BRICS

Basic Research in Computer Science

BRICS RS-97-11 Čerāns et al.: Timed Modal Specification — Theory and Tools

Timed Modal Specification — Theory and Tools

Kārlis Čerāns
Jens Chr. Godskesen
Kim G. Larsen

BRICS Report Series **RS-97-11**

ISSN 0909-0878 **April 1997**

See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/97/11/`

Timed Modal Specification — Theory and Tools*

Kārlis Čerāns†, Jens Chr. Godskesen‡and Kim G. Larsen§

Abstract

In this paper we present the theory of *Timed Modal Specifications* (TMS) together with its implementation, the tool Epsilon. TMS and Epsilon are timed extensions of respectively *Modal Specifications* [Lar90, LT88] and the Tav system [GLZ89, BLS92].

The theory of TMS is an extension of real–timed process calculi with the specific aim of allowing *loose* or *partial* specifications. Looseness of specifications allows implementation details to be left out, thus allowing several and varying implementations. We achieve looseness of specifications by introducing two modalities to transitions of specifications: a *may* and a *must* modality. This allows us to define a notion of *refinement*, generalizing in a natural way the classical notion of bisimulation. Intuitively, the more must–transitions and the fewer may–transitions a specification has, the finer it is. Also, we introduce notions of refinements abstracting from time and/or internal computation.

TMS specifications may be combined with respect to the constructs of the real–time calculus [Wan90]. "Time–sensitive" notions of refinements that are preserved by these constructs are defined [1], thus enabling compositional verification.

Epsilon provides automatic tools for verifying refinements. We apply Epsilon to a compositional verification of a train crossing example.

1 Introduction

In this paper we present the theory of *Timed Modal Specifications* (TMS) together with its implementation, the tool Epsilon. TMS and Epsilon are timed extensions of respectively *Modal Specifications* [Lar90, BL90, LT88, HL89] and the Tav system [GLZ89, BLS92].

During the last few years various process calculi have been extended to include real–time in order to handle quantitative aspects of real–time systems, for instance that some critical event must not or should happen within a certain time period. We mention the calculi defined in [Wan90] and the ones defined in [NSY91] and [BB89]. Common to these real–time calculi is that time is represented by some dense time domain, e.g. the non–negative reals.

*This work has been supported by the Danish National Science Research Council project DART and the ESPRIT Basic Research Action 7166, CONCUR2. For the first author part of this work was performed while on a post–doc leave at Chalmers University of Technology, Göteborg, Sweden.

†Adress: Inst. of Math. and Comp. Sc., University of Latvia., Rainis blvd. 29, LV–1459 Riga, Latvia. E–mail: karlis@mii.lu.lv.

‡Address: Tele Danmark Research, Lyngsø Allé 2, DK–2970 Hørsholm, Denmark. E–mail: jcg@tdr.dk

§Address: Dep. of Math. and Comp. Sc., Aalborg University, Fredrik Bajers Vej 7, 9220 Aalborg, Denmark. E–mail: kgl@iesd.auc.dk.

[1] To be precise, when abstracting from internal composition refinement is not preserved by choice for the usual reasons.

As argued in [Lar90] process calculi are often too concrete in the sense that when a system has been specified the set of possible implementations are restricted to one and only one equivalence class of processes (e.g. the class of bisimulation [Mil89, Par81] equivalent processes). Moreover, as correctness is given by the equivalence, the set of possible implementations remains constant under refinement. Hence, stepwise development methodologies are not well supported in classical process calculi. As an example, using the notation from [Wan90], a disposable medium $M_d^{a,b}$ with some delay d between input and output can be specified by

$$M_d^{a,b} \stackrel{def}{=} a.\epsilon(d).\overline{b}$$

meaning that after input a the output $\overline{b}$ is first enabled after d time units. But, as a specification this may be too precise and perhaps all that needs to be required of the medium is that it enables its output at some point in the time interval $[e, f]$ after a message has been received. Hence, using a suggestive notation, a more loose specification like

$$M_{e,f}^{a,b} \stackrel{def}{=} a.\epsilon[e, f].\overline{b}$$

is needed. Intuitively, we want $M_{e,f}^{a,b}$ to mean that after input a the output $\overline{b}$ may be enabled in the interval $[e, f)$ but is first guaranteed to be enabled after f time units. It is however impossible to give a loose specification, like $M_{e,f}^{a,b}$, using process calculi.

The theory of TMS is an extension of real–timed process calculi with the specific aim of allowing *loose* or *partial* specifications. Looseness here means that a specification S can have various implementations because implementation details may be left out in S. For instance, as in the example above, we can be liberal as to how long a medium may delay before it can deliver. The looseness of specifications is achieved by introducing two modalities to transitions: a *may* and a *must* modality, denoted by indices $\Diamond$ and $\Box$ respectively on actions. Using modalities the loosely specified media above can be specified by

$$S_{e,f}^{a,b} \stackrel{def}{=} a_{\Box}.(\epsilon(e).\overline{b}_{\Diamond} + \epsilon(f).\overline{b}_{\Box})$$

that is, $S_{e,f}^{a,b}$ specifies disposable media that must input a. After e time units from reception, but not before, $\overline{b}$ may be enabled but only after f time units the enabling of $\overline{b}$ is *required*. Obviously we expect $M_d^{a,b}$ to implement $S_{e,f}^{a,b}$ whenever $d \in [e, f]$.

