Reasoning about QoS Contracts in the Probabilistic Duration Calculus

Dimitar P. Guelev*and Dang Van Hung†

February 24, 2008

Abstract

The notion of *contract* was introduced to component-based software development in order to facilitate the semantically correct composition of components. We extend the form of this notion which is based on *designs* to capture probabilistic requirements on execution time. We show how reasoning about such requirements can be done in an infinite-interval-based system of probabilistic duration calculus.

Introduction

Combining off-the-shelf and dedicated components has become an established approach to achieving reuse, modularity, productivity and reliability. *Contracts* facilitate the correct use of components. A contract is a collection of requirements which are written in terms of the component interface. Contract requirements should be satisfied by implementations of the component, provided that the items imported from other components also satisfy the requirements appearing in the contract for them. Four levels of contracts have been identified in [BJPW99]. These are the *syntactical* level, the *behavioural* level, the *synchronization* level and the *quality of service* level. Quality of Service (QoS) is a collective term for non-functional requirements such as worst-case and average execution time, and the consumption of resources such as memory, power, bandwidth, etc.

Component models are built around appropriate formalisations of the notions of *interface*, *contract*, *component composability*, *composition*, etc. A contract theory for components based on the notion of *design* from [HH98] has been proposed in [HLL06, HXZ06] and has become known as the rCOS model. Since designs capture input-output relations, this model is mostly about the *functional* requirements on components and leaves out the QoS level from [BJPW99]. In our previous work we extended the rCOS model to capture requirements on timing and resources [Dan05, HD07]. We have considered *hard* requirements, where, e.g., missing a deadline is regarded as fatal. We used the Duration Calculus (DC) as our notation. QoS is mainly concerned with *soft* requirements, where, e.g., missing a deadline by little and not too often is tolerable. Handling requirements on the QoS involves reasoning about probability.

In this paper we extend designs to capture probabilistic requirements on execution time and develop a technique to reason about QoS of real-time embedded systems using an infinite-interval-based system of probabilistic DC (PDC) which was proposed in [Gue07] as an extension of a corresponding system of Probabilistic Interval Temporal Logic with infinite intervals ($PITL$). PDC with infinite intervals subsumes the systems of PDC from [LRSZ93, DZ99, Gue00b] and has a relatively complete proof system to support formal reasoning. The fitness of (non-probabilistic) DC for reasoning about real-time systems has been asserted by numerous case studies [ZZ94, DW96, SX98, Dan98, LH99]. Since DC is interval-based, reasoning about the behaviour of whole method executions, including their execution time, is relatively straightforward in DC. By using a probabilistic extension of DC we are able to enjoy this advantage when reasoning about QoS requirements too.

*Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Bl. 8, Acad G. Bonchev str. 1113 Sofia, Bulgaria, fax: +359 2 9713649, e-mail gelevdp@math.bas.bg. Work on this paper was done during the D. Guelev's visit to UNU/IIST in August-September 2007.

†International Institute of Software Technology of the United Nations University (UNU/IIST), Est. do Engenheiro Trigo No. 4, P.O. Box 3058, Macao SAR, China, fax: +853 2871 2940, e-mail: dvh@iist.unu.edu

1 Preliminaries

We consider only the extended set of the real numbers $\overline{\mathbf{R}} = \mathbf{R} \cup \{\infty\}$ as the flow of time in *PITL* and *PDC*. In order to facilitate the description of repetitive behaviour, we include a least-fixed-point operator for non-probabilistic formulas, which was introduced in [Pan95] and studied in [Gue00a]. *ITL with infinite intervals* [ZDL95, PWX98, SX98, WX04] is the underlying non-probabilistic logic of *PITL* and *PDC*. It extends the syntax of predicate logic by a binary modality $(.;.)$, known as *chop*.[1] Non-logical symbols are divided into *rigid* and *flexible* depending on whether their meaning is required to be the same at all reference intervals or not. Individual variables are rigid.

An *interpretation of a vocabulary* $\mathbf{L}$ is a function I on $\mathbf{L}$ which maps the symbols from $\mathbf{L}$ to members of $\overline{\mathbf{R}}$, functions and predicates on $\overline{\mathbf{R}}$, according to the type and arity of symbols. $I(s)$ takes an interval from $\tilde{\mathbf{I}}$ as an additional argument in case s is flexible. We use the sets of intervals

$$\mathbf{I}^{fin} = \{[\tau_1, \tau_2] : \tau_1, \tau_2 \in \mathbf{R}, \tau_1 \leq \tau_2\}, \mathbf{I}^{inf} = \{[\tau, \infty] : \tau \in \mathbf{R}\} \text{ and } \tilde{\mathbf{I}} = \mathbf{I}^{fin} \cup \mathbf{I}^{inf}.$$

Given $\sigma_1 \in \mathbf{I}^{fin}$ and $\sigma_2 \in \tilde{\mathbf{I}}$ such that $\max \sigma_1 = \min \sigma_2$, $\sigma_1; \sigma_2$ stands for $\sigma_1 \cup \sigma_2$. Given an interpretation I, the values $I_\sigma(t)$ of terms t at intervals $\sigma \in \tilde{\mathbf{I}}$ are defined in the usual way, with the reference interval being an additional argument for flexible symbols. Satisfaction $\models$ is defined with respect to an interpretation and a reference interval. Flexible relation symbols are interpreted predicates which take the reference interval as an argument too. The clauses for $\bot$, $\Rightarrow$ and $\exists$ are the usual ones. The clause for $(.;.)$ is

$$I, \sigma \models (\varphi; \psi) \text{ iff } I, \sigma_1 \models \varphi \text{ and } I, \sigma_2 \models \psi \text{ for some } \sigma_1 \in \mathbf{I}^{fin} \text{ and } \sigma_2 \in \tilde{\mathbf{I}} \text{ such that } \sigma_1; \sigma_2 = \sigma.$$

0, ∞, $+$ and $=$ are mandatory in *ITL* vocabularies and always have the usual interpretation. A mandatory flexible constant ℓ always evaluates to the length of the reference interval. *Infix* notation for $+$ and $=$ and $\top$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$ and $\forall$ are used in the usual way. *ITL*-specific abbreviations include

$$(\varphi_1; \ldots; \varphi_{n-1}; \varphi_n) \rightleftharpoons (\varphi_1; \ldots (\varphi_{n-1}; \varphi_n) \ldots), \ \Diamond\varphi \rightleftharpoons (\top; \varphi; \top) \vee (\top; \varphi), \ \Box\varphi \rightleftharpoons \neg\Diamond\neg\varphi.$$

$\Diamond$ and $\Box$ bind more tightly and $(.;.)$ binds less tightly than the boolean connectives.

A complete proof system for *ITL* with infinite intervals with respect to an appropriate abstract domain of durations was presented in [WX04].

Vocabularies for *DC* with infinite intervals additionally include *state variables* $P, Q, \ldots$; *state expressions* S are boolean combinations of state variables with the logical constants written as $\mathbf{0}$ and $\mathbf{1}$ and in turn appear as the argument of *duration terms* $\int S$, which are the *DC*-specific construct in the syntax of *DC* terms. Formulas in *DC* are as in *ITL*. State variables evaluate to piece-wise constant functions of type $\overline{\mathbf{R}} \to \{0,1\}$. The value $I_\tau(S)$ of state expression S at time τ is defined using $I(P)(\tau)$ for the involved state variables P in the usual way. Values of duration terms are defined by the equality

$$I_\sigma(\textstyle\int S) = \displaystyle\int_{\min \sigma}^{\max \sigma} I_\tau(S) d\tau$$

$I_\sigma(\int S)$ can be ∞ for $\sigma \in \mathbf{I}^{inf}$. The expression $\lceil S \rceil$ abbreviates $\ell \neq 0 \wedge \int \neg S = 0$ and ℓ can be viewed as an abbreviation for $\int \mathbf{1}$ in *DC*.

Axioms and rules for *DC* (with infinite intervals) which are complete relative to validity in real-time *ITL* (with infinite intervals), have been presented in [HZ92] (resp. [Gue07].)

The least-fixed-point operator If $\varphi_1, \ldots, \varphi_n$ have no negative occurrences of the propositional variables $X_1, \ldots, X_n$ and $i \in \{1, \ldots, n\}$, then $\mu_i X_1 \ldots X_n.\varphi_1, \ldots, \varphi_n$ is well-formed and $I, \sigma \models \mu_i X_1 \ldots X_n.\varphi_1, \ldots, \varphi_n$ iff $\sigma \in A_i$, where $A_1, \ldots, A_n$ are the smallest subsets of $\tilde{I}$ which satisfy the equalities

$$A_i = \{\sigma \in \tilde{\mathbf{I}} : I_{X_1, \ \ldots \ ,X_n}^{\lambda\sigma.\sigma\in A_1, \ldots, \lambda\sigma.\sigma\in A_n}, \sigma \models \varphi_i\}, \ i = 1, \ldots, n.$$

Iteration, also known as Kleene star, can be defined using μ by the equivalence $\varphi^* \rightleftharpoons \mu_1 X.\ell = 0 \vee (\varphi; X)$. $I, \sigma \models \varphi^*$ can be defined independently by the condition:

$$\min \sigma = \max \sigma \text{ or } \sigma = \sigma_1; \ldots; \sigma_n \text{ and } I, \sigma_i \models \varphi, \ i = 1, \ldots, n, \text{ for some } n < \omega, \sigma_1, \ldots, \sigma_n \in \tilde{\mathbf{I}}.$$

[1] Many authors write chop as $\varphi^\frown\psi$ instead of $(\varphi; \psi)$.

Axioms and rules for μ and * in *DC* were proposed in [Pan95, Gue00a, GD05].

Higher-order quantifiers We use $\exists$ on flexible constants and state variables with the usual meaning, in order to describe the semantics of local variables. The deductive power of some axioms and rules for this usage has been studied in [ZGZ00, Gue00a, GD05].

Probabilistic ITL and DC with infinite intervals (PITL) extends the syntax of *ITL* terms by *probability terms* of the form $p(\varphi)$ where φ is a formula. Formula syntax is as in *ITL*, with μ and higher-order quantifiers included. A *PITL* model is based on a collection of interpretations of a given vocabulary $\mathbf{L}$. Each interpretation is meant to describe a possible behaviour of the modelled system. Consider a non-empty set $\mathbf{W}$, a function I on $\mathbf{W}$ into the set of the *ITL* interpretations of $\mathbf{L}$ and a function P of type $\mathbf{W}\times\overline{\mathbf{R}}\times 2^{\mathbf{W}}\to[0,1]$. Let I^w and P^w abbreviate $I(w)$ and $\lambda\tau,X.P(w,\tau,X)$, respectively, for all $w\in\mathbf{W}$. I^w and P^w, $w\in\mathbf{W}$, are intended to represent the set of behaviours and the associated probability distributions for every $\tau\in\overline{\mathbf{R}}$ in the *PITL* models for $\mathbf{L}$.

