Proc. Estonian Acad. Sci. Engng., 1998, 4, 2, 130–137
https://doi.org/10.3176/eng.1998.2.05

COMBINING COMMUNICATING SEQUENTIAL PROCESSES AND TEMPORAL LOGIC

Justin PEARSON[a] and Jeremy BRYANS[b]

[a] Department of Information Technology, Mid Sweden University of Science & Engineering, MSU-ITE, S-851 70 Sundsvall, Sweden; e-mail: justin@nts.mh.se
[b] Computing Laboratory, University of Kent, Canterbury, Kent, CT 2 7NF, United Kingdom; e-mail: J.W.Bryans@ukc.ac.uk

Received 19 January 1998, in revised form 23 February 1998

Abstract. This work presents a novel method for specifying real-time processes. The contribution is threefold: we present a novel denotational model for the process algebra of communicating sequential processes (CSP); we show how to interpret a real-time temporal logic over this model; finally, we develop a theory of refinement which encompasses processes and temporal specifications. This allows us to separately develop and refine the timed and the behavioural aspects of specifications, and combine the two into a timed CSP process at the end of the development.

Key words: real-time CSP, temporal logic, specification, refinement.

1. INTRODUCTION

In this paper, we illustrate a novel method for the specification of real-time processes. Our framework employs the process algebra of communicating sequential processes (CSP) [1] for the specification of the behavioural aspects of a system and the propositional temporal logic (PTL) [2] for the specification of its temporal properties. We can separately develop and refine these two aspects, and then combine them into a real-time CSP process at the end of the development.

Underpinning this work is a new denotational model $\mathcal{M}_U$ for real-time CSP, discussed in Section 2, that allows unguarded CSP processes as specifications. In Section 3, the temporal logic PTL is presented and interpreted over the CSP model $\mathcal{M}_U$, and thus the model $\mathcal{M}_U$ is a common semantic framework for both aspects of the development process. In Section 4, we give a definition of what it means for a

process to satisfy a PTL formula and present a number of rules for determining if a particular process satisfies a particular formula.

We then consider a specification to be a pair (P, ϕ), where P is a CSP process and ϕ is a formula of temporal logic. In Section 5, we develop a theory of refinement based on this idea.

2. THE CSP MODEL

The syntax used in this paper is presented below. It is largely similar to most other timed CSP models [3]. The significant differences are: first, we include a second prefix operator ($\mapsto$), and secondly, to simplify the presentation, we do not include message passing or variables.

If P is a process, t is a time value ($t \in [0, \infty)$), A is a set of events, a is a single event and I is an index set, then the syntax used in this paper is

$$P ::= \bot \mid Stop \mid Skip \mid Wait(t) \mid P \,|[A]|\, P \mid P \,\Box\, P \mid \\ \sqcap_{i \in I} P \mid a \rightarrow P \mid a \mapsto P \mid P \,;\, P \mid P \,\triangleright\{t\}\, P \mid P \setminus A$$

These represent: the most nondeterministic process, $\bot$; the broken process, *Stop*; successful termination, *Skip*; delay, $Wait(t)$; parallel composition, $P \,|[A]|\, P$; external choice, $P \,\Box\, P$; nondeterministic (or internal) choice, $\sqcap_{i \in I} P$; first form of action prefix, $a \rightarrow P$; second form of action prefix, $a \mapsto P$; sequential composition, $P \,;\, P$; timeout, $P \,\triangleright\{t\}\, P$ and hiding, $P \setminus A$.

We will develop a framework where a specifier may use an untimed CSP (the language above, with the exclusion of the wait, timeout and second action prefix operators) to describe the behaviour of the system. The first action prefix operator ($a \rightarrow P$) says nothing about the timing of its arguments, while the second ($a \mapsto P$) insists that the action a must be available immediately and that the process P is invoked immediately subsequent to the action a being performed. Thus, the operator $\mapsto$ takes the role of the standard real-time CSP action prefix, while the process $a \rightarrow P$ nondeterministically allows arbitrary delays before the action a is offered, and between the action a being performed and the process P being invoked.

2.1. The semantic model

A denotational semantic model identifies a process with the set of all observations that may be made of it. These sets are subject to certain *consistency conditions*, similar to those outlined in [4]. These insist, for example, that a set containing a particular observation must also contain all prefixes of that observation.