Generalizing in a natural way the notion of bisimulation we introduce a *refinement ordering* $\lhd$ between timed modal specifications. As indicated above, a timed modal specification may specify a whole range of implementations or processes. Thus conceptually one may view a modal specification S as the set of processes satisfying S, and the refinement ordering attempts to capture the corresponding set inclusion between specifications. Intuitively, we expect a modal specification S to be a refinement of specification T when all transition *allowed* by S are also allowed by T, and, conversely, all transitions *required* by T are also required by S. As an example, we expect $S_{e,f}^{a,b} \lhd S_{g,h}^{a,b}$, whenever $g \leq e$ and $f \leq h$.

In practical applications it is often advantageous to abstract from certain aspects when analyzing a system. In particular, internal computation will normally be considered unobservable, and in a first analysis of a large combined system explicit timing information may be irrelevant. In the paper we therefore introduce notions of refinements abstracting from time and internal computation, and, of course, our tool Epsilon supports automatic verification based on the refinements presented. For total specifications, i.e. specifications with no looseness in the sense that all events are labelled with $\Box$–modalities, our refinements collapses into standard process calculi (bisimulation) equivalences. We therefore consider TMS a conservative extension of timed process calculi,

The automatic refinement checking for TMS can be performed through adopting the techniques for checking bisimulation equivalences between networks of timed regular processes, developed in [Č92b] for timed (time–sensitive) and [LW90] for time–abstracted cases (an alternative approach for deciding time–abstracted equivalences can be found in [ACH+92]). We have implemented these technique in the tool Epsilon so indeed automatic refinement checking between TMS specifications is feasible. For untimed specifications the algorithms of Epsilon coincide with those of the Tav [GLZ89, BLS92] system[2].

We intend TMS and Epsilon to be useful during the process of design and implementation. In particular, we want to support system development through *stepwise refinement*. In a stepwise refinement development of a system the initial specification is rather abstract permitting a wide range of implementations. An idealized development now consists in a series of small and successive refinements, each restricting the set of permitted implementations, until eventually an implementation can be extracted directly. Each refinement can be relatively small, consisting typically in the replacement of a single component of the current specification with more concrete ones. To illustrate the first step of an idealized stepwise refinement development, suppose we have an initial specification of disposable media, say

$$S^{a,b}_{e,f}$$

required to deliver in the interval from e to f after it receives its input. A possible refinement step could be to replace $S^{a,b}_{e,f}$ by the composed specifications

$$(M^{a,c}_{d} \mid S^{c,b}_{e-d,f-d})\backslash c \tag{1}$$

That is, the initial specification has been refined to a more concrete specification demanding the implementation to consists of a medium with a fixed delay d ($d \leq e$) together with some medium delivering between $e - d$ to $f - d$ time units after it received. The two components communicate via the internal channel c.[3] The refined specification may be considered more concrete because structural information has been added to the specification. Obviously, we expect (1) to be a correct refinement of $S^{a,b}_{e,f}$ since the total delay is still within the interval from e to f. Using the verification tool Epsilon we can automatically prove the correctness of this refinement step; more precisely, we can prove

$$(M^{a,c}_{d} \mid S^{c,b}_{e-d,f-d})\backslash c \;\trianglelefteq\; S^{a,b}_{e,f}$$

where $\trianglelefteq$ indicates a refinement abstracting from internal events.

In general we would like the correctness of a refinement step to be immediately implied by the correctness of the refinement of the replaced component by the one replacing it as this obviously will greatly simplify the task of verification. That is, we want to support *compositional* verification and hence the refinements to be preserved by composition as much as possible. As an example, we want to be able to infer that the combined medium

$$(M^{a,c}_{d} \mid M^{c,b}_{f-d})\backslash c$$

is a correct implementation of $S^{a,b}_{e,f}$ directly from (1) and the obvious fact that $M^{c,b}_{f-d} \trianglelefteq S^{c,b}_{e-d,f-d}$. Clearly, this inference is possible provided the refinement $\trianglelefteq$ is preserved by parallel composition and restriction.

[2]Tav is a system for deciding various equivalences between CCS processes [Mil89].

[3]As usual we take $S\backslash c$ to mean S but restricted from c.

TMS and the various notions of refinements also makes our correctness proofs far more *general* than correctness proofs within standard (timed) process calculi. That is, a single correctness proof in TMS may capture a whole (possibly) infinite family of correctness proofs in process calculi. For instance, in the example above the correctness proof of (1) establishes in a single refinement the fact that $(M_d^{a,c} \mid M)\backslash c$ is a correct implementation of $S_{e,f}^{a,b}$ whenever M is chosen from the infinite set $\{M_g^{c,b} \mid g \in [e-d, f-d]\}$.

