangle 3$. $\mathit{lub}' \in U$
      $\langle 4 \rangle 1$. ASSUME: $\mathit{lub}' \notin U$
        PROVE: $\bot$
        $\langle 5 \rangle 1$. CASE: $\mathit{lub}' > 1$
          $\langle 6 \rangle 1$. LET: $q \in \mathbb{R}$ s.t. $1 < q < \mathit{lub}'$
            PROOF: By assumption $\langle 5 \rangle 1$
          $\langle 6 \rangle 2$. $\exists\, u' \in U : u' > q$
            $\langle 7 \rangle 1$. $\forall\, u \in U : \mathit{lub}' > u$
              PROOF: By assumption $\langle 5 \rangle 1$ and $\langle 2 \rangle 1$
            $\langle 7 \rangle 2$. Q.E.D.
              PROOF: By $\langle 7 \rangle 1$ and $\langle 3 \rangle 2$ (insert q for r in second conjunct)
          $\langle 6 \rangle 3$. $\exists\, u' \in U : u' > 1$
            PROOF: By $\langle 6 \rangle 2$ and $\langle 6 \rangle 1$
          $\langle 6 \rangle 4$. Q.E.D.
            PROOF: By $\langle 6 \rangle 3$ and $\langle 2 \rangle 1$
        $\langle 5 \rangle 2$. CASE: $\mathit{lub}' \leq 1$
          $\langle 6 \rangle 1$. $\exists\, j \in \mathbb{N} : \bar{s}[j] < \mathit{lub}'$
            $\langle 7 \rangle 1$. $\mathit{lub}' \in [0, 1]$

PROOF: By ⟨3⟩2 ($lub' \in \mathbb{R}$) and assumption ⟨5⟩2

⟨7⟩2. $\forall\, r \in [0,1] : r \notin U \Rightarrow \exists\, j \in \mathbb{N} : \bar{s}[j] < r$

PROOF: By ⟨2⟩1

⟨7⟩3. Q.E.D.

PROOF: By ⟨7⟩1, ⟨7⟩2 and assumption ⟨4⟩1

⟨6⟩2. LET: $j \in \mathbb{N}$ s.t. $\bar{s}[j] < lub'$

PROOF: By ⟨6⟩1

⟨6⟩3. $\forall\, u \in U : u \leq \bar{s}[j]$,
i.e. $\bar{s}[j]$ is an upper bound for U

PROOF: By ⟨2⟩1

⟨6⟩4. $\exists\, u' \in U : u' > \bar{s}[j]$

PROOF: By ⟨6⟩2, $\bar{s}[j] < lub'$. ⟨6⟩4 then follows by the second conjunct of ⟨3⟩2.

⟨6⟩5. Q.E.D.

PROOF: By ⟨6⟩3 and ⟨6⟩4

⟨5⟩3. Q.E.D.

PROOF: The cases ⟨5⟩1 and ⟨5⟩2 are exhaustive

⟨4⟩2. Q.E.D.

PROOF: $\bot$-rule

⟨3⟩4. $lub' \in [0,1]$

PROOF: By ⟨3⟩3 and ⟨2⟩1

⟨3⟩5. Q.E.D.

PROOF: By ⟨3⟩2 and ⟨3⟩4

⟨2⟩3. LET: $lub \in [0...1]$ s.t. $\forall\, u \in U : lub \geq u \wedge$
$\forall\, r \in \mathbb{R} : (r < lub \Rightarrow \exists\, u' \in U : u' > r)$

PROOF: By ⟨2⟩2

⟨2⟩4. $\forall\, \epsilon > 0 : \exists\, i \in \mathbb{N} : \forall\, j > i : \bar{s}[j] - lub < \epsilon$

⟨3⟩1. ASSUME: $\epsilon > 0$
PROVE: $\exists\, i \in \mathbb{N} : \forall\, j > i : \bar{s}[j] - lub < \epsilon$

⟨4⟩1. ASSUME: $\neg(\exists\, i \in \mathbb{N} : \forall\, j > i : \bar{s}[j] - lub < \epsilon)$
PROVE: $\bot$

⟨5⟩1. $\exists\, m \in \mathbb{N} : \bar{s}[m] - lub < \epsilon$

⟨6⟩1. ASSUME: $\forall\, n \in \mathbb{N} : \bar{s}[n] - lub \geq \epsilon$
PROVE: $\bot$

⟨7⟩1. $\epsilon + lub \in U$

⟨8⟩1. $\forall\, n \in \mathbb{N} : \epsilon + lub \leq \bar{s}[n]$

PROOF: By assumption ⟨6⟩1 and assumption ⟨3⟩1

⟨8⟩2. $\epsilon + lub \in [0,1]$

⟨9⟩1. $\epsilon + lub > 0$

PROOF: By assumption ⟨3⟩1 and ⟨2⟩3 ($lub \in [0,1]$)

⟨9⟩2. $\epsilon + lub \leq 1$

⟨10⟩1. $\forall\, n \in \mathbb{N} : \bar{s}[n] \leq 1$

PROOF: By the overall assumption

⟨10⟩2. Q.E.D.