The novelty of the model $\mathcal{M}_U$ lies in the fact that it allows processes of the form $P = a \mapsto P$, where there need be no delay between recursive instantiations of P.

We assume a set of actions Σ and use these to build *timed actions*: pairs of the form (t, a), where t is a time value and $a \in \Sigma$.

An observation within the model $\mathcal{M}_U$ is a triple, (s, X, d), where s is a timed trace, X is a timed refusal, and d is a divergence value.

A *timed trace* is a chronologically ordered sequence of timed actions and may be finite or infinite. A *timed refusal* is a set of timed actions. A *divergence value* is a non-negative real number or the symbol ∞.

If an observation (s, X, d) is part of the semantic set of a process P, then we say that the P can *perform* the observation (s, X, d). This means that it may perform the timed actions indicated by the trace s, while refusing to perform the timed actions in the set X. If d is a real number, then the process was observed to have diverged by the time d, and if d is the symbol ∞, then no divergence was observed.

We define an information ordering over observations, such that any observation of a process may be extended. This may be done in two possible ways.

An observation is an *extension* of another observation if it contains the same trace, and more complete refusal or divergence information, if it has an extended trace. Formally:

Definition 1. *An observation* (s', X', d') *is an extension of an observation* (s, X, d), *written* $(s', X', d') \geqslant_E (s, X, d)$ *if*

$$s \leqslant s' \wedge X \subseteq X' \wedge d \geqslant d' \text{ if } d \text{ is finite, otherwise } d' \geqslant \mathrm{end}(s, X, d),$$

where $\mathrm{end}(s, X, d)$ *denotes the largest time value in the observation, and* $s \leqslant s'$ *stipulates that the trace* s *is a prefix of the trace* s'.

Observations which contain complete information for every point in time, up to the end of the observation, are called *point-wise maximal observations*.

Formally:

Definition 2. (s, X, d) *is point-wise maximal in* $[\![P]\!]$ *whenever*

$$\begin{aligned} &\forall (s', X', d') \in [\![P]\!] . (s', X', d') \geqslant_E (s, X, d) \Rightarrow \\ &\quad (s', X', d') = (s, X, d) \\ &\quad \vee \\ &\quad \mathrm{end}(s', X', d') > \mathrm{end}(s, X, d) \end{aligned}$$

3. PROPOSITIONAL TEMPORAL LOGIC

In this section, we introduce the logic, PTL [2], which will be used for temporal specifications. Our version of PTL has three classes of atomic propositions. The atom $\mathbf{P}_a$ expresses the fact that a process does the action a, $\mathbf{O}_a$ that the action a

is offered, and **D** that the process has diverged. The logic is defined in terms of a satisfaction relation $\models_t$ between observations and formulae of the logic. The expression $(s, X, d) \models_t \phi$ asserts that the formula ϕ holds on the observation (s, X, d) at time t.

We define $\models_t$ for $t > \text{end}(s, X, d)$ as

- $(s, X, d) \not\models_t \phi$

and for $t \leqslant \text{end}(s, X, d)$, we define $\models_t$ inductively as follows. For atoms:

- $(s, X, d) \models_t \mathbf{D}$ whenever $d \leqslant t$;
- $(s, X, d) \models_t \mathbf{P}_a$ whenever $(t, a) \in s$;
- $(s, X, d) \models_t \mathbf{O}_a$ whenever $(t, a) \notin X$.

So the assertion $(s, X, d) \models_t \mathbf{P}_a$ asserts that the action a is actually performed at time t. Note that we use negative information to characterize offers. We say that an observation (s, X, d) offers an a at time t if it does not refuse it.

- $(s, X, d) \models_t \phi \wedge \psi$ whenever $(s, X, d) \models_t \phi$ and $(s, X, d) \models_t \psi$;
- $(s, X, d) \models_t \neg \phi$ whenever $(s, X, d) \not\models_t \phi$;
- $(s, X, d) \models_t \phi \; \mathcal{U}_{<\tau} \psi$ whenever

$$\exists\, (\textit{finite}) t_1 : t_1 > t \text{ and } \mid t - t_1 \mid < \tau \bullet$$
$$(s, X, d) \models_{t_1} \psi \text{ and}$$
$$\forall\, t_2 : t < t_2 < t_1 \; (s, X, d) \models_{t_2} \phi;$$

- $(s, X, d) \models_t \phi \; \mathcal{S}_{<\tau} \psi$ whenever

$$\exists\, t_1 : t_1 < t \text{ and } \mid t - t_1 \mid < \tau \bullet$$
$$(s, X, d) \models_{t_1} \phi \text{ and}$$
$$\forall\, t_2 : t_1 < t_2 < t \; (s, X, d) \models_{t_2} \psi.$$

The logic also includes the next state and previous state operators, but these are omitted due to lack of space.