Temporal logics with explicit time provides an alternative framework for expressing loose specifications of real–time systems. In fact there have been quite substantial work on model–checking with respect to such logics [ACD90], and automatic tools for carrying out the model–checking has been or are under implementation (e.g. [NSY92]). However, no results as to the composition of (timed) logical specifications has been offered so far, and compositional verification and stepwise refinement are therefore not supported. Rather the tools and techniques developed within the (timed) logical framework always compare a (final) implementation with the (initial) specification. In contrast, the theory of TMS and the Epsilon tool are intended to be used throughout the entire development process.

The remainder of this paper is organized as follows. The next section contains a brief outline of Modal Specifications. Section 3 gives the definition of TMS and provides an operational semantics for composition with respect to the constructs of TCCS. Various notions of abstracting refinements are introduced in Section 4. The theory underlying TMS and the tools of Epsilon are applied in Section 5 in the verification of a (classical) train crossing example. In Section 6 we propose a new refinement abstracting from internal events. The last section offers a compact outline of the actual refinement checking algorithms as carried out by Epsilon.

2 Modal Specifications

Modal Specifications (or Modal Transition Systems) were introduced in [LT88] in order to allow loose or partial specifications to be expressed in a process algebraic framework. Semantically, Modal Specifications are given an operational interpretation imposing restrictions on the transitions of possible implementations by telling which transitions are *necessary* and which are *admissible*. The transition systems for Modal Specifications therefore have *two* transition relations: $\longrightarrow_\Box$ describing the *required* transitions and $\longrightarrow_\Diamond$ describing the *allowed* transitions.

Definition 2.1 *A modal transition system is a structure $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \longrightarrow_\Box, \longrightarrow_\Diamond)$, where $\mathcal{S}$ is a set of specifications, $\mathcal{A}$ is set of actions and $\longrightarrow_\Box$, $\longrightarrow_\Diamond \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, satisfying the condition $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$.*

The condition $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$ simply says that anything required is also allowed, thus ensuring consistency of any Modal Specification. Clearly, the more a specification allows and the less it requires the looser the specification is. This idea is formalized in the following notion of refinement

Definition 2.2 *A refinement $\mathcal{R}$ is a binary relation on $\mathcal{S}$ such that whenever $S\mathcal{R}T$ and $a \in \mathcal{A}$ then the following holds:*

1. *Whenever $S \xrightarrow{a}_\Diamond S'$, then $T \xrightarrow{a}_\Diamond T'$ for some T' with $S'\mathcal{R}T'$,*

2. *Whenever $T \xrightarrow{a}_\Box T'$, then $S \xrightarrow{a}_\Box S'$ for some S' with $S'\mathcal{R}T'$.*

We say that S refines T whenever (S, T) is contained in some refinement $\mathcal{R}$. In this case we write $S \lhd T$.

The behaviour of processes themselves is assumed to be given in terms of a standard labelled transition system, which may be seen as a special case of Modal Specifications with all transitions being required (i.e. $\longrightarrow_\Box = \longrightarrow_\Diamond$). In this case the new notion of refinement coincides with the classical notion of bisimulation.

3 Timed Modal Specifications

The language we use to describe timed processes is the real–time calculus TCCS of Wang [Wan90]. This calculus is essentially Milner's CCS [Mil89] extended with a delay construct $\epsilon(d).P$, which informally means "wait for d units of time and then behave like the process P", where $d \in \mathbf{R}_+$ is a positive real.

The semantics of TCCS applies the "two–phase functioning principle" outlined in [NSY91]. That is, the behaviour of a real–time system is divided into two phases: one phase in which the components of the system agrees to let time pass, and a second phase in which the system computes by performing (instantaneous) actions. In the operational semantics of TCCS this is reflected by having transitions labelled by either *action names* or *delays* being positive reals.

Similar to the Modal Specification extensions of classical Process Algebra (e.g. CCS), and for the very same reasons,[4] we offer in the following a Modal Specification extension of the real–time calculus TCCS.

3.1 Informal Semantics

First consider the TCCS process term $a.P$. As a specification this term is quite specific, in that it *requires* an implementation *at any moment* to be able to perform the action a and implement P thereafter. This interpretation is formalized by the following *required* transitions for $a.P$

$$a.P \xrightarrow{a}_\Box P \qquad a.P \xrightarrow{\epsilon(d)}_\Box a.P \text{ for all } d > 0$$

In the following we shall adopt the notation $a_\Box.P$ for $a.P$.

To obtain looseness[5] we introduce a new *may* prefix construct $a_\Diamond.P$. As a specification this term will at any moment *allow* (without requiring) an implementation to have an a–transition as long as the result of such a transition implements P. Formally this is reflected in the following transitions

$$a_\Diamond.P \xrightarrow{a}_\Diamond P \qquad a_\Diamond.P \xrightarrow{\epsilon(d)}_\Box a_\Diamond.P \text{ for all } d > 0$$

[4] I.e. to obtain looseness in specifications.
[5] and analogous with the Modal Specification extension of CCS.

Informally, it should be rather clear that $a_\diamond.P$ is a fairly loose specification covering a range of different implementations. In particular, $a_\diamond.P$ will contain as typical implementations the processes $a.P$, for obvious reasons; nil, which never enables any actions and therefore trivially satisfies P after any possible a–transition, and, for any d, $\epsilon(d).a.P$, which possess no actions before having delayed d units after which any a–transition clearly will lead to a derivative implementing P. To formally verify that (say) $\epsilon(d).a.P$ is indeed an implementation (i.e. refines) $a_\diamond.P$, simply note that the relation

$$\mathcal{R} = \{(\epsilon(d).a.P, a_\diamond.P) \mid d > 0\} \cup \mathrm{Id}$$

is a refinement.