PROOF: By ⟨10⟩1 and ⟨8⟩1

⟨9⟩3. Q.E.D.
PROOF: By ⟨9⟩1 and ⟨9⟩2
⟨8⟩3. Q.E.D.
PROOF: By ⟨8⟩1, ⟨8⟩2 and ⟨2⟩1
⟨7⟩2. $\epsilon + lub > lub$
PROOF: By assumption ⟨3⟩1
⟨7⟩3. $lub \geq \epsilon + lub$
PROOF: By ⟨7⟩1 and ⟨2⟩3 (first conjunct)
⟨7⟩4. Q.E.D.
PROOF: By ⟨7⟩2 and ⟨7⟩3
⟨6⟩2. Q.E.D.
PROOF: $\bot$-rule
⟨5⟩2. LET: $m \in \mathbb{N}$ s.t. $\bar{s}[m] - lub < \epsilon$
PROOF: By ⟨5⟩1
⟨5⟩3. $\forall\, i \in \mathbb{N} : \exists\, j > i : \bar{s}[j] - lub \geq \epsilon$
PROOF: By assumption ⟨4⟩1
⟨5⟩4. LET: $j > m$ s.t. $\bar{s}[j] - lub \geq \epsilon$
PROOF: By ⟨5⟩3
⟨5⟩5. $\bar{s}[j] \leq \bar{s}[m]$
PROOF: By ⟨5⟩4 ($j > m$) and the overall assumption; definition 39 ensures that $j > m \Rightarrow \bar{s}[j] \leq \bar{s}[m]$
⟨5⟩6. $\bar{s}[j] - lub \geq \epsilon \wedge \bar{s}[m] - lub < \epsilon \wedge \bar{s}[j] \leq \bar{s}[m]$
PROOF: $\wedge$-intro ⟨5⟩4, ⟨5⟩2 and ⟨5⟩5
⟨5⟩7. Q.E.D.
PROOF: By ⟨5⟩6
⟨4⟩2. Q.E.D.
PROOF: $\bot$-rule
⟨3⟩2. Q.E.D.
PROOF: $\forall$-rule
⟨2⟩5. Q.E.D.
PROOF: By ⟨2⟩3 ($lub \in [0...1]$) and ⟨2⟩4; lub is the k we are looking for
⟨1⟩2. Q.E.D.
PROOF: By definition of limit

□

C Motivation behind changes

A number of changes have been made to the version presented in [RHS05]. In that version the semantics of a probabilistic sequence diagram was given as a multiset instead of as a set, and the definition of the refinement relation was a bit different. Also, the definition of the alt operator has been changed. We refer to [RHS05] for the definitions. This section is aimed at readers who are familiar with [RHS05] and want to know why changes have been made.
The changes were primarily motivated by non-monotonicity of the operators seq, par and alt w.r.t. $\rightsquigarrow$. The subsections C.1 and C.2 give examples that show the lack of monotonicity when the original definitions as presented in [RHS05] are used. As has been shown in Section B, the current definitions ensure that the refinement relation is now monotonic w.r.t. all these operators. The changes have also generally led to much simpler proofs. In the rest of this section we assume that the original definitions as given in [RHS05] are applied.

C.1 Non-monotonicity of seq and par w.r.t. $\rightsquigarrow$ with definitions from [RHS05].

The seq and par operators are not monotonic w.r.t. the refinement relation when the original definitions from [RHS05] are used. This is shown (for seq) by the following example, where we assume all events occur on the same lifeline. Let

$$
\begin{aligned}
O_1 &= \{((\{\langle a\rangle\},\varnothing),\{0.5\}),((\{\langle ab\rangle\},\varnothing),\{0.5\}),((\varnothing,\varnothing),\{1\})\}\\
O_1' &= \{((\{\langle a\rangle,\langle ab\rangle\},\varnothing),\{0.5\}),((\{\langle d\rangle\},\varnothing),\{0.5\}),((\varnothing,\varnothing),\{1\})\}\\
O_2 &= \{((\{\langle bc\rangle,\langle c\rangle\},\varnothing),\{1\})\}\\
O_2' &= O_2
\end{aligned}
$$