Definition 1 Let $\tau\in\overline{\mathbf{R}}$. Then $w\equiv_\tau v$ iff

$I^w(s)=I^v(s)$ for all rigid symbols $s\in\mathbf{L}$, except possibly the individual variables;

$I^w(s)(\sigma,d_1,\ldots,d_{\#s})=I^v(s)(\sigma,d_1,\ldots,d_{\#s})$ for all flexible $s\in\mathbf{L}$, all $d_1,\ldots,d_{\#s}\in\overline{\mathbf{R}}$ and all $\sigma\in\tilde{\mathbf{I}}$ such that $\max\sigma\leq\tau$;

$P^w(\tau',X)=P^v(\tau',X)$ for all $X\subseteq\mathbf{W}$ and all $\tau'\leq\tau$.

Clearly $\equiv_\tau$ is an equivalence relation on W for all $\tau\in\overline{\mathbf{R}}$. Members of $\mathbf{W}$ which are τ-equivalent model the same behaviour up to time τ. If $\tau_1>\tau_2$, then $\equiv_{\tau_1}\subset\equiv_{\tau_2}$ and $w\equiv_\infty v$ holds iff $P^w=P^v$ and I^w and I^v agree on all symbols, except possibly some individual variables. $[w]_{\equiv_\tau}$ is the set of those $v\in\mathbf{W}$ which represent the probabilistic branching of w from time τ onwards.

Definition 2 A *general PITL model* for $\mathbf{L}$ is a tuple of the form $\langle\mathbf{W},I,P\rangle$ where F, $\mathbf{W}$, I and P are as above and satisfy the following requirements for every $w\in\mathbf{W}$:

• $\mathbf{W}$ *is closed under variants of interpretations.* If $w\in\mathbf{W}$, x is an individual variable from $\mathbf{L}$ and $a\in\overline{\mathbf{R}}$, then there is a $v\in\mathbf{W}$ such that $P^v=P^w$ and $I^v=(I^w)^a_x$, where $(I^w)^a_x$ maps x to a and is the same as I^w on other symbols.

• *The functions P^w are probability measures.* For every $w\in W$ and $\tau\in\overline{\mathbf{R}}$ the function $\lambda X.P^w(\tau,X)$ is a probability measure on the boolean algebra $\langle 2^{\mathbf{W}},\cap,\cup,\emptyset,\mathbf{W}\rangle$. Furthermore $\lambda X.P^w(\tau,X)$ is required to be *concentrated* on $[w]_{\equiv_\tau}$: $P^w(\tau,X)=P^w(\tau,X\cap[w]_{\equiv_\tau})$ for all $X\subseteq\mathbf{W}$.

Informally, the probability for a behaviour in $X\subseteq[w]_{\equiv_\tau}$ to be chosen is $P^w(\tau,X)$. Satisfaction $\models$ is defined in *PITL* with respect to a model $M=\langle\mathbf{W},I,P\rangle$, a $w\in\mathbf{W}$, and a $\sigma\in\tilde{\mathbf{I}}$. If ψ is a sentence, then

$$[\![\psi]\!]_{M,w,\sigma}=\{v\in[w]_{\equiv_{\max\sigma}}:M,v,[\min\sigma,\infty]\models\psi\}.$$

This means that $[\![\psi]\!]_{M,w,\sigma}$ consists of the interpretations v which are $\max\sigma$-equivalent to w and satisfy ψ at the infinite interval starting at $\min\sigma$. In case ψ has free variables $x_1,\ldots,x_n$, $M,v,[\min\sigma,\infty]\models\psi$ should be evaluated with $I^w(x_1),\ldots,I^w(x_n)$ as the values of $x_1,\ldots,x_n$, in order to preserve the intended meaning:

$$[\![\psi]\!]_{M,w,\sigma}=\{v\in[w]_{\equiv_{\max\sigma}}:(\forall v'\in W)(P^{v'}=P^v\wedge I^{v'}=(I^v)^{I^w(x_1),\ldots,I^w(x_n)}_{x_1\ ,\ \ldots\ ,\ x_n}\Rightarrow M,v',[\min\sigma,\infty]\models\psi)\}.$$

Using this notation, term values $w_\sigma(t)$ of probability terms t can be defined by putting

$$w_\sigma(p(\psi))=P^w(\max\sigma,[\![\psi]\!]_{M,w,\sigma}).$$

Values of terms of other forms are defined as in (non-probabilistic) *ITL*.

The probability functions $\lambda X.P^w(\tau,X)$ for $w\in\mathbf{W}$ and $\tau\in T$ in general *PITL* models $M=\langle\mathbf{W},I,P\rangle$ are needed just as much as they provide values for probability terms. That is why we accept structures of the form $\langle\mathbf{W},P,I\rangle$ with their probability functions $\lambda X.P^w(\tau,X)$ be defined just on the (generally smaller) algebras $\langle\{[\![\psi]\!]_{M,w,\sigma}:\psi\in\mathbf{L},\sigma\in\tilde{\mathbf{I}},\max\sigma=\tau\},\cap,\cup,\emptyset,[w]_{\equiv_\tau}\rangle$ as general *PITL* models too.

PITL is a conservative extension of *ITL*. Axioms and a proof rule which extend the proof system for *ITL* with infinite intervals to a system for *PITL* were shown in [Gue07] to be complete with respect to a generalisation of the $\mathbf{R}$-based semantics, where $\overline{\mathbf{R}}$ is replaced by an abstract domain and the probability measures are required to be only finitely aditive.

The probability functions $\lambda X.P^w(\tau, X)$ need not be related to each other in general models for *PITL*, whereas applications typically lead to models with an origin of time $\tau_0 = \min T$ and a distinguished $w_0 \in \mathbf{W}$ such that $[w_0]_{\equiv_{\tau_0}} = \mathbf{W}$ and $\lambda X.P^{w_0}(\tau_0, X)$ can be regarded as the global probability function. Then, given an arbitrary $w \in \mathbf{W}$ and $\tau \in \mathbf{R}$, the probability function $\lambda X.P^w(\tau, X)$ should represent *conditional* probability, the condition being τ-equivalence with w. Hence we should have

$$P^{w_0}(\tau, A) = \int\limits_{w \in [w_0]_{\equiv_\tau}} P^w(\tau', A) d(\lambda X.P^{w_0}(\tau, X)). \tag{1}$$

The following rules enable approximating (1) with arbitrary precision in *PITL* proofs:

$$(\underline{P}) \qquad \frac{\varphi \Rightarrow \neg(\varphi; \ell \neq 0)}{\ell = 0 \wedge p(\varphi \wedge p(\psi) < x; \top) = 0 \Rightarrow p((\varphi; \top) \wedge \psi) \geq x.p(\varphi; \top)}$$

$$(\overline{P}) \qquad \frac{\varphi \Rightarrow \neg(\varphi; \ell \neq 0)}{\ell = 0 \wedge p(\varphi \wedge p(\psi) > x; \top) = 0 \Rightarrow p((\varphi; \top) \wedge \psi) \leq x.p(\varphi; \top)}$$

The proof system for *PITL* from [Gue07] is minimal. Using the abbreviations

$$\varphi_l^h \rightleftharpoons \varphi \wedge \ell \geq l \wedge \ell \leq h \text{ and } [ET_\varphi \in [l, h]]_x \rightleftharpoons \ell = 0 \wedge p(\varphi; \top) = 1 \Rightarrow p(\varphi_l^h; \top) = x,$$

we can write the derived rule

$$(Seq) \qquad \frac{\alpha \Rightarrow \neg(\alpha; \ell \neq 0),\ \beta \Rightarrow \neg(\beta; \ell \neq 0),\ [ET_\alpha \in [l_1, h_1]]_{x_1},\ [ET_\beta \in [l_2, h_2]]_{x_2}}{\ell = 0 \wedge p(\alpha; \beta; \top) = 1 \Rightarrow p(\alpha_{l_1}^{h_1}; \beta_{l_2}^{h_2}; \top) = x_1 x_2},$$

which is particularly important to our examples.

The system of *probabilistic DC (PDC) with infinite intervals* which we use in this paper is obtained by adding state variables and duration terms to *PITL* in the way used to obtain (non-probabilistic) *DC* from *ITL*. The axioms and rules for *DC* with infinite intervals are complete for *PDC* relative to validity in *PITL* models based on $\overline{\mathbf{R}}$.

2 A toy concurrent programming language and its semantics in *DC* with infinite intervals

We propose a toy language to illustrate our approach. It is shaped after that from [GD02] and has restricted form of method call, in order to set the stage for the use of components and contracts.

Programs consist of *components*, which import and/or export *methods*. Their syntax is:

component ::= **component** *name*
  {**import** *method*}*
  {**export** *method*}*
end *name*

method ::= *name*(*parameter list*)[*code*];

The part *code* is required only for exported methods. It has the syntax

code	::=	**stop** |	(thread) termination statement
		return[e] |	return control and possibly a value
		X	continuation
		($x := e$; *code*) |	assignment
		(**delay** r; *code*) |	delay by the specified amount of time
		(**call** [$x :=$]*name*(*parameter list*); *code*) |	call method and possibly obtain a value
		if b **then** *code* **else** *code* |	conditional statement
		letrec *code* **where** X : *code*; ... ; X : *code* |	mutual recursion statement
		var x; *code*	local variable declaration
		code||*code*	parallel composition

We do not allow **var** to occur in the scope of other control statements. Assignments are atomic. Parameters are passed by value. A mutual recursion statement can trigger an infinite computation. Components are passive. The *active* part of a program is just a piece of code, typically a collection of concurrently running interleaved threads. The syntax of control statements deliberately makes *tail-recursion* the only expressible form of repetitive behaviour. We give no details on the type system and tacitly assume an appropriately many-sorted system of DC.