In order to support compositional verification we want the ability to combine Timed Modal Specifications with respect to the process constructs of TCCS, in particular with respect to that of parallel composition. Here we recall that the real–time calculus TCCS applies the *maximal progress* assumption: i.e. if a process is in a state in which it can perform internal (τ) computations then time is not allowed to pass. As an example consider the following combined Timed Modal Specification

$$(\overline{a}_\diamond.S \mid a_\diamond.T)\backslash a \tag{2}$$

Implementations of (2) should essentially be of the form $(P \mid Q)\backslash a$, where P implements $\overline{a}_\diamond.S$ and Q implements $a_\diamond.T$. From the discussion above we already know three typical implementations of a may–prefixed term. Hence, typical implementations[6] of (2) will be of the form

$$(\epsilon(d).\overline{a}.P' \mid \epsilon(d').a.Q')\backslash a \tag{3}$$

where P' and Q' implements S and T, respectively. Now, the operational semantics to be given to the combined Timed Modal Specification (2) should capture precisely these desired implementations. Choosing $d = d' = 0$ in (3), we obtain a desired implementation which — due to the maximal progress assumption — can perform nothing but a τ–transition (in particular it cannot delay). Thus, it is clear that $(\overline{a}_\diamond.S \mid a_\diamond.T)\backslash a$ should (at least) allow τ–transitions, whereas delay–transitions can not be required. In general, however, implementations of the form (3) are not immediately able to perform a τ–transition; rather a delay of $max\{d, d'\}$ time units must elapse. Hence, $(\overline{a}_\diamond.S \mid a_\diamond.T)\backslash a$ cannot insist on an immediate τ–transition, and should on the other hand allow implementations to delay. In summary, $(\overline{a}_\diamond.S \mid a_\diamond.T)\backslash a$ is given the following transitions

$$(\overline{a}_\diamond.S \mid a_\diamond.T)\backslash a \xrightarrow{\tau}_\diamond (S \mid T)\backslash a$$

$$(\overline{a}_\diamond.S \mid a_\diamond.T)\backslash a \xrightarrow{\epsilon(d)}_\diamond (\overline{a}_\diamond.S \mid a_\diamond.T)\backslash a$$

Now it is reasonable to expect that $(\overline{a}_\diamond.S \mid a_\diamond.T)\backslash a$ should be equivalent to the specification $\tau_\diamond.((S \mid T)\backslash a)$. This means that the semantics of specifications of the form $\tau_\diamond.S$ should be given by the following two axioms

$$\tau_\diamond.S \xrightarrow{\tau}_\diamond S \qquad\qquad \tau_\diamond.S \xrightarrow{\epsilon(d)}_\diamond \tau_\diamond.S$$

[6] besides nil.

$$\frac{-}{a_\diamond.S \xrightarrow{a}_\diamond S} \qquad \frac{-}{a_\Box.S \xrightarrow{a}_m S} \qquad \frac{S \xrightarrow{a}_m S'}{X \xrightarrow{a}_m S'}\ [X \overset{def}{=} S] \in \mathcal{E} \qquad \frac{S \xrightarrow{a}_m S'}{\epsilon(0).S \xrightarrow{a}_m S'}$$

$$\frac{S \xrightarrow{a}_m S'}{S\backslash A \xrightarrow{a}_m S'\backslash A}\ a, \overline{a} \notin A \qquad \frac{S \xrightarrow{a}_m S'}{S + T \xrightarrow{a}_m S'} \qquad \frac{T \xrightarrow{a}_m T'}{S + T \xrightarrow{a}_m T'}$$

$$\frac{S \xrightarrow{a}_m S'}{S \,|\, T \xrightarrow{a}_m S' \,|\, T} \qquad \frac{T \xrightarrow{a}_m T'}{S \,|\, T \xrightarrow{a}_m S \,|\, T'} \qquad \frac{S \xrightarrow{\alpha}_m S' \quad T \xrightarrow{\overline{\alpha}}_m T'}{S \,|\, T \xrightarrow{\tau}_m S' \,|\, T'}$$

Table 1: Action Rules for TMS ($m \in \{\Box, \diamond\}$).

3.2 Formal Syntax and Semantics

After the introductory discussion, we are now ready to formally present the syntax and semantics for TMS. As in CCS, we assume a set $\Lambda = \Delta \cup \bar{\Delta}$ with $\bar{\bar{\alpha}} = \alpha$ for all $\alpha \in \Lambda$, ranged over by α, β representing external actions, and a distinct symbol τ representing internal actions. We use $\mathcal{A}ct$ to denote the set $\Lambda \cup \{\tau\}$ ranged over by a, b representing both internal and external actions. Further, assume a set of process variables ranged over by X.