This means that we have

$$
\begin{aligned}
O_1 &\rightsquigarrow O_1'\\
O_2 &\rightsquigarrow O_2'\\
O_1 \succsim O_2 &= \{((\{\langle abc\rangle,\langle ac\rangle\},\varnothing),\{0.5\}),\\
&\quad((\{\langle abbc\rangle,\langle abc\rangle\},\varnothing),\{0.5\}),\\
&\quad((\varnothing,\varnothing),\{1\})\}\\
O_1' \succsim O_2' &= \{((\{\langle abc\rangle,\langle ac\rangle,\langle abbc\rangle\},\varnothing),\{0.5\}),\\
&\quad((\{\langle dbc\rangle,\langle dc\rangle\},\varnothing),\{0.5\}),\\
&\quad((\varnothing,\varnothing),\{1\})\}
\end{aligned}
$$

Now let

$$
S = \{((\{\langle abc\rangle,\langle ac\rangle\},\varnothing),\{0.5\}),((\{\langle abbc\rangle,\langle abc\rangle\},\varnothing),\{0.5\})\}
$$

which means that

$$\begin{aligned} S &\subseteq O_1 \succsim O_2 \\ \bar{\oplus} S &= ((\{\langle abc \rangle\}, \varnothing), \{1\}) \\ 0 &\notin \pi_2.\bar{\oplus} S \end{aligned}$$

But there is no $S' \subseteq O_1' \succsim O_2'$ such that $\bar{\oplus} S \rightsquigarrow \bar{\oplus} S'$.
This counter example exploit the following facts:

- Both the two leftmost p-obligations in $O_1 \succsim O_2$ are refined by the same p-obligation in $O_1' \succsim O_2'$.
- In $O_1 \succsim O_2$ the trace $\langle abc \rangle$ is inconclusive in the p-obligation with probability 1 even though it is positive in both the other p-obligations.
- $\bar{\oplus}(S_1 \succsim S_2) = \bar{\oplus} S_1 \succsim \bar{\oplus} S_2$ does not generally hold. Therefore, we cannot deduce $\bar{\oplus}(S_1 \succsim S_2) \rightsquigarrow \bar{\oplus} S_1' \succsim \bar{\oplus} S_2'$ from $\bar{\oplus} S_1 \rightsquigarrow \bar{\oplus} S_1' \wedge \bar{\oplus} S_1 \rightsquigarrow \bar{\oplus} S_1'$. Letting $S_1 = \{((\{\langle a \rangle\}, \varnothing), \{0.5\}), ((\{\langle ab \rangle\}, \varnothing), \{0.5\})\}$ and $S_2 = O_2$ we see that $\langle abc \rangle \in pos.\bar{\oplus}(S_1 \succsim S_2)$ but $\langle abc \rangle \notin pos.(\bar{\oplus} S_1 \succsim \bar{\oplus} S_2)$.

A similar example can be given for par by replacing $\succsim$ with $\|$.

C.2 Non-monotonisity of alt w.r.t. $\rightsquigarrow$ with definitions from [RHS05]

The alt operator is not monotonic w.r.t. the refinement relation when the original definitions from [RHS05] are used. This is shown by the following example: Let

$$\begin{aligned} [\![\ d_1\ ]\!] &= \{(o_1, [0, ..., 1]), (o_2, [0, ..., 1]), (\oplus\{o_1, o_2\}, \{1\})\} \\ [\![\ d_1'\ ]\!] &= \{(\oplus\{o_1, o_2\}, \{1\})\} \\ [\![\ d_2\ ]\!] &= \{(o_3, \{0.5\}), (o_4, \{0.5\}), (\oplus\{o_3, o_4\}, \{1\})\} \\ [\![\ d_2'\ ]\!] &= [\![\ d_2\ ]\!] \end{aligned}$$

We now have $d_1 \rightsquigarrow d_1'$. (The two first p-obligations in $[\![\ d_1\ ]\!]$ need not be represented at the concrete level, since 0 is an allowed probability.) We also have $d_2 \rightsquigarrow d_2'$.
However, $d_1\ \mathsf{alt}\ d_2 \rightsquigarrow d_1'\ \mathsf{alt}\ d_2'$ does not hold. To see this, note that $[\![\ d_1\ \mathsf{alt}\ d_2\ ]\!] =$