The execution of code can be described in terms of the values of its signals, variables and parameters as functions of time. The semantic function $[\![.]\!]$ defined below maps every piece of code to a DC formula which defines the set of its observable behaviours. We model each program variable v by a corresponding pair of flexible constants v and v', which denote the value of v at the beginning and at the end of the reference interval and therefore satisfy the axiom $\forall x \neg(v' = x; v \neq x)$ where x is a rigid individual variable. We model methods m which return a value by a corresponding flexible function symbol. A formula of the form $v' = m(e_1, \ldots, e_n)$ means that the reference interval describes a complete invocation of m with $e_1, \ldots, e_n$ as the input parameters and v' as the value. We use a flexible predicate symbol for methods which return no value. We use dedicated state variables R and W to indicate that the thread is currently running, or has terminated, respectively. Building on the work from [PD98, GD02], we use a state variable N to mark computation time, which, unlike the time consumed by the execution of *delay* statements, waiting for the reaction of the environment, etc., is regarded as *N*egligible, in order to simplify calculations. R, W and N satisfy the axioms

$$\mathsf{T}(R, W) \rightleftharpoons \lceil R \Rightarrow N \rceil \wedge \lceil R \Rightarrow \neg W \rceil \wedge \Box\neg(\lceil W \rceil; \lceil \neg W \rceil),$$

which express that computation time is negligible, a process can never be both running and terminated, and, once terminated, is never re-activated. A dedicated pair of state variables R and W describes the status of each thread. N marks negligible time for all threads. The formulas

$$\mathsf{K}(V) \rightleftharpoons \bigwedge_{x \in V} x' = x \text{ and } \mathsf{K}^R(V) \rightleftharpoons \mathsf{K}(V) \wedge \lceil R \rceil.$$

mean that the variables from V preserve their values. $\mathsf{K}^R(V)$ additionally means that the thread is active throughout the reference interval. The clauses below define $[\![.]\!]_V$, where V is the set of program variables which are in the scope in the given code.

$$\begin{array}{lcl}
[\![\mathbf{stop}]\!]_V & \rightleftharpoons & \lceil W \rceil \\
[\![\mathbf{return}\ e]\!]_V & \rightleftharpoons & (\lceil \neg R \rceil; \mathsf{K}^R(V) \wedge \mathbf{r}' = e) \\
[\![\mathbf{return}]\!]_V & \rightleftharpoons & \lceil \neg R \rceil \\
[\![X]\!]_V & \rightleftharpoons & X \\
[\![(C_1; C_2)]\!]_V & \rightleftharpoons & ([\![C_1]\!]_V; [\![C_2]\!]_V) \\
[\![x := e]\!]_V & \rightleftharpoons & (\lceil \neg R \rceil; \mathsf{K}^R(V \setminus \{x\}) \wedge x' = e) \\
[\![\mathbf{delay}\ r]\!]_V & \rightleftharpoons & \lceil \neg R \rceil \wedge r = \int \neg N \\
[\![\mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2]\!]_V & \rightleftharpoons & (\lceil \neg R \rceil; (b \wedge \mathsf{K}^R(V); [\![C_1]\!]_V) \vee (\neg b \wedge \mathsf{K}^R(V); [\![C_2]\!]_V)) \\
[\![\mathbf{call}\ v := m(e_1, \ldots, e_n)]\!]_V & \rightleftharpoons & (\lceil \neg R \rceil; \mathsf{K}(V \setminus \{v\}) \wedge v' = m(e_1, \ldots, e_n)) \\
[\![\mathbf{call}\ m(e_1, \ldots, e_n)]\!]_V & \rightleftharpoons & (\lceil \neg R \rceil; \mathsf{K}(V) \wedge m(e_1, \ldots, e_n))
\end{array}$$

$$[\![\mathbf{letrec}\ C\ \mathbf{where}\ X_1 : C_1;\ \ldots\ X_n : C_n]\!]_V \rightleftharpoons \mu_{n+1} X_1 \ldots X_n Y. [\![C_1]\!]_V, \ldots, [\![C_n]\!]_V, [\![C]\!]_V$$

$$[\![\mathbf{var}\ v;\ C]\!]_V \rightleftharpoons \exists v \exists v' (\Box((\lceil \neg R \rceil \Rightarrow v' = v) \wedge \forall x \neg(v' = x; v \neq x)) \wedge \wedge [\![C]\!]_{V \cup \{v\}}$$

$$[\![(C_1 \| C_2)_V]\!] \rightleftharpoons \exists R_1 \exists R_2 \exists W_1 \exists W_2 \left(\begin{array}{l} \lceil W \Leftrightarrow W_1 \wedge W_2 \rceil \wedge \lceil R \Leftrightarrow R_1 \vee R_2 \rceil \wedge \lceil \neg R_1 \wedge R_2 \rceil \wedge \\ \mathsf{T}(R_1, W_1) \wedge [R_1/R, W_1/W][\![C_1]\!]_V \wedge \\ \mathsf{T}(R_2, W_2) \wedge [R_2/R, W_2/W][\![C_2]\!]_V \end{array} \right)$$

$$[\![\mathbf{export}\ m(p_1, \ldots, p_n)\ \mathit{code}]\!] \rightleftharpoons \Box \forall p_1 \ldots \forall p_n \forall \mathbf{r}' (\mathbf{r}' = m(p_1, \ldots, p_n) \Leftrightarrow [\![\mathit{code}]\!]_\emptyset), \text{ if } m \text{ returns a value}$$
$$[\![\mathbf{export}\ m(p_1, \ldots, p_n)\ \mathit{code}]\!] \rightleftharpoons \Box \forall p_1 \ldots \forall p_n (m(p_1, \ldots, p_n) \Leftrightarrow [\![\mathit{code}]\!]_\emptyset), \text{ if } m \text{ returns no value}$$

The semantics of a component is the conjunction of the formulas $[\![\mathbf{export}\ m(p_1, \ldots, p_n)\ \mathit{code}]\!]$ for its exported methods. Declarations of imported methods carry only typing information.

3 Reasoning about timed programs in PDC: pattern and examples

Let C be a piece of code. Then the formula $[\![C]\!]_V$ contains the flexible function and relation symbols for the methods with calls in C. Let m be such a method; let m return no value for the sake of simplicity. Let B be the body of m. By replacement of equivalents we can derive

$$[\![\mathbf{export}\ m(p_1,\ldots,p_n)\ B]\!] \wedge [\![C]\!]_V \Rightarrow [[\![B]\!]_\emptyset/m][\![C]\!]_V,$$

where the substitution $[[\![B]\!]_\emptyset/m]$ distributes over the boolean connectives, chop and quantifiers and $[\![B]\!]_\emptyset/m]m(e_1,\ldots,e_n)$ is defined as $[e_1/p_1,\ldots,e_n/p_n][\![B]\!]_\emptyset$. Assume that the satisfaction of a requirement Req_C written in DC by C is expressed as the equivalence

$$[\![C]\!]_V \Rightarrow Req_C$$

and, according to a contract, m is supposed to satisfy a requirement Req_m, that is,

$$[\![B]\!]_\emptyset \Rightarrow Req_m$$

is valid for every acceptable B. Then the formula

$$[Req_m/m][\![C]\!]_V \Rightarrow Req_C$$

states that C would satisfy Req_C, provided that the imported implementation of m satisfies Req_M.

This setting enables reasoning about the probability distribution of the execution time of code that calls imported methods too. Let C and m be as above. Then the probability for C to terminate within d time units can be expressed as the PDC term

$$p([\![C]\!]_V \wedge \textstyle\int \neg N \leq d; \top),$$

where we use $\int \neg N$ to measure only *non-negligible* execution time spent on the execution of **delay** or by other processes. Now let F_m be a rigid function symbol such that $F_m(x)$ denotes a lower bound for the probability for m to terminate within time x. Let P_m be the precondition for the successful execution of m. Let $\overline{p}$ abbreviate $p_1,\ldots,p_n$. Then

$$\forall x(\ell = 0 \Rightarrow p(P_m(\overline{p}) \wedge m(\overline{p}) \wedge \textstyle\int \neg N > x; \top) < 1 - F_m(x)) \vdash_{PITL} p([\![C]\!]_V \wedge \textstyle\int \neg N \leq d; \top) \geq c$$

means that the probability for C to terminate within d time units is at least c. The correspondence between the assumption on the execution time of m and the derived estimate of the execution time of C can be expressed even more accurately, if we make d and F_m parameters in an appropriate expression F_C in place of c:

$$\forall x(\ell = 0 \Rightarrow p(P_m(\overline{p}) \wedge m(\overline{p}) \wedge \textstyle\int \neg N > x; \top) < 1 - F_m(x)) \vdash_{PITL} p([\![C]\!]_V \wedge \textstyle\int \neg N > d; \top) < 1 - F_C(d, F_m).$$

In general F_C represents a mapping from distributions to distributions, but if the form of F_m is known up to numerical parameters such as *mean* and *variance*, then F_C can be defined as a mapping from their numerical domains instead of the space of distributions.

Example 1 Consider downloading e-mail, which consists of establishing a connection with a server, followed by the actual download. Let the code C for this call two imported methods, *connect*() and *getMail*():

$$\mathbf{var}\ ok;\ \mathbf{call}\ ok := connect();\ \mathbf{if}\ ok\ \mathbf{then}\ (\mathbf{call}\ getMail(); \mathbf{stop})\ \mathbf{else\ stop}$$

Let $F_{connect}(t)$ be the probability for connecting within time t. Let the amount of the e-mail be probabilistically distributed too and the probability for downloading it in time t be $F_{getMail}(t)$. Then lower bounds F_C for the distribution of the execution time of C satisfy the formula:

$$\ell = 0 \Rightarrow p([\![\mathbf{call}\ ok := connect()]\!] \wedge \textstyle\int \neg N > t; \top) < 1 - F_{connect}(t),$$
$$\ell = 0 \Rightarrow p(ok \wedge [\![\mathbf{call}\quad getMail()]\!] \wedge \textstyle\int \neg N > t; \top) < 1 - F_{getMail}(t)\ \vdash_{PITL} p([\![C]\!] \wedge \textstyle\int \neg N > t; \top) < 1 - F_C(t)$$

Since the time for connecting and the quantity of e-mail to download can be assumed independent,

$$F_C(t) = \int_0^y F_{connect}(y - t) dF_{getMail}(t). \tag{2}$$

F_C can be derived in *PITL* only approximately, because *PITL* does not capture taking the limits involved in the definition of the integral in (2). This corresponds to the established use of numerical approximations for distributions. Except for some thoroughly studied distributions, cummulative probability functions seldom have a closed form. Using contracts makes it natural to work with lower bounds and not exact probabilities. The latter may as well not exist. This makes approximations satisfactory. To derive such approximations for (2) in *PITL*, we find a sequence A_k, $k = 0, 1, \ldots$, of terms involving $F_{connect}$, $F_{getMail}$ and t such that

$$p([\![C]\!] \wedge \textstyle\int \neg N > t; \top) < 1 - A_k$$

for all k can be *derived in PITL* and, by the definition of $\int$, $\lim_k A_k = F_C(t)$. Taking this limit briefly takes us *outside PITL*. The part of the derivation within *PITL* is a formalisation of a standard calculation. Let

$$\varphi_{t_1}^{t_2} \rightleftharpoons \varphi \wedge \textstyle\int \neg N > t_1 \wedge \int \neg N \leq t_2 \tag{3}$$

Every method call can terminate at most once. This implies the validity of the formulas $connect \Rightarrow \neg(connect; \ell \neq 0)$ and $getMail \Rightarrow \neg(getMail; \ell \neq 0)$ and enables an application of *Seq* to derive

$$\ell = 0 \wedge p(connect; getMail; \top) = 1 \Rightarrow$$
$$p(connect_{\frac{lt}{k}}^{\frac{(l+1)t}{k}}; getMail_{\frac{mt}{k}}^{\frac{(m+1)t}{k}}; \top) = (F_{connect}(\tfrac{(l+1)t}{k}) - F_{connect}(\tfrac{lt}{k}))(F_{getMail}(\tfrac{(m+1)t}{k}) - F_{getMail}(\tfrac{mt}{k}))$$

for all $l, m \in \{0, \ldots, k-1\}$. Now by a repeated application of the *PITL* axiom P_+, and using that $F_{connect}(0) = 0$, we obtain

$$\begin{aligned}
p((connect; getMail) \wedge \textstyle\int \neg N \leq t; \top) \quad &\leq \quad \sum_{l+m \leq k-1} p(connect_{\frac{lt}{k}}^{\frac{(l+1)t}{k}}; getMail_{\frac{mt}{k}}^{\frac{(m+1)t}{k}}; \top) + \\
&\qquad \sum_{l+m=k} p(connect_{\frac{lt}{k}}^{\frac{(l+1)t}{k}}; getMail_{\frac{mt}{k}}^{\frac{(m+1)t}{k}}; \top) \\
&= \quad \underbrace{\sum_{m \leq k-1} F_{connect}(\tfrac{(k-m+1)t}{k})(F_{getMail}(\tfrac{(m+1)t}{k}) - F_{getMail}(\tfrac{mt}{k}))}_{S_k} + B_k
\end{aligned}$$

where $B_k \leq \max_{l \leq k-1} F_{connect}(\frac{(l+1)t}{k}) - F_{connect}(\frac{lt}{k})$, and therefore $\lim_k B_k = 0$. By the definition of Stieltjes integral, we have $\lim_k S_k = F_C(t)$. Hence we can take A_k to be the expression on the right of $\leq$ above.

Note that *Seq* was formulated with $\varphi_{t_1}^{t_2}$ standing for $\varphi \wedge l \geq t_1 \wedge \ell \leq t_2$, but it applies to (3) as well.

Example 2 Consider attempting to download 5 files in quick succession. With a server which allows at most 4 files to be downloading simultaneously, the 5th request can be cancelled by the browser due to a timeout. We are interested in the probability of cancellation. Here follows an extremely simplified variant of the relevant browser code:

$$\begin{array}{l}
\textbf{letrec } X \textbf{ where} \\
\quad X : \textbf{if } \mathit{userRequest} \textbf{ then} \\
\qquad (\mathit{userRequest} := \textbf{false}; \\
\qquad X \| \left(\begin{array}{l}
\textbf{call } \mathit{handle} := \mathit{requestDownload}(\mathit{url}, \mathit{timeout}); \\
\textbf{if } \mathit{handle}! = \textbf{null} \\
\quad \textbf{then } (\textbf{call } \mathit{download}(\mathit{handle}); \textbf{stop}) \\
\quad \textbf{else } (\textbf{call } \mathit{signalTimeout}(\mathit{url}); \textbf{stop})
\end{array} \right) \\
\qquad) \\
\qquad \textbf{else } X
\end{array} \tag{4}$$

A separate process is assumed to indicate the arrival of a new download request by setting the shared variable *userRequest* and placing the URL in the shared variable *url*. Let

$$\alpha(R, T) \rightleftharpoons \left(\begin{array}{l} (\lceil \neg R \rceil; \mathsf{K}^R(V) \wedge \neg \mathit{userRequest})^*; \lceil \neg R \rceil; \mathsf{K}^R(V) \wedge \mathit{userRequest}; \\ \lceil \neg R \rceil; \mathsf{K}^R(V \setminus \{\mathit{userRequest}\}) \wedge \mathit{userRequest}' = \textbf{false} \end{array} \right) \wedge \textstyle\int \neg N = T$$

and

$$\beta(R, W, T) \rightleftharpoons \left(\begin{array}{l} \lceil \neg R \rceil; \mathsf{K}^R(V \setminus \{\mathit{handle}\}) \wedge \mathit{handle}' = \mathit{requestDownload}(\mathit{url}, \mathit{timeout}); \\ \lceil \neg R \rceil; \mathsf{K}^R(V) \wedge \mathit{handle}! = \textbf{null}; \lceil \neg R \rceil; \mathsf{K}^R(V) \wedge \mathit{download}(\mathit{handle}) \wedge \int \neg N = T; \lceil W \rceil \end{array} \right)$$

According to the semantics of (4), $\alpha(R, T)$ describes the repeated execution of lines 2-3 and 9 until *userRequest* becomes true with T denoting the overall execution time, and $\beta(R, W, T)$ corresponds to the execution of lines 4-6, with R and W describing the status of the thread created in order to complete the requested download, and T denoting the download time. The scenario of launching the five downloads involves six threads: one for each download and one to keep the system ready for further requests. Let $R_1, \ldots, R_6, W_1, \ldots, W_6$ describe the status of the six threads. Then the scenario can be described by the formula

$$\exists R_1 \ldots \exists R_6 \exists W_1 \ldots \exists W_6 \left(\lceil W \Leftrightarrow \bigwedge_{i=1}^{6} W_i \rceil \wedge \lceil R \Leftrightarrow \bigvee_{i=1}^{6} R_i \rceil \wedge \lceil \bigwedge_{1 \leq i < j \leq 6} \neg(R_i \wedge R_j) \rceil \wedge \bigwedge_{i=1}^{6} \mathsf{T}(R_i, W_i) \wedge \gamma \right) \quad (5)$$

where γ describes the concurrent execution of the six threads and is written using the additional abbreviations:

$$\alpha_i(T) \rightleftharpoons \alpha(R_{i+1} \vee \ldots \vee R_6, W_{i+1} \wedge \ldots \wedge W_6, T) \text{ and } \beta_i(T) \rightleftharpoons \beta(R_i, W_i, T).$$

With these abbreviations γ can be written as

$$\left(\alpha(R, W, T_0); \left(\begin{array}{l} \beta_1(D_1) \wedge \\ \left(\alpha_1(T_1); \left(\begin{array}{l} \beta_2(D_2) \wedge \\ \left(\alpha_2(T_2); \left(\begin{array}{l} \beta_3(D_3) \wedge \\ \left(\alpha_3(T_3); \left(\begin{array}{l} \beta_4(D_4) \wedge \\ (\alpha_4(T_4); \xi \wedge \eta) \end{array} \right) \right) \end{array} \right) \right) \end{array} \right) \right) \end{array} \right) \right)$$

Here T_i denotes the time between launching the ith and the $i+1$st download and D_i denotes the duration of the ith download, $i = 1, \ldots, 5$. The formulas ξ and η denote

$$(\lceil \neg R_5 \rceil; \mathsf{K}^{R_5}(V \setminus \{handle\}) \wedge handle' = requestDownload(url, timeout) \wedge \textstyle\int \neg N = x; \top))$$

and

$$((\lceil \neg R_6 \rceil; \mathsf{K}^{R_6}(V) \wedge \neg userRequest)^*; \top),$$

and correspond to the thread for the 5th download and the thread for subsequent user requests after the 5th download request. The occurrences of $\top$ in them mark future behaviour which is not specified in our scenario. The semantics of **letrec** implies (5); this can be established using the validity of $\mu X.\varphi \Leftrightarrow [\mu X.\varphi/X]\varphi$. Assuming that the rate of downloading is the limiting factor for the working of the entire system, which allows us to ignore time taken for dialog, computation and by *requestDownload* for the first four downloads, the 5th download becomes cancelled in case x exceeds *timeout*, which is equivalent to

$$timeout + \sum_{j=1}^{4} T_i < \min_{i=1}^{4} \left(D_i + \sum_{j=1}^{i-1} T_j \right).$$

Let $F(l, t)$ be a lower bound for the probability for *download* do complete a download of length l within time t. It can be assumed that $F(al, at) = F(l, t)$ for all $a > 0$ and that $F(l, t) = 0$ in case $\frac{l}{t}$ exceeds the top transmission rate v. Let l_i be the length of the ith download, $i = 1, \ldots, 5$. Let $l_i > v(T_1 + T_2 + T_3 + T_4)$ for $i = 1, \ldots, 4$, that is, none of the downloads can be over before all of them have been launched, for the sake of simplicity. Then the probability P_i for $i \in \{1, \ldots, 4\}$ to be the first download to complete, and to complete before the timeout for the pending 5th download is at least

$$\int\limits_{\{\langle q_1, \ldots, q_4 \rangle : l_i - q_i \leq l_j - q_j, j = 1, \ldots, 4\}} \tfrac{\partial}{\partial q} F(q, t)(l_i - q_i, timeout). \prod_{k=1}^{4} F'(q_k, \sum_{s=k}^{4} T_s) dq_1 \ldots dq_4,$$

The probability for the 5th download not to be cancelled is $P_1 + \ldots + P_4$. Approximations of the above integral can be derived in *PITL* using *Seq* much like in Example 1.

Using a contract in which the execution time of *download* is approximated by a distribution depending just the amount of data to transmit is too crude. A more accurate calculation is possible by taking the amount of competing traffic in account, but the form of contract that we propose does not enable it.

4 Probabilistic timed designs

A design $\langle P, R\rangle$, usually written as $P \vdash R$, describes a computation by a *precondition* P, an *input-output relation* R. P constrains the initial values v of the variables, and R is a relation between v and the final values v' of the variables, which holds if v initially satisfy P. A *probabilistic timed design* $\langle P, R, F\rangle$ additionally includes an *execution time distribution* F. $F(v,t)$ is a lower bound for the probability for the computation to terminate within time t, provided that $P(v)$ holds. A *hard* bound d on execution time can be expressed by a F satisfying $F(d') = 1$ for all $d' \geq d$.

4.1 Describing designs in *PDC*

The property of method m encoded by $\langle P, R, F\rangle$ can be written as the *PITL* formulas

$$m \Rightarrow (P(v) \Rightarrow R(v,v')) \text{ and } \ell = 0 \Rightarrow p(P(v) \wedge m \wedge \ell > t; \top) < 1 - F(v,t).$$

The first one is for the *functional* behaviour of m. The second one states that if $P(v)$ holds, then m to takes more than t time units with probability less than $1 - F(v,t)$. F is just a lower bound, because an *exact* probability need not exist.

4.2 Refinement of probabilistic timed designs

Design $D_1 = \langle P_1, R_1, F_1\rangle$ *refines* design $D_2 = \langle P_2, R_2, F_2\rangle$, written $D_1 \sqsubseteq D_2$, if

$$\forall x(P_2(x) \Rightarrow P_1(x)),\ \forall x\forall x'(R_1(x,x') \Rightarrow R_2(x,x')),\ \text{and } \forall x\forall t(F_2(x,t) \leq F_1(x,t)).$$

This means that D_1 has a *weaker or equivalent* precondition and a *stronger or equivalent* input-output relation, and on average terminates *at least as fast* as D_2. Obviously if $D_1 \sqsubseteq D_2$, then

$$P_1(v) \wedge m \Rightarrow R_1(v,v') \text{ and } \forall x(\ell = 0 \Rightarrow p(P_1(v) \wedge m \wedge \ell > x; \top) < 1 - F_1(v,x))$$

entail

$$P_2(v) \wedge m \Rightarrow R_2(v,v') \text{ and } \forall x(\ell = 0 \Rightarrow p(P_2(v) \wedge m \wedge \ell > x; \top) < 1 - F_2(v,x)).$$

5 Probabilistic timed contracts

The execution time of a method depends on the execution times of the methods which have calls in its body.

Definition 3 (component declaration) A *component declaration* is a pair $\langle M_i, M_e\rangle$ where M_i and M_e are disjoint sets of declarations for *imported* and *exported* methods, respectively.

Definition 4 (probabilistic timed contract) Let $\langle M_i, M_e\rangle$ be a component declaration and V_m be the set of the valuations for the variables of declaration m, $m \in M_i \cup M_e$. The tuple $C = \langle D_m : m \in M_i \cup M_e\rangle$ is a *contract* for $\langle M_i, M_e\rangle$, if D_m are of the form $\langle P_m, R_m, F_m\rangle$ where

(i) $\langle P_m, R_m\rangle$ is a (non-probabilistic) design for m, $m \in M_i \cup M_e$.

(ii) F_m is a variable of type $V_m \times \mathbf{R}_+ \to [0,1]$ for method declarations $m \in M_i$ and is meant to denote a distribution of the execution time of implementations of m.

(iii) For declarations $m \in M_e$, F_m an expression for the distibution of the execution time of an implementation of declaration m as in probabilistic designs in terms of F_n, $n \in M_i$.

We denote $\{n \in M_i : F_n \text{ occurs in } F_m\}$ by $C_{i,m}$. Semantically, if $m \in M_e$, then the type of F_m is

$$\left(\prod_{n \in C_{i,m}} V_n \times \mathbf{R}_+ \to [0,1]\right) \to (V_m \times \mathbf{R}_+ \to [0,1]).$$

Syntactically we assume that F_m is an expression such as, e.g., (2). A contract C is meant to express that if the methods $m \in M_i$ satisfy their corresponding designs D_m and the distribution variables F_m are assigned lower bounds for the distributions of their execution times, then the methods from M_e satisfy their corresponding designs and the expressions F_m evaluate to lower bounds for the distributions of their execution times too. If $C_{i,m} = 0$ then $\langle P_m, R_m, F_m\rangle$ is essentially a probabilistic timed design.

Definition 5 (refinement of probabilistic timed contracts) Let C and C' be probabilistic timed contracts for $\langle M_i, M_e\rangle$ and $\langle M_i', M_e'\rangle$, respectively. Let $C = \langle\langle P_m, R_m, F_m\rangle : m \in M_i \cup M_e\rangle$ and $C' = \langle\langle P_m', R_m', F_m'\rangle : m \in M_i' \cup M_e'\rangle$. Then, C' refines C, written $C' \sqsubseteq C$, if

(i) $M_i' \subseteq M_i$, $M_e' \supseteq M_e$;

(ii) $\langle P_m, R_m\rangle \sqsubseteq \langle P_m', R_m'\rangle$ for $m \in M_i'$, $\langle P_m', R_m'\rangle \sqsubseteq \langle P_m, R_m\rangle$ for $m \in M_e$;

(iii) $F_m(v,t) \leq F_m'(v,t)$ for $m \in M_e, v \in V_m, t \in \mathbf{R}_+$ regardless of the values of F_n, $n \in M_i$.

5.1 Composing probabilistic timed contracts

Let $A^k = \langle M_i^k, M_e^k\rangle$ and $C^k = \langle\langle P_m^k, R_m^k, F_m^k\rangle : m \in M_i^k \cup M_e^k\rangle$, $k = 1,2$, be two component declarations and probabilistic timed contracts for them, respectively. A^1 and A^2 are composable, if $M_e^1 \cap M_e^2 = \emptyset$. C^1 and C^2 are composable, if A^1 and A^2 are composable, and $D_m^k \sqsubseteq D_m^{2-k}$ for $m \in M_e^k \cap M_i^{2-k}$, $k = 1,2$. The composition of C^1 and C^2, written $C^1 \cup C^2$, is $\langle\langle P_m, R_m, F_m\rangle : m \in M_i^1 \cup M_e^1 \cup M_i^2 \cup M_e^2\rangle$ where:

(i) $P_m(v) \rightleftharpoons P_m^1(v) \wedge P_m^2(v)$, $R_m(v,v') \rightleftharpoons R_m^1(v,v') \wedge R_m^2(v,v')$ and $F_m = F_m^1 = F_m^2$ for $m \in M_i^1 \cap M_i^2$;

(ii) $P_m = P_m^k$ and $R_m = R_m^k$ for $m \in M_e^k \cup (M_i^k \setminus M_i^{2-k})$, $k = 1,2$;

(iii) $F_m = F_m^k$ for $m \in M_i^k \setminus M_i^{2-k}$, $k = 1,2$.

To facilitate the understanding, we first define F_m, $m \in M_e^1 \cup M_e^2$, in case C^1 and C^2 allow no *circular dependency* between the methods, that is, if there is no sequence $m_0, \ldots, m_{2s-1}$ such that $m_r \in M_e^1 \cap M_i^2$ for $r = 1,3,\ldots,2s-1$, $M_r \in M_e^2 \cap M_i^1$ for $r = 0,2,\ldots,2s-2$ and $m_r \in C^{2-r \bmod 2}_{i,m_{r+1 \bmod 2s}}$, $r = 0,\ldots,2s-1$. Given that there is no circular dependency, we can define *dependency depth* of m from $C^1 \cup C^2$ as the length s of the longest sequence of the form $m_1, \ldots, m_s$ such that $m_1 \in C_{i,m}$ and $m_{r+1} \in C^k_{i,m_r}$, where k is such that $m_r \in M_e^k$, for $r = 1,\ldots,s-1$, and we can define F_m by induction on the dependency depth of m by the clauses:

$F_m = F_m^k$ for $m \in M_e^k$ of dependency depth 0;

$F_m = [F_n/F_n^k : n \in C^k_{i,m}]F_m^k$ for $m \in M_e^k$ of nonzero dependency depth.

Note that the substitution replaces F_n^k with the expression for it from C^{2-k}, in case $n \in M_e^{2-k}$. Otherwise F_n^k is not affected by this substitution.

If thereare circular dependencies between C^1 and C^2, then the F_ms for the exported methods in $C^1 \cup C^2$ should be a solution of the system of equations

$$X_m = [X_n/F_n^k : n \in C_{i,m}]F_m^k.$$

Solving it without restrictions on F_m^k can be hard[2], but if F_m^k and monotonic, then X_m can be obtained as the limits of the sequences X_m^j, $j < \omega$, where $X_m^0 \equiv 0$ and

$$X_m^{j+1} = [X_n^j/F_n^k : n \in C^k_{i,m}]F_m^k \text{ for } m \in M_e^k.$$

Observe that $X_m^0 \equiv 0$ implies that X_m^1 would give non-zero termination probability only to runs of m with no calls to other imported methods; X_m^2 would give non-zero probability for runs with calls to imported methods which themselves lead to no further calls, etc. Since F_m^k are meant to be *under*-approximations, and the monotonicity of F_m^k entails $X_m^s \leq X_m^{s+1} \leq \lim_j X_m^j$ for all $s < \omega$, X_m^s can be used as F_m instead of $\lim_j X_m^j$ for sufficiently large s, to achieve a crude, but less expensive approximation.

Concluding remarks

Here we focused just on soft requirements on execution time, but we believe that the approach can be used to capture other QoS requirements involving probability as well. The notion of QoS originated from telecommunications. Our examples come from everyday use of the Internet and need no expertise to understand. However, we believe that our technique would work just as well in other areas such as embedded systems.

[2]In practice F_m^k *can* be non-monotonic: increasing the execution time of an imported method may indeed shorten the execution time of code which would abort if an imported method misses a deadline.

References

[BJPW99] Antoine Beugnard, Jean-Marc Jézéquel, Noël Plouzeau, and Damien Watkins. Making Components Contract Aware. *Computer*, 32(7):38–45, 1999.

[Dan98] Dang Van Hung. Modelling and Verification of Biphase Mark Protocols in Duration Calculus Using PVS/DC$^-$. In *Proceedings of the 1998 International Conference on Application of Concurrency to System Design (CSD'98)*, pages 88–98. IEEE Computer Society Press, March 1998.

[Dan05] Dang Van Hung. Toward a formal model for component interfaces for real-time systems. In Tiziana Margaria and Mieke Massink, editors, *Proceedings of the 10th international workshop on Formal methods for industrial critical systems*, pages 106 – 114. ACM Press, 2005.

[DW96] Dang Van Hung and Wang Ji. On The Design of Hybrid Control Systems Using Automata Models. In *Proceedings of FST TCS 1996*, volume 1180 of *LNCS*, pages 156–167. Springer, 1996.

[DZ99] Dang Van Hung and Zhou Chaochen. Probabilistic Duration Calculus for Continuous Time. *Formal Aspects of Computing*, 11(1):21–44, 1999.

[GD02] Dimitar P. Guelev and Dang Van Hung. Prefix and Projection onto State in Duration Calculus. In *Proceedings of TPTS'02*, volume 65(6) of *ENTCS*. Elsevier Science, 2002.

[GD05] Dimitar P. Guelev and Dang Van Hung. On the Completeness and Decidability of Duration Calculus with Iteration. *Theoretical Computer Science*, 337:278–304, 2005.

[Gue00a] Dimitar P. Guelev. A Complete Fragment of Higher-order Duration μ-calculus. In *Proceedings of FST TCS 2000*, volume 1974 of *LNCS*, pages 264–276. Springer, 2000.

[Gue00b] Dimitar P. Guelev. Probabilistic Neighbourhood Logic. In Mathai Joseph, editor, *Proceedings of FTRTFT 2000*, volume 1926 of *LNCS*, pages 264–275. Springer, 2000. A proof-complete version is available as UNU/IIST Technical Report 196 from `http://www.iist.unu.edu`.

[Gue07] Dimitar P. Guelev. Probabilistic Interval Temporal Logic and Duration Calculus with Infinite Intervals: Complete Proof Systems. *Logical Methods in Computer Science*, 3(3), 2007. URL: `http://www.lmcs-online.org/`.

[HD07] Hung Ledang and Dang Van Hung. Concurrency and Schedulability Analysis in Component-based Real-Time System Development. In *Proceedings of the 1st IEEE & IFIP International Symposium on Theoretical Aspects of Software Engineering*. IEEE Computer Society Press, 2007.

[HH98] C.A.R. Hoare and He Jifeng. *Unifying Theories of Programming*. Prentice Hall, 1998.

[HLL06] He Jifeng, Li Xiaoshan, and Liu Zhiming. A Theory of Reactive Components. In Liu Zhiming and Luis Barbosa, editors, *Proceedings of the International Workshop on Formal Aspects of Component Software (FACS 2005)*, volume 160 of *ENTCS*, pages 173–195. Elsevier, 2006.

[HXZ06] He Jifeng, Xiaoshan Li, and Zhiming Liu. A refinement calculus of object systems. *Theoretical Computer Science*, 365(1-2):109–142, 2006.

[HZ92] Michael R. Hansen and Zhou Chaochen. Semantics and Completeness of Duration Calculus. In *Real-Time: Theory and Practice*, volume 600 of *LNCS*, pages 209–225. Springer, 1992.

[LH99] Li Li and He Jifeng. A Denotational Semantics of Timed RSL using Duration Calculus. In *Proceedings of RTCSA'99*, pages 492–503. IEEE Computer Society Press, 1999.

[LRSZ93] Liu Zhiming, A. P. Ravn, E. V. Sørensen, and Zhou Chaochen. A Probabilistic Duration Calculus. In H. Kopetz and Y. Kakuda, editors, *Dependable Computing and Fault-tolerant Systems Vol. 7: Responsive Computer Systems*, pages 30–52. Springer, 1993.

[Pan95] Paritosh K. Pandya. Some extensions to Mean-Value Calculus: Expressiveness and Decidability. In *Proceedings of CSL'95*, volume 1092 of *LNCS*, pages 434–451. Springer, 1995.

[PD98] Paritosh K. Pandya and Dang Van Hung. Duration Calculus of Weakly Monotonic Time. In *Proceedings of FTRTFT'98*, volume 1486 of *LNCS*, pages 55–64. Springer, 1998.

[PWX98] Paritosh K. Pandya, Wang Hanping, and Xu Qiwen. Towards a Theory of Sequential Hybrid Programs. In D. Gries and W.-P. de Roever, editors, *Proceedings of IFIP Working Conference PROCOMET'98*, pages 336–384. Chapman & Hall, 1998.

[SX98] Gerardo Schneider and Xu Qiwen. Towards a Formal Semantics of Verilog Using Duration Calculus. In Anders P. Ravn and Hans Rischel, editors, *Proceedings of FTRTFT'98*, volume 1486 of *LNCS*, pages 282–293. Springer, 1998.

[WX04] Wang Hanpin and Xu Qiwen. Completeness of Temporal Logics over Infinite Intervals. *Discrete Applied Mathematics*, 136(1):87–103, 2004.

[ZDL95] Zhou Chaochen, Dang Van Hung, and Li Xiaoshan. A Duration Calculus with Infinite Intervals. In Horst Reichel, editor, *Fundamentals of Computation Theory*, volume 965 of *LNCS*, pages 16–41. Springer, 1995.

[ZGZ00] Zhou Chaochen, Dimitar P. Guelev, and Zhan Naijun. A Higher-order Duration Calculus. In *Millennial Perspectives in Computer Science*, pages 407–416. Palgrave, 2000.

[ZZ94] Zheng Yuhua and Zhou Chaochen. A Formal Proof of a Deadline Driven Scheduler. In *Proceedings of FTRTFT'94*, volume 863 of *LNCS*, pages 756–775. Springer, 1994.

A Proof systems

A.1 Proof system for *ITL* with infinite intervals

The following axioms and rules have been shown to form a complete proof system for *ITL* with infinite intervals when added to a Hilbert-style proof system for classical first-order predicate logic and appropriate axioms about an abstract domain of durations in [WX04]:

$(A1)$ $(\varphi;\psi) \wedge \neg(\chi;\psi) \Rightarrow (\varphi \wedge \neg\chi;\psi)$, $(\varphi;\psi) \wedge \neg(\varphi;\chi) \Rightarrow (\varphi;\psi \wedge \neg\chi)$

$(A2)$ $((\varphi;\psi);\chi) \Leftrightarrow (\varphi;(\psi;\chi))$

(R) $(\varphi;\psi) \Rightarrow \varphi$, $(\psi;\varphi) \Rightarrow \varphi$ if φ is rigid

(B) $(\exists x\varphi;\psi) \Rightarrow \exists x(\varphi;\psi)$, $(\psi;\exists x\varphi) \Rightarrow \exists x(\psi;\varphi)$ if x has no free occurrences in ψ

$(L1)$ $(\ell = x;\varphi) \Rightarrow \neg(\ell = x;\neg\varphi)$, $(\varphi;\ell = x \wedge x \neq \infty) \Rightarrow \neg(\neg\varphi;\ell = x)$

$(L2)$ $\ell = x + y \wedge x \neq \infty \Leftrightarrow (\ell = x;\ell = y)$

$(L3)$ $\varphi \Rightarrow (\ell = 0;\varphi)$, $\varphi \wedge \ell \neq \infty \Rightarrow (\varphi;\ell = 0)$

$(S1)$ $(\ell = x \wedge \varphi;\psi) \Rightarrow \neg(\ell = x \wedge \neg\varphi;\chi)$

$(P1)$ $\neg(\ell = \infty;\varphi)$

$(P2)$ $(\varphi;\ell = \infty) \Rightarrow \ell = \infty$

$(P3)$ $(\varphi;\ell \neq \infty) \Rightarrow \ell \neq \infty$

(N) $\dfrac{\varphi}{\neg(\neg\varphi;\psi)}, \dfrac{\varphi}{\neg(\psi;\neg\varphi)}$

$(Mono)$ $\dfrac{\varphi \Rightarrow \psi}{(\varphi;\chi) \Rightarrow (\psi;\chi)}, \dfrac{\varphi \Rightarrow \psi}{(\chi;\varphi) \Rightarrow (\chi;\psi)}$

Using the first order logic axiom

$(\exists_r)$ $[t/x]\varphi \Rightarrow \exists x\varphi.$

is correct only if no variable in t becomes bound due to the substitution, and either t is rigid or $(.;.)$ does not occur in φ.

A.2 Axioms and rules for *DC* with infinite intervals

The axioms and rules below were proposed for *DC* with *finite* intervals and have been shown to be complete relative to validity in real-time *ITL* in [HZ92].

$(DC1)$ $\int \mathbf{0} = 0$

$(DC2)$ $\int \mathbf{1} = \ell$

$(DC3)$ $\int S \geq 0$

$(DC4)$ $\int S_1 + \int S_2 = \int(S_1 \vee S_2) + \int(S_1 \wedge S_2)$

$(DC5)$ $(\int S = x;\int S = y) \Rightarrow \int S = x + y$

$(DC6)$ $\int S_1 = \int S_2$ if S_1 and S_2 are propositionally equivalent

$(IR1)$ $\dfrac{[\ell = 0/A]\varphi \quad \varphi \Rightarrow [A \vee (A;\llbracket S \rrbracket \vee \llbracket \neg S \rrbracket)/A]\varphi}{[\top/A]\varphi}$

$(IR2)$ $\dfrac{[\ell = 0/A]\varphi \quad \varphi \Rightarrow [A \vee (\llbracket S \rrbracket \vee \llbracket \neg S \rrbracket;A)/A]\varphi}{[\top/A]\varphi}$

The completeness proof from [HZ92] involves two theorems which can be derived using the rules $IR1$ and $IR2$, instead of the rules themselves. The second of these theorems does not hold for infinite intervals and therefore we modify it appropriately:

$(T1)$ $\quad \ell = 0 \vee (\lceil S \rceil; \top) \vee (\lceil \neg S \rceil; \top)$

$(T2)$ $\quad \ell = 0 \vee \ell = \infty \vee (\top; \lceil S \rceil) \vee (\top; \lceil \neg S \rceil)$

$DC1$-$DC6$, $T1$ and the infinite-interval variant of $T2$ form a relatively complete proof system for DC with infinite intervals.

A.3 Proof system for $PITL$

$PITL$ is a conservative extension of ITL. Adding the axioms and a proof rule below to the proof system for ITL leads to a system which is complete for $PITL$ with respect to a generalisation of the **R**-based semantics, where $\overline{\mathbf{R}}$ is replaced by an abstract domain and the probability measures are required to be only finitely aditive.

Extensionality

$(P_;)$ $\quad (\ell = x; p(\psi) = y) \Rightarrow p((\ell = x; \psi)) = y$

(P_∞) $\quad \ell = \infty \Rightarrow (\varphi \Leftrightarrow p(\varphi) = 1)$

$(P_\leq)$ $\quad \dfrac{\vdash (\varphi; \ell = \infty) \Rightarrow (\psi \Rightarrow \chi)}{\vdash \varphi \wedge \ell < \infty \Rightarrow p(\psi) \leq p(\chi)}$

Arithmetics of probabilities

$(P_\perp)$ $\quad p(\perp) = 0$

$(P_\top)$ $\quad p(\top) = 1$

(P_+) $\quad p(\varphi) + p(\psi) = p(\varphi \vee \psi) + p(\varphi \wedge \psi)$

A.4 Useful theorems and derived rules for $PITL$

All the theorems and rules below except $P'_;$ are valid in general $PITL$ models. $P'_;$ is valid in $PITL$ models with global probability.

$(P^\infty_\leq)$ $\quad \dfrac{(\varphi; \ell = \infty) \vee (\varphi \wedge \ell = \infty) \Rightarrow (\psi \Rightarrow \chi)}{\varphi \Rightarrow p(\psi) \leq p(\chi)}$

$(PITL1)$ $\quad \dfrac{\varphi \Rightarrow \psi}{p(\varphi) \leq p(\psi)}, \dfrac{\varphi \Leftrightarrow \psi}{p(\varphi) = p(\psi)}$

$(PITL2)$ $\quad p(\varphi) + p(\neg\varphi) = 1$

$(PITL3)$ $\quad p(\varphi) < p(\psi) \Rightarrow p(\psi \wedge \neg\varphi) \neq 0$

$(PITL4)$ $\quad p(\varphi) = p(\varphi \wedge \ell = \infty)$

$(PITL5)$ $\quad p(\varphi) \leq 1$

$(PITL6)$ $\quad \dfrac{\varphi}{p(\varphi) = 1}, \dfrac{\neg\varphi}{p(\varphi) = 0}$

$(PITL7)$ $\quad (\varphi; \top) \Rightarrow p(\varphi; \top) = 1$

$(PITL8)$ $\quad p(\varphi) = 1 \wedge p(\psi) = x \Rightarrow p(\varphi \wedge \psi) = x$

$(PITL9)$ $\quad p(\varphi \Rightarrow \psi) = 1 \Rightarrow (p(\varphi) = 1 \Rightarrow p(\psi) = 1)$

$\quad p(\varphi \Rightarrow \psi) = 1 \Rightarrow (p(\psi) = 0 \Rightarrow p(\varphi) = 0)$

$(PITL10)$ $\quad p(\varphi) + p(\psi) > 1 \Rightarrow p(\varphi \wedge \psi) > 0$

$(P'_;)$ $\quad \dfrac{\varphi \Rightarrow \neg(\varphi; \ell \neq 0)}{(\varphi; p(\psi) = x) \Rightarrow p(\varphi; \psi) = x}$

Here follow the proofs of the above $PITL$ theorems and derived rules. The purely ITL parts are skipped and marked “ITL” for the sake of brevity.

$P_\leq^\infty$:

$$\begin{array}{lll}
1 & (\varphi;\ell=\infty)\Rightarrow(\psi\Rightarrow\chi) & \text{assumption}, ITL\\
2 & \varphi\wedge\ell<\infty\Rightarrow p(\psi)\leq p(\chi) & 1, P_\leq\\
3 & \ell=\infty\wedge\varphi\Rightarrow(p(\psi)=0\wedge p(\chi)=0) & \text{assumption}, P_\infty, PITL2\\
 & \qquad\qquad\qquad\vee(p(\psi)=0\wedge p(\chi)=1) & \\
 & \qquad\qquad\qquad\vee(p(\psi)=1\wedge p(\chi)=1) & \\
4 & \varphi\wedge\ell=\infty\Rightarrow p(\psi)\leq p(\chi) & 3, ITL\\
5 & \ell<\infty\vee\ell=\infty & ITL\\
6 & \varphi\Rightarrow p(\psi)\leq p(\chi) & 2,4,5
\end{array}$$

$PITL1$:

$$\begin{array}{lll}
1 & \varphi\Rightarrow\psi & \text{assumption}\\
2 & (\top;\ell=\infty)\vee(\top\wedge\ell=\infty)\Rightarrow(\varphi\Rightarrow\psi) & 1, ITL\\
3 & p(\varphi)\leq p(\psi) & 2, P_\leq^\infty
\end{array}$$

The second rule $PITL1$ is proved by two applications the first.
$PITL2$:

$$\begin{array}{lll}
1 & \varphi\wedge\neg\varphi\Leftrightarrow\bot & ITL\\
2 & p(\varphi\wedge\neg\varphi)=p(\bot) & 1, PITL1\\
3 & p(\varphi\wedge\neg\varphi)=0 & 2, P_\bot\\
4 & \varphi\vee\neg\varphi\Leftrightarrow\top & ITL\\
5 & p(\varphi\vee\neg\varphi)=p(\top) & 4, PITL1\\
6 & p(\varphi\wedge\neg\varphi)=1 & 5, P_\top\\
7 & p(\varphi)+p(\neg\varphi)=p(\varphi\wedge\neg\varphi)+p(\varphi\wedge\neg\varphi) & P_+\\
8 & p(\varphi)+p(\neg\varphi)=1 & 2,6,7, ITL
\end{array}$$

$PITL3$:

$$\begin{array}{lll}
1 & p(\psi)\leq p(\varphi\vee\psi) & P_\leq^\infty\\
2 & p(\varphi)+p(\psi\wedge\neg\varphi)=p(\varphi\wedge\psi\wedge\neg\varphi)+p(\varphi\vee\psi\wedge\neg\varphi) & P_+\\
3 & p(\varphi)+p(\psi\wedge\neg\varphi)=p(\varphi\vee\psi) & 2, PITL1, P_\bot\\
4 & p(\varphi)<p(\psi)\Rightarrow p(\varphi)<p(\varphi\vee\psi) & 1\\
5 & p(\varphi)<p(\psi)\Rightarrow p(\psi\wedge\neg\varphi)\neq 0 & 3,4
\end{array}$$

$PITL4$ is obtained by applying $P_\leq^\infty$ to the ITL theorems

$$(\top;\ell=\infty)\vee(\top\wedge\ell=\infty)\Rightarrow(\varphi\Rightarrow\varphi\wedge\ell=\infty)\text{ and }(\top;\ell=\infty)\vee(\top\wedge\ell=\infty)\Rightarrow(\ell=\infty\wedge\varphi\Rightarrow\varphi).$$

$PITL5$:

$$\begin{array}{lll}
1 & \varphi\Rightarrow\top & ITL\\
2 & p(\varphi)\leq p(\top) & 1, PITL1\\
3 & p(\varphi)\leq 1 & 2, P_\top
\end{array}$$

$PITL6$:

$$\begin{array}{lll}
1 & \top\Rightarrow\varphi & \text{assumption}\\
2 & p(\top)\leq p(\varphi) & 1, PITL1\\
3 & 1\leq p(\varphi) & 2, P_\top\\
4 & p(\varphi)\leq 1 & PITL5\\
5 & p(\varphi)=1 & 3,4
\end{array}$$

$$\begin{array}{lll}
1 & \neg\varphi & \text{assumption}\\
2 & p(\neg\varphi)=1 & 1, PITL6\\
3 & p(\varphi)=0 & PITL2
\end{array}$$

*PITL*7:

1	$(\varphi;\top;\ell=\infty)\vee((\varphi;\top)\wedge\ell=\infty)\Rightarrow(\top\Rightarrow(\varphi;\top))$	*ITL*
2	$(\varphi;\top)\Rightarrow p(\varphi;\top)=1$	$P_{\leq}^{\infty}$

*PITL*8:

1	$p(\varphi)=1\wedge p(\psi)=x\Rightarrow p(\varphi\wedge\psi)+p(\varphi\vee\psi)=1+x$	P_+
2	$\varphi\Rightarrow(\varphi\vee\psi)$	*ITL*
3	$p(\varphi)\leq p(\varphi\vee\psi)$	$2, PITL1$
4	$p(\varphi)=1\Rightarrow p(\varphi\vee\psi)=1$	$3, PITL5$
5	$p(\varphi)=1\wedge p(\psi)=x\Rightarrow p(\varphi\wedge\psi)=x$	$1,4$

*PITL*9:

1	$p(\varphi\Rightarrow\psi)=1\wedge p(\varphi)=1\Rightarrow p((\varphi\Rightarrow\psi)\wedge\psi)=1$	*PITL*8
2	$(\varphi\Rightarrow\psi)\wedge\psi\Rightarrow\psi$	
3	$p((\varphi\Rightarrow\psi)\wedge\psi)\leq p(\psi)$	$2, PITL1$
4	$p(\psi)\leq 1$	*PITL*5
5	$p(\varphi\Rightarrow\psi)=1\Rightarrow(p(\varphi)=1\Rightarrow p(\psi)=1)$	$1-4$

1	$p((\varphi\Rightarrow\psi)\Rightarrow(\neg\psi\Rightarrow\neg\varphi))=1$	*PITL*6
2	$p(\varphi\Rightarrow\psi)=1\Rightarrow p(\neg\psi\Rightarrow\neg\varphi)=1$	$1, PITL9$
3	$p(\neg\psi\Rightarrow\neg\varphi)=1\Rightarrow(p(\neg\psi)=1\Rightarrow p(\neg\varphi)=1)$	*PITL*9
4	$p(\neg\psi\Rightarrow\neg\varphi)=1\Rightarrow(p(\neg\psi)=0\Rightarrow p(\neg\varphi)=0)$	$3, PITL2$
5	$p(\varphi\Rightarrow\psi)=1\Rightarrow(p(\neg\psi)=0\Rightarrow p(\neg\varphi)=0)$	$2,4$

*PITL*10:

1	$p(\varphi)+p(\psi)>1\Rightarrow p(\varphi\wedge\psi)+p(\varphi\vee\psi)>1$	P_+
2	$p(\varphi\vee\psi)\leq 1$	*PITL*5
3	$p(\varphi)+p(\psi)>1\Rightarrow p(\varphi\wedge\psi)>0$	$1,2$

$P'_;$:

1	$(\varphi;p(\psi)=x)\Rightarrow\exists t((\varphi\wedge\ell=t;\top)\wedge(\ell=t;p(\psi)=x))$	*ITL*
2	$(\varphi\wedge\ell=t;\top)\Rightarrow p(\varphi\wedge\ell=t;\top)=1$	*PITL*7
3	$(\ell=t;p(\psi)=x)\Rightarrow p(\ell=t;\psi)=x$	$P_;$
4	$p(\varphi\wedge\ell=t;\top)=1\wedge p(\ell=t;\psi)=x\Rightarrow p(\varphi\wedge\ell=t;\psi)=x$	$PITL8, PITL1, ITL$
5	$(\varphi\wedge\ell=t;\psi)\Rightarrow(\varphi;\psi)$	*ITL*
6	$p(\varphi\wedge\ell=t;\psi)=x\Rightarrow p(\varphi;\psi)\geq x$	$5, PITL1$
7	$\exists t((\varphi\wedge\ell=t;\top)\wedge(\ell=t;p(\psi)=x))\Rightarrow\exists t(p(\varphi;\psi)\geq x)$	$2-6, ITL$
8	$\exists t(p(\varphi;\psi)\geq x)\Leftrightarrow p(\varphi;\psi)\geq x$	*ITL*
9	$(\varphi;p(\psi)=x)\Rightarrow p(\varphi;\psi)\geq x$	$1,7,8$
10	$(\varphi;p(\psi)=x)\Leftrightarrow(\varphi;p(\neg\psi)=1-x)$	$PITL2, ITL$
11	$(\varphi;p(\neg\psi)=1-x)\Rightarrow p(\varphi;\neg\psi)\geq 1-x$	like $1-9$, but with $\neg\psi$ as ψ
12	$(p(\varphi;\psi)>x\wedge p(\varphi;\neg\psi)\geq 1-x)\vee(p(\varphi;\psi)\geq x\wedge p(\varphi;\neg\psi)>1-x)$ $\Rightarrow p((\varphi;\psi)\wedge(\varphi;\neg\psi))>0$	*PITL*10
13	$(\varphi;\psi)\wedge(\varphi;\neg\psi)\wedge\neg(\varphi\wedge(\varphi;\ell\neq 0);\top)\Rightarrow\bot$	*ITL*
14	$p((\varphi;\psi)\wedge(\varphi;\neg\psi)\wedge\neg(\varphi\wedge(\varphi;\ell\neq 0);\top))=0$	$13, PITL6$
15	$p(\neg(\varphi\wedge(\varphi;\ell\neq 0);\top))=1$	assumption, *PITL*6
16	$p((\varphi;\psi)\wedge(\varphi;\neg\psi))=0$	$14, 15, PITL8$
17	$p(\varphi;\neg\psi)\geq 1-x\wedge p(\varphi;\psi)\geq x\Rightarrow p(\varphi;\psi)\leq x\wedge p(\varphi;\neg\psi)\leq 1-x$	$12, 16, ITL$
18	$(\varphi;p(\psi)=x)\Rightarrow p(\varphi;\psi)=x$	$9, 11, 17, ITL$

A.5 The rule *Seq*

In the proof of the admissibility of *Seq* below $\varphi \wedge \ell \geq l \wedge \ell \leq h$ is abbreviated by φ_l^h.

1	$(\ell = 0 \wedge p(\alpha;\beta;\top) = 1;\top) \Rightarrow p(p(\alpha;\beta;\top) = 1 \wedge \ell = 0;\top) = 1$	$PITL7$
2	$(\alpha; p(\beta;\top) = x \wedge \ell = 0) \Rightarrow p(\alpha;\beta;\top) = x$	$P'_{;}$, assumptions
3	$\exists x((\alpha;\top) \Rightarrow (\alpha; p(\beta;\top) = x \wedge \ell = 0;\top))$	ITL
4	$p(\alpha;\beta;\top) = 1 \Rightarrow$ $\exists x((\alpha;\top) \Rightarrow p(\alpha;\beta;\top) = x \wedge (\alpha; p(\beta;\top) = x \wedge \ell = 0;\top) \wedge p(\alpha;\beta;\top) = 1)$	$2, 3, ITL$
5	$p(\alpha;\beta;\top) = 1 \Rightarrow \exists x((\alpha;\top) \Rightarrow (\alpha; p(\beta;\top) = 1 \wedge \ell = 0;\top))$	$4, ITL$
6	$p(\alpha;\beta;\top) = 1 \wedge (\alpha;\top) \Rightarrow (\alpha; p(\beta;\top) = 1 \wedge \ell = 0;\top)$	$5, ITL$
7	$p(p(\alpha;\beta;\top) = 1 \wedge (\alpha;\top) \Rightarrow (\alpha; p(\beta;\top) = 1 \wedge \ell = 0;\top)) = 1$	$6, PITL6$
8	$p(p(\alpha;\beta;\top) = 1;\top) = 1 \wedge p(\alpha;\top) = 1 \Rightarrow p(\alpha; p(\beta;\top) = 1 \wedge \ell = 0;\top) = 1$	$7, PITL9$
9	$\ell = 0 \wedge p(\alpha;\beta;\top) = 1 \wedge p(\alpha;\top) = 1 \Rightarrow p(\alpha; p(\beta;\top) = 1 \wedge \ell = 0;\top) = 1$	$1, 8$
10	$p(\alpha;\beta;\top) = 1 \Rightarrow p(\alpha;\top) = 1$	$PITL9$
11	$\ell = 0 \wedge p(\alpha;\beta;\top) = 1 \Rightarrow p(\alpha; p(\beta;\top) = 1 \wedge \ell = 0;\top) = 1$	$9, 10$
12	$p(\alpha; p(\beta;\top) = 1 \wedge \ell = 0;\top) = 1 \wedge p(\alpha; p(\beta;\top) \neq 1 \wedge \ell = 0;\top) = x \Rightarrow$ $p((\alpha; p(\beta;\top) = 1 \wedge \ell = 0;\top) \wedge (\alpha; p(\beta;\top) \neq 1 \wedge \ell = 0;\top)) = x$	$PITL8$
13	$p((\alpha; p(\beta;\top) = 1 \wedge \ell = 0;\top) \wedge (\alpha; p(\beta;\top) \neq 1 \wedge \ell = 0;\top)) = 0$	$PITL6$
14	$p(\alpha; p(\beta;\top) = 1 \wedge \ell = 0;\top) = 1 \Rightarrow p(\alpha; p(\beta;\top) \neq 1 \wedge \ell = 0;\top) = 0$	$12, 13$
15	$\ell = 0 \wedge p(\alpha;\beta;\top) = 1 \Rightarrow p(\alpha; p(\beta;\top) \neq 1 \wedge \ell = 0;\top) = 0$	$11, 14$
16	$\ell = 0 \Rightarrow (p(\beta_{l_2}^{h_2};\top) \neq x_2 \Rightarrow p(\beta;\top) \neq 1)$	$[ET_\beta \in [l_2, h_2]]_{x_2}$
17	$\alpha_{l_1}^{h_1} \Rightarrow \alpha$	ITL
18	$(\alpha_{l_1}^{h_1}; p(\beta_{l_2}^{h_2};\top) \neq x_2 \wedge \ell = 0;\top) \Rightarrow (\alpha; p(\beta;\top) \neq 1 \wedge \ell = 0;\top)$	$16, 17, ITL$
19	$\ell = 0 \wedge p(\alpha;\beta;\top) = 1 \Rightarrow p(\alpha_{l_1}^{h_1}; p(\beta_{l_2}^{h_2};\top) \neq x_2 \wedge \ell = 0;\top) = 0$	$15, 18, PITL6, PITL9$
20	$\neg(\alpha \wedge (\alpha; \ell \neq 0);\top) \Rightarrow$ $((\alpha_{l_1}^{h_1}; p(\beta_{l_2}^{h_2};\top) \neq x_2 \wedge \ell = 0;\top) \Leftrightarrow (\alpha_{l_1}^{h_1};\top) \wedge (\alpha; p(\beta_{l_2}^{h_2};\top) \neq x_2 \wedge \ell = 0;\top))$	ITL
21	$p(\neg(\alpha \wedge (\alpha; \ell \neq 0);\top)) = 1$	assumption, $PITL6$
22	$p(\alpha_{l_1}^{h_1}; p(\beta_{l_2}^{h_2};\top) \neq x_2 \wedge \ell = 0;\top) = 0 \Leftrightarrow$ $p((\alpha_{l_1}^{h_1};\top) \wedge (\alpha; p(\beta_{l_2}^{h_2};\top) \neq x_2 \wedge \ell = 0;\top)) = 0$	$20, 21, PITL9$
23	$\alpha \wedge p(\alpha; \beta_{l_2}^{h_2};\top) \neq x_2 \Rightarrow \neg(\alpha; p(\beta_{l_2}^{h_2};\top) = x_2 \wedge \ell = 0)$	$P'_{;}$
24	$\neg(\alpha; p(\beta_{l_2}^{h_2};\top) = x_2) \wedge \alpha \Rightarrow (\alpha; p(\beta_{l_2}^{h_2};\top) \neq x_2 \wedge \ell = 0)$	ITL
25	$\alpha \wedge p(\alpha; \beta_{l_2}^{h_2};\top) \neq x_2 \Rightarrow (\alpha; p(\beta_{l_2}^{h_2};\top) \neq x_2 \wedge \ell = 0)$	$23, 24$
26	$p((\alpha_{l_1}^{h_1};\top) \wedge (\alpha; p(\beta_{l_2}^{h_2};\top) \neq x_2 \wedge \ell = 0;\top)) = 0 \Rightarrow$ $p((\alpha_{l_1}^{h_1};\top) \wedge (\alpha \wedge p(\alpha; \beta_{l_2}^{h_2};\top) \neq x_2;\top)) = 0$	$25, PITL9, PITL6$
27	$\neg(\alpha \wedge (\alpha; \ell \neq 0);\top) \Rightarrow$ $((\alpha_{l_1}^{h_1};\top) \wedge (\alpha \wedge p(\alpha; \beta_{l_2}^{h_2};\top) \neq x_2;\top) \Leftrightarrow (\alpha_{l_1}^{h_1} \wedge p(\alpha; \beta_{l_2}^{h_2};\top) \neq x_2;\top))$	ITL
28	$p((\alpha_{l_1}^{h_1};\top) \wedge (\alpha \wedge p(\alpha; \beta_{l_2}^{h_2};\top) \neq x_2;\top)) = 0 \Leftrightarrow p(\alpha_{l_1}^{h_1} \wedge p(\alpha; \beta_{l_2}^{h_2};\top) \neq x_2;\top) = 0$	$21, 27, PITL9$
29	$\ell = 0 \wedge p(\alpha;\beta;\top) = 1 \Rightarrow p(\alpha_{l_1}^{h_1} \wedge p(\alpha; \beta_{l_2}^{h_2};\top) \neq x_2;\top) = 0$	$19, 22, 26, 28$
30	$\ell = 0 \wedge p(\alpha_{l_1}^{h_1} \wedge p(\alpha; \beta_{l_2}^{h_2};\top) \neq x_2;\top) = 0 \Rightarrow$ $p((\alpha_{l_1}^{h_1};\top) \wedge (\alpha; \beta_{l_2}^{h_2};\top)) = x_2.p(\alpha_{l_1}^{h_1};\top)$	$\overline{P}, \underline{P}$, assumptions
31	$\neg(\alpha \wedge (\alpha; \ell \neq 0);\top) \Rightarrow ((\alpha_{l_1}^{h_1};\top) \wedge (\alpha; \beta_{l_2}^{h_2};\top) \Leftrightarrow (\alpha_{l_1}^{h_1}; \beta_{l_2}^{h_2};\top))$	ITL
32	$(\alpha_{l_1}^{h_1};\top) \wedge (\alpha; \beta_{l_2}^{h_2};\top) \Leftrightarrow (\alpha_{l_1}^{h_1}; \beta_{l_2}^{h_2};\top)$	assumption, 31
33	$p((\alpha_{l_1}^{h_1};\top) \wedge (\alpha; \beta_{l_2}^{h_2};\top)) = p(\alpha_{l_1}^{h_1}; \beta_{l_2}^{h_2};\top)$	$32, PITL1$
34	$\ell = 0 \wedge p(\alpha;\top) = 1 \Rightarrow p(\alpha_{l_1}^{h_1};\top) = x_1$	$[ET_\alpha \in [l_1, h_1]]_{x_1}$
35	$\ell = 0 \wedge p(\alpha;\beta;\top) = 1 \Rightarrow p(\alpha_{l_1}^{h_1}; \beta_{l_2}^{h_2};\top) = x_2.x_1$	$10, 29, 30, 33, 34$