We adopt a two–phase syntax to describe networks of regular TMS. First, regular TMS expressions are generated by the following grammar

$$E ::= nil \mid \epsilon(d).E \mid a_\diamond.E \mid a_\Box.E \mid E + E \mid X$$

where X ranges over a finite set of variables $\mathcal{V}ar$ and d ranges over $\mathbf{R}_+$ (the positive reals). We shall assume that process variables are defined by a recursive equation system

$$\mathcal{E} = \{X \overset{def}{=} E_X \mid X \in \mathcal{V}ar\}$$

where all variables in E_X are guarded in the sense that each variable occurrence is within the scope of an action or delay prefix. Networks of regular TMS' are composite expressions of the form

$$(S_1 \mid \ldots \mid S_n)\backslash A$$

where S_i are regular TMS and $\mid$ and $\backslash A$ denote CCS parallel composition and restriction respectively. We will use S and T to range over (networks of regular) Timed Modal Specifications.

We now offer a modal transition semantics for TMS. This semantics is a conservative extension of the semantics for TCCS developed in [Wan90]. We present the transition rules in two groups: rules for actions in Table 1 and rules for delays in Table 2.[7] It should be rather clear from Table 1 and 2 that we indeed have defined a modal transition system, i.e. $\longrightarrow_\Box \subseteq \longrightarrow_\diamond$. As such, it can readily be seen that $\longrightarrow_\Box$–transitions may be derived for a combined specification ($S + T$ or $S \,|\, T$) only if $\longrightarrow_\Box$–transitions can be inferred for the contributing components. That is, in the derivation of transitions for a combined specification, the transitions contributed by the components should agree with respect to modality.

The side condition for the delay rule of parallel composition is to guarantee that (parallel composed) specifications satisfy the following two *maximal progress* assumptions

[7] In Table 2, we use d to stand for a non-zero real; this implies that an $\epsilon(0)$–transition can never be inferred by the inference rules. However, we shall apply the convention that $S \xrightarrow{\epsilon(0)}_m S$ for all S.

$$\frac{-}{nil \xrightarrow{\epsilon(d)}_m nil} \qquad \frac{-}{\alpha_n.S \xrightarrow{\epsilon(d)}_m \alpha_n.S} \qquad \frac{-}{\tau_\diamond.S \xrightarrow{\epsilon(d)}_\diamond \tau_\diamond.S}$$

$$\frac{-}{\epsilon(c+d).S \xrightarrow{\epsilon(d)}_m \epsilon(c).S} \qquad \frac{S \xrightarrow{\epsilon(d)}_m S'}{\epsilon(c).S \xrightarrow{\epsilon(c+d)}_m S'} \qquad \frac{S \xrightarrow{\epsilon(d)}_m S'}{S\backslash A \xrightarrow{\epsilon(d)}_m S'\backslash A}$$

$$\frac{S \xrightarrow{\epsilon(d)}_m S' \qquad T \xrightarrow{\epsilon(d)}_m T'}{S+T \xrightarrow{\epsilon(d)}_m S'+T'} \qquad \frac{S \xrightarrow{\epsilon(d)}_m S'}{X \xrightarrow{\epsilon(d)}_m S'}\ [X \stackrel{def}{=} S] \in \mathcal{E}$$

$$\frac{S \xrightarrow{\epsilon(d)}_m S' \qquad T \xrightarrow{\epsilon(d)}_m T'}{S\,|\,T \xrightarrow{\epsilon(d)}_m S'\,|\,T'}\ \mathcal{S}ort_{m^c}(d,S) \cap \overline{\mathcal{S}ort_{m^c}(d,T)} = \emptyset$$

Table 2: Delay Rules for TMS ($m, n \in \{\Box, \Diamond\}$ and $\Box^c = \Diamond$, $\Diamond^c = \Box$).

- *a timed specification will not allow delays if it requires an internal action τ.*

- *a timed specification will not require delays if it allows an internal action τ.*

For timed processes (i.e. timed specifications with all transitions being required) these two assumptions coincide and reduce exactly to the notion of maximal progress for processes. The two conditions above are formalized by means of two functions $\mathcal{S}ort_\Box$ and $\mathcal{S}ort_\Diamond$ defined (inductively) in Table 3. Intuitively, given a positive real d and a timed modal specification S, $\mathcal{S}ort_\Box(d,S)$ ($\mathcal{S}ort_\Diamond(d,S)$) includes all external actions that S requires (allows) an implementation to enable within d time units; hence the side condition $\mathcal{S}ort_\Box(d,S) \cap \overline{\mathcal{S}ort_\Box(d,T)} = \emptyset$ ($\mathcal{S}ort_\Diamond(d,S) \cap \overline{\mathcal{S}ort_\Diamond(d,T)} = \emptyset$) means that implementations of S and T will not necessarily (can not possibly) be able to communicate with each other within d time units.

Definition 3.1 *Given a timed modal specification S, we define $\mathcal{S}ort_\Box(0,S) = \mathcal{S}ort_\Diamond(0,S) = \emptyset$ and $\mathcal{S}ort_\Box(c,S)$ and $\mathcal{S}ort_\Diamond(c,S)$ for $c \neq 0$ to be the least sets satisfying the equations[8] given in Table 3.*

The following properties of timed specifications are obvious generalizations of central properties of timed processes in [Wan90]

Proposition 3.2

1. *(Maximal progress) If $S \xrightarrow{\tau}_m S'$ for some S', then $S \xrightarrow{\epsilon(d)}_{m^c} S''$ for no d and S''.*
2. *(Time determinism) Whenever $S \xrightarrow{\epsilon(d)}_m S'$ and $S \xrightarrow{\epsilon(d)}_m S''$ then $S' = S''$.*
3. *(Persistence) If $S \xrightarrow{\epsilon(d)}_m S'$ and $S \xrightarrow{\alpha}_m T$ for some S' and T, then $S' \xrightarrow{\alpha}_m T'$ for some T'.*
4. *(Time continuity) For all c, d and S'', $S \xrightarrow{\epsilon(c+d)}_m S''$ iff $S \xrightarrow{\epsilon(c)}_m S' \xrightarrow{\epsilon(d)}_m S''$ for some S'.*
5. *(Transition liveness) Either $S \xrightarrow{\tau}_m S'$ for some S' or $S \xrightarrow{\epsilon(d)}_{m^c} S''$ for some S'' and $d > 0$.*

[8] In Table 3, $c \dot{-} d$ is defined to be $c - d$ if $c > d$, 0 otherwise.

$$
\begin{array}{rcl}
\mathcal{S}ort_m(d, nil) & = & \emptyset \\
\mathcal{S}ort_\Box(d, \alpha_\Box.S) & = & \{\alpha\} \\
\mathcal{S}ort_\Box(d, \alpha_\Diamond.S) & = & \emptyset \\
\mathcal{S}ort_m(d, \tau_n.S) & = & \emptyset \\
\mathcal{S}ort_\Diamond(d, \alpha_m.S) & = & \{\alpha\} \\
\mathcal{S}ort_m(d, \epsilon(c).S) & = & \mathcal{S}ort_m(d \dot{-} c, S) \\
\mathcal{S}ort_m(d, S+T) & = & \mathcal{S}ort_m(d, S) \cup \mathcal{S}ort_m(d, T) \\
\mathcal{S}ort_m(d, S \mid T) & = & \mathcal{S}ort_m(d, S) \cup \mathcal{S}ort_m(d, T) \\
\mathcal{S}ort_m(d, S \backslash A) & = & \mathcal{S}ort_m(d, S) \setminus (A \cup \overline{A}) \\
\mathcal{S}ort_m(d, X) & = & \mathcal{S}ort_m(d, S), \;\; [X \stackrel{def}{=} S] \in \mathcal{E}
\end{array}
$$

Table 3: Definition of $\mathcal{S}ort_\Box$ and $\mathcal{S}ort_\Diamond$ $(m, n \in \{\Box, \Diamond\})$.

4 Abstracting Refinements

As already mentioned, TMS together with the two modal transition relations $\longrightarrow_\Box$ and $\longrightarrow_\Diamond$ defined in Table 1 and 2 constitutes a modal transition system. As such, we may readily apply the general notion of refinement from Definition 2.2 to TMS.

However, this refinement will often be too strong in practical applications. In particular, a refinement based directly on $\longrightarrow_\Box$ and $\longrightarrow_\Diamond$ will be completely sensitive to internal computation. In contrast, practical applications often need to abstract away from internal computation of systems. Also, when reasoning about large combined real–time systems, explicit timing information may in a first analysis be unimportant, in which case a time–abstracting refinement will suffice. Though such a refinement yields no information about the timing behaviour of the overall system, it will demand proper interaction between the timing properties of the components of the system.

Abstracting refinements with the above properties will be obtained through the definition of similarly abstracting versions of the modal transition relations $\longrightarrow_\Box$ and $\longrightarrow_\Diamond$. The abstracting transition relations will in all cases be generated by an abstraction function Φ on labels.[9] Now, recall that the modal transitions for TMS are labeled by elements of the following set

$$\mathcal{L} = \mathcal{A}ct \cup \mathcal{D}elay$$

where

$$\mathcal{D}elay = \{\epsilon(d) \mid d > 0\}$$

An abstraction function Φ maps sequences of (concrete) labels into a single (abstract) label. More

[9]This is strongly inspired by the notion of observation criterion in AUTO [SV89].

precisely, an abstraction function Φ is a *partial* function of the following type

$$\Phi : \mathcal{L}^* \hookrightarrow \mathcal{L} \cup \{\varepsilon\}$$

The partiality of Φ indicates that not all sequences of (concrete) labels makes sense as abstract actions. Also, $\Phi(s) = \varepsilon$ signifies that s is unobservable when viewed through the abstraction given by Φ.

Given an abstraction function we can now define abstracting transition relations.

Definition 4.1 *Let Φ be an abstraction function. Then the abstracting transitions relations $\longrightarrow_\Box^\Phi$ and $\longrightarrow_\Diamond^\Phi$ are defined as (m ranges over $\Box$ and $\Diamond$)*

- $S \overset{\sigma}{\longrightarrow}{}_m^\Phi S'$ *whenever* $S \overset{\mu_1}{\longrightarrow}_m \cdots \overset{\mu_n}{\longrightarrow}_m S'$ *with* $\Phi(\mu_1 \ldots \mu_n) \simeq \sigma$ *for some* $\mu_1, \ldots, \mu_n, \sigma$. [10]

Now it is easy to verify that TMS equipped with any abstracting transition relations $\longrightarrow_\Box^\Phi$ and $\longrightarrow_\Diamond^\Phi$ does indeed constitute a modal transition system in the sense of Definition 2.1. Hence, we may apply the notion of refinement from Definition 2.2

Notation 4.2 *If S refines T with respect to the Φ–abstracting transition relations $\longrightarrow_\Box^\Phi$ and $\longrightarrow_\Diamond^\Phi$, we say that S refines T with respect to the abstraction function Φ.*

We now present the abstraction functions which will induce the desired τ– and time–abstracting refinements.

Definition 4.3 *The τ–abstracting function Φ_τ is defined as follows*

$$\begin{aligned}
&\Phi_\tau(\tau^k) = \varepsilon\,; \;\; k \geq 0 \\
&\Phi_\tau(\tau^{k_0}\epsilon(d_1)\tau^{k_1}\ldots\epsilon(d_n)\tau^{k_n}) = \epsilon(d_1 + \ldots + d_n)\,; \;\; k_j \geq 0 \\
&\Phi_\tau(\tau^k \alpha \tau^j) = \alpha\,; \;\; k, j \geq 0
\end{aligned}$$

Whenever S refines T with respect to Φ_τ we say that S weakly refines T. We write $S \trianglelefteq T$ in this case.

In the following we shall use also a more standard process algebraic notation $S \overset{\mu}{\Longrightarrow}_m S'$ for $S \overset{\mu}{\longrightarrow}{}_m^{\Phi_\tau} S'$ for any admissible label $\mu \in \{\varepsilon\} \cup \Lambda \cup \mathcal{D}elay$.

Following the proofs for timed equivalence in [Wan90] it can be shown ([God94]) that the *non–abstracting refinement* $\triangleleft$ is preserved by all constructs of TMS. For the weak refinement $\trianglelefteq$ a certain very natural syntactic conditions on specifications can be defined (see Section 6.1) which ensure also it to be preserved by all TMS constructs except summation (as usual for τ-abstracted process algebraic constructs). We study the (general lack of) compositionality of $\trianglelefteq$ in more detail in Section 6, where we also propose a new τ–abstracting refinement that is preserved by parallel composition.

The two following functions abstract from time (and internal computation)

Definition 4.4 *The time–abstracting function Φ_ϵ is defined as follows*

$$\begin{aligned}
&\Phi_\epsilon(s) = \varepsilon\,; \;\; s \in \mathcal{D}elay^* \\
&\Phi_\epsilon(s_1 a s_2) = a\,; \;\; s_1, s_2 \in \mathcal{D}elay^*
\end{aligned}$$

Whenever S refines T with respect to Φ_ϵ we say that S is a time–abstracted refinement of T. We write $S \overset{\bullet}{\triangleleft} T$ in this case.

[10] For expressions e_1 and e_2, $e_1 \simeq e_2$ holds if both expressions are defined and have the same value.

Definition 4.5 *The τ– and time–abstracting function $\Phi_{\tau\epsilon}$ is defined as follows*

$$\begin{array}{l}\Phi_{\tau\epsilon}(s) = \varepsilon\,;\;\; s \in (\mathcal{D}elay \cup \{\tau\})^* \\ \Phi_{\tau\epsilon}(s_1 \alpha s_2) = \alpha\,;\;\; s_1, s_2 \in (\mathcal{D}elay \cup \{\tau\})^*\end{array}$$

Whenever S refines T with respect to $\Phi_{\tau\epsilon}$ we say that S is a weak time–abstracted refinement of T. We write $S \stackrel{\bullet}{\trianglelefteq} T$ in this case.

The two time–abstracting refinements are *not* preserved by the constructs of TMS (in particularly not parallel composition). However, the full abstractness result for time abstracted equivalences proved in [LW90] extends to $\stackrel{\bullet}{\lhd}$. That is, the largest pre–order contained in $\stackrel{\bullet}{\lhd}$ which is also preserved by parallel composition will be $\lhd$. The proof can be found in [God94].

$$\begin{tikzcd} & \stackrel{\bullet}{\trianglelefteq} & \\ \stackrel{\bullet}{\lhd} \arrow[ur] & & \trianglelefteq \arrow[ul] \\ & \lhd \arrow[ul] \arrow[ur] & \end{tikzcd}$$

Figure 1: Ordering refinements.

In Figure 1 we illustrate the relationship between the four refinements introduced. The arrows between, $\lhd$ and $\stackrel{\bullet}{\lhd}$ say, represents the set inclusion $\lhd \subseteq \stackrel{\bullet}{\lhd}$. The proof of these inclusions, that they are strict and also the only inclusions among the four refinements are straightforward. Also, it is easy to prove that the refinements, when restricted to timed processes, coincide with the corresponding equivalences studied at length in [LW90].

Example 4.6 *Recall the combined media specification (1) from the introduction*

$$(M_d^{a,c} \mid S_{e-d,f-d}^{c,b}) \backslash c$$

where [11]

$$\begin{array}{rcl} M_x^{a,b} & \stackrel{def}{=} & a_\Box.\epsilon(x).\overline{b}_\Box. \\ S_{x,y}^{a,b} & \stackrel{def}{=} & a_\Box.(\epsilon(x).\overline{b}_\Diamond. + \epsilon(y).\overline{b}_\Box.) \end{array}$$

Then the following abstracting refinements may be deduced

$$\begin{array}{rcl} (M_d^{a,c} \mid S_{e-d,f-d}^{c,b}) \backslash c & \trianglelefteq & S_{e,f}^{a,b} \\ (M_d^{a,c} \mid S_{e-d,f-d}^{c,b}) \backslash c & \stackrel{\bullet}{\trianglelefteq} & S_{0,0}^{a,b} \end{array}$$

i.e. the combined media weakly refines a media with delay between e and f and is a weak time–abstracted refinement of a medium which enables its output immediately after reception of its input.

[11] We omit trailing *nil*'s; i.e. a_m abbreviates $a_m.nil$.

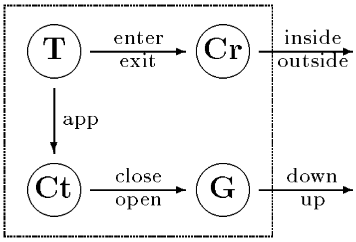

Figure 2: The Train Crossing.

The verification tool EPSILON supports automatic verification for all four types of refinements we have considered so far. In fact, all instances of the weak and time–abstracted refinements in the above example can be automatically checked using EPSILON. To be precise, the tool EPSILON supports automatic refinement checking for a slightly restricted class of TMS, namely, for those specifications which are *initially integral timed* (also called simply *integral timed* and abbreviated IT) according to the following definition.

Definition 4.7 *S is initially integral timed iff every delay prefix $\epsilon(d)$ occurring in S has the delay $d \in \boldsymbol{N}$.*

Actually, the algorithms would apply also for specifications with delays being positive rationals, as we may simply multiply all delays with some rational q in order to end up with a comparison of initially integral processes. It should also be noted that an IT TMS specification still can perform delays of arbitrary *real* length[12], as well as all instances of TMS specifications naturally appearing in the considered Train Crossing example (see Section 5) are initially integral timed.

In the next section of this paper, we shall demonstrate the tools of EPSILON on a somewhat larger and more complex example.

5 The Train Crossing

In this section we demonstrate the applicability of the TMS theory and the tool EPSILON to a small example of a train crossing. Similar examples can be found elsewhere in the literature, e.g. [AD91].

The Train Crossing (see Figure 2) is a small idealized example of a real world train crossing. It consists of four components: the crossing (Cr), a train (T), the gate (G) and the controller (Ct). When a train approaches the crossing it sends a signal to the controller. Having received the signal, after some delay the controller starts closing the gate. Then, after some more delay the controller starts opening the gate. The train is assumed to have moved through the crossing while the gate was closed.

The external events of our model system will be `down` and `up` meant to occur when the gate is becoming closed or open, as well as `inside` and `outside` representing the moments of the train actually entering and leaving the crossing.

The TMS specification of the crossing is given in Figure 3 by `TrainCrossing(X,A,B,C,U,V)`. It consists of four parallel components, namely the *crossing*, the *gate*, a *train* and the *controller*. We have made

[12]For instance, $\epsilon(1).a_{\Box}.nil \stackrel{\epsilon(0.34)}{\longrightarrow} \epsilon(0.66).a_{\Box}.nil$.

```
Crossing :=:  in(enter);inside!Crossing_With_Train
Crossing_With_Train :=:  in(exit);outside!Crossing

Gate(X) :=:  in(close);ϵ[0,X].tau;down!Gate_Closed(X)
Gate_Closed(X) :=:  in(open);ϵ[0,X].tau;up!Gate(X))

Train(A,B,C) :=:  out(app)?Train_To_Enter(A,B,C)
Train_To_Enter(A,B,C) :=:  ϵ[A,B].out(enter);Train_In(A,B,C)
Train_In(A,B,C) :=:  ϵ[0,C].out(exit);Train(A,B,C)

Controller(U,V) :=:  in(app);Contr_Close(U,V)
Contr_Close(U,V) :=:  ϵ[0,U].out(close);Contr_Open(U,V)
Contr_Open(U,V) :=:  ϵ(V);out(open);Controller(U,V)

TrainCrossing(X,A,B,C,U,V) :=:
(Crossing/Gate(X)/Train(A,B,C)/Controller(U,V))\[enter,exit,close,open,app]
```

Figure 3: Specification of the train crossing in EPSILON.

the specification explicitly dependent on a number of time parameters to illustrate the effect of their modification on the properties of the specified system.

In the specification we have introduced some (possibly generally useful) derived language constructs, namely,

- $\epsilon[d,e].a.S$ is a shorthand for $\epsilon(d).a_\diamond.S + \epsilon(e).a_\Box.S$ (it means that the transition with the action a, leading to S *may* be enabled after d time units, but it *must* be enabled after e time units (usually $d < e$)), and
- $a!S$ abbreviates $T \stackrel{def}{=} a_\Box.S + \tau_\Box.T$ (it is a kind of "time-lock" operation: the $\tau_\Box$–loop around T ensures that the time is not allowed to pass due to the maximal progress assumption, so the only