The other operators can be derived in the usual way: $\phi \vee \psi$ is defined as $\neg (\neg \phi \wedge \neg \psi)$; $\phi \Rightarrow \psi$ is defined as $\psi \vee \neg \phi$; $\Diamond_{<\tau} \phi$ is defined as $\text{true} \; \mathcal{U}_{<\tau} \phi$ and $\Box_{<\tau} \phi$ is defined as $\neg \Diamond_{<\tau} (\neg \phi)$.

To define what it means for a process to satisfy a PTL formula, we make use of the $\geqslant_E$ ordering on observations, and of maximal refusal sets (a refusal set is a maximal refusal set if it contains all possible refusal information up to the time of the end of the trace.)

Definition 3. *A process P satisfies ϕ if for all observations (s, X, d) of process P, where X is a maximal refusal set, we have that either $(s, X, d) \models_0 \phi$ or $\exists (s', X', d') \in \mathcal{F}_U[\![P]\!].(s', X', d') \geqslant_E (s, X, d) \bullet (s', X', d') \models_0 \phi$.*

4. RULES FOR FINITE PROCESSES

In this section, we present a representative subset of rules for finite processes (processes which are constructed without recursion), which connect the process constructors with the rules of the logic. Each rule has the form

$$\frac{\text{antecedent}}{\text{conclusion}}$$

which reads that the antecedent implies the conclusion. Rules with no antecedent read that the conclusion is always true.

The *Stop* process simply refuses all actions for all time. Therefore we have a family of rules

$$\frac{}{Stop \models \Box(\neg \mathbf{O}_a)}$$

for each possible action a.

The semantics of the binary internal choice $P \sqcap Q$ is simply the union of the behaviours of the two processes. This makes the rule for internal choice particularly simple:

$$\frac{P \models \phi \quad Q \models \psi}{P \sqcap Q \models \phi \vee \psi}.$$

The behaviours of $P \,\Box\, Q$ is a subset of the union of the behaviours of P and Q, and so we have the following rule:

$$\frac{P \models \phi \quad Q \models \phi}{P \,\Box\, Q \models \phi}.$$

In the semantics of the immediate prefix process $a \mapsto P$, the process is able to offer an a and when it performs the a, it acts as P. But such a process need not actually perform the action a if placed in an uncooperative environment. This leads to the following rule:

$$\frac{P \models \phi}{a \mapsto P \models \mathbf{O}_a \,\mathcal{U}_{<\infty}(\mathbf{P}_a \Rightarrow \phi)}.$$

The skip process simply offers the tick action $\surd$, until it is performed

$$\frac{}{Skip \models \mathbf{O}_{\surd} \,\mathcal{U}_{<\infty} \mathbf{P}_{\surd}}.$$

The process $P ; Q$ will behave as P until P offers a $\surd$ action, at which time it will behave as process Q. The effect of the compose operator is that the $\surd$ action becomes urgent and hidden. This leads to the rule

$$\frac{P \models \phi \quad Q \models \psi \quad P \models \text{true } \mathcal{U}_{<\tau} \mathbf{O}_{\checkmark}}{P \,;\, Q \models \phi \; \mathcal{U}_{<\tau} \psi}$$

with the extra side condition that ϕ does not contain $\mathbf{P}_{\checkmark}$ or $\mathbf{O}_{\checkmark}$ as subformulae.

The rule for the timeout operator is

$$\frac{P \models \phi \quad Q \models \psi}{P \triangleright \{t\}\, Q \models \phi \; \mathcal{U}_{<\tau} \psi}$$

with the side condition that the process formula ϕ must contain no subformulae of the form $\phi' \; \mathcal{U}_{<t'} \psi'$ with $t' > t$.

For hiding, we have the following rule:

$$\frac{P \models \phi}{P \setminus A \models \phi \wedge (\bigwedge_{a \in A} \neg(\mathbf{O}_a \vee \mathbf{P}_a))}$$

provided $\forall a \in A \neg\, \mathbf{O}_a, \mathbf{P}_a$ are not subformulae of ϕ.

The rules presented here are not complete with respect to the CSP model – not everything expressible in CSP is expressible in the logic. This is not a problem for us, since we intend that particular requirements be captured in the most appropriate formalism.

5. SPECIFICATION AND REFINEMENT

In this section, we define the notion of a process specification pair. This gives a way of combining two specification styles: CSP refinement and specification by temporal logic formulae. We can then deal with separate aspects of a design separately, by capturing the behaviour using CSP processes and the timing information using PTL.

Broadly, a process specification pair (P, ϕ) is the set of all behaviours of P which satisfy the formula ϕ subject to certain consistency conditions. This leads to the following definition:

Definition 4. *A process specification pair is a pair (P, ϕ) consisting of a CSP process and a PTL formula. It is defined in terms of observations as*

$$\begin{aligned}(P, \phi) = \{(s, X, d) \in [\![P]\!] \mid \\ \exists\, \textit{pointwise-maximal observation}(s', X', d') \bullet \\ (s', X', d') \geqslant_E (s, X, d) \wedge (s', X', d') \models \phi \wedge \\ \forall (s'', X'', d'') \geqslant_E (s', X', d') \textit{with } X'' \textit{ maximal} \\ (s'', X'', d'') \models \phi\},\end{aligned}$$

where (s', X', d') and (s'', X'', d'') are observations of P.

It is important to realize that this pair may not define a legitimate process. If the requirements set by the CSP specification and the PTL formula are inconsistent, then the specification pair will contain observations which cannot be extended in time. These correspond to *time-stop* requirements – requirements which cannot be implemented.

We now define the standard notation of semantic entailment for formulae of the logic.

Definition 5. *A temporal formula ψ follows from a temporal logic formula ϕ (written $\phi \vdash \psi$) if* $\forall X.X \models \psi \Rightarrow X \models \phi$.

Given two specifications (P, ϕ) and (Q, ψ), we can define a refinement notion exactly analogous with the standard CSP refinement.

Definition 6. *(P, ϕ) is refined by (Q, ψ) (written $(P, \phi) \sqsubseteq (Q, \psi)$) whenever* $(Q, \psi) \subseteq (P, \phi)$.

We are now able to present our general refinement theorem.

Theorem.

$$\frac{P \sqsubseteq Q \quad \psi \vdash \phi}{(P, \phi) \sqsubseteq (Q, \psi)}.$$

Proof. Consider $(s, X, d) \in (Q, \psi)$. We know $(s, X, d) \in [\![P]\!]$. But since $\psi \vdash \phi$, we know that $(s, X, d) \models \phi$ (or some extension does), therefore $(s, X, d) \in (P, \phi)$.

6. CONCLUSION

In this paper, we have developed a novel framework of refinement, based on separately specifying behavioural and timing requirements. Each type of requirement is specified, using an appropriate language: CSP for behavioural requirements, and PTL for timing requirements. We have given rules to decide when processes and temporal logic formulae are compatible, and a general refinement theorem for process specification pairs.

REFERENCES

1. Hoare, C. A. R. *Communicating Sequential Processes*. Prentice Hall, Englewood Cliffs, N. J., 1985.
2. Emerson, E. A. Temporal and modal logic. In *Handbook of Theoretical Computer Science* (Leeuwen, J. van, ed.). Elsevier Science Publishers, Amsterdam, 1990.
3. Davies, J. *Specification and proof in real time CSP*. Cambridge University Press, Cambridge, 1993.
4. Bryans, J. W. *Denotational Semantic Models for Real-time LOTOS*. PhD thesis. Reading University, UK, Nov. 1996.

SUHTLEVATE JADAPROTSESSIDE JA TEMPORAALLOOGIKA KOOSLUSEST

Justin PEARSON ja Jeremy BRYANS

On esitatud uudne reaalajaprotsesside spetsifitseerimise meetod käsitledes protsessialgebra suhtlevate jadaprotsesside uut denotatsioonmudelit, reaalaja temporaalloogika interpreteerimist sellel mudelil ning protsesside ja nende spetsifikatsioonide täpsustamise täiendatud teooriat. Meetod võimaldab eraldi kirjeldada ja konkretiseerida spetsifikatsioonide talitluslikke ja ajalisi aspekte ning arendussammude järel need taas kokku sulatada ajaga suhtlevateks jadaprotsessideks.