$$\begin{aligned} &\{(o_1 \uplus o_3, \{0.5\}), (o_1 \uplus o_4, \{0.5\}), (o_1 \uplus \oplus\{o_3, o_4\}, \{1\}) \\ &(o_2 \uplus o_3, \{0.5\}), (o_2 \uplus o_4, \{0.5\}), (o_2 \uplus \oplus\{o_3, o_4\}, \{1\}) \\ &(\oplus\{o_1, o_2\} \uplus o_3, \varnothing), (\oplus\{o_1, o_2\} \uplus o_4, \varnothing), (\oplus\{o_1, o_2\} \uplus \oplus\{o_3, o_4\}, \{1\})\} \end{aligned}$$

while $[\![\ d_1'\ \mathsf{alt}\ d_2'\ ]\!] =$

$$\{(\oplus\{o_1, o_2\} \uplus o_3, \varnothing), (\oplus\{o_1, o_2\} \uplus o_4, \varnothing), (\oplus\{o_1, o_2\} \uplus \oplus\{o_3, o_4\}, \{1\})\}$$

Now let

$$\begin{aligned} o_1 &= (\{t_1\}, \varnothing) \\ o_2 &= (\{t_2\}, \varnothing) \\ o_3 &= (\{t_3\}, \varnothing) \\ o_4 &= (\{t_4\}, \varnothing) \end{aligned}$$

and let

$$S = \{(o_1 \uplus o_3, \{0.5\})\}$$

Then $S \subseteq [\![\ d_1 \ \mathsf{alt}\ d_2\ ]\!]$ and $0 \notin \pi_2.\bar{\oplus}S$, but there is no $S' \subseteq [\![\ d_1' \ \mathsf{alt}\ d_2'\ ]\!]$ such that $\bar{\oplus}S \rightsquigarrow \bar{\oplus}S'$. This is clear from the fact that $\bar{\oplus}S = ((\{t_1, t_3\}, \varnothing), \{0.5\})$ while $\oplus\{o_1, o_2\} \uplus o_3 = (\{t_3\}, \varnothing)$, $\oplus\{o_1, o_2\} \uplus o_4 = (\{t_4\}, \varnothing)$ and $\oplus\{o_1, o_2\} \uplus \oplus\{o_3, o_4\} = (\varnothing, \varnothing)$. Therefore t_1 will be inconclusive at the concrete level for any combination of p-obligations from $[\![\ d_1' \ \mathsf{alt}\ d_2'\ ]\!]$.
We may also note that

$$\bar{\oplus}O_1 \uplus \bar{\oplus}O_2 \rightsquigarrow \bar{\oplus}O_1' \uplus \bar{\oplus}O_2' \not\rightsquigarrow \bar{\oplus}(O_1 \uplus O_2) \rightsquigarrow \bar{\oplus}(O_1' \uplus O_2')$$

for arbitrary sets of p-obligations O_1, O_1', O_2 and O_2'. To see this, let

$$\begin{aligned} O_1 &\stackrel{\mathsf{def}}{=} \{(o_a, \{0.7\}), (o_b, \{0.1\})\} \\ O_2 &\stackrel{\mathsf{def}}{=} \{(o_c, \{0.6\}), (o_d, \{0.2\})\} \\ O_1' &\stackrel{\mathsf{def}}{=} \{(o_e, \{0.8\})\} \\ O_2' &\stackrel{\mathsf{def}}{=} \{(o_f, \{0.8\})\} \end{aligned}$$

This means that

$$\sum_{(o_1, Q_1) \in O_1} Q_1 = \sum_{(o_2, Q_2) \in O_2} Q_2 = \sum_{(o_1', Q_1') \in O_1'} Q_1' = \sum_{(o_2', Q_2') \in O_2'} Q_2' = \{0.8\}$$

and therefore

$$\sum_{(o_1', Q_1') \in O_1'} Q_1' \cap \sum_{(o_2', Q_2') \in O_2'} Q_2' \subseteq \sum_{(o_1, Q_1) \in O_1} Q_1 \cap \sum_{(o_2, Q_2) \in O_2} Q_2$$

which gives

$$\pi_2.\bar{\oplus}O_1' \uplus \bar{\oplus}O_2' \subseteq \pi_2.\bar{\oplus}O_1 \uplus \bar{\oplus}O_2$$

At the same time we have

$$\begin{aligned} \sum_{(o', Q') \in O_1' \uplus O_2'} Q' &= \{0.8\} \\ \sum_{(o, Q) \in O_1 \uplus O_2} Q &= \varnothing \end{aligned}$$

which means that

$$\sum_{(o',Q')\in O_1'\uplus O_2'} Q' \not\subseteq \sum_{(o,Q)\in O_1\uplus O_2} Q$$

and therefore

$$\pi_2.\bar{\oplus}(O_1'\uplus O_2') \not\subseteq \pi_2.\bar{\oplus}(O_1\uplus O_2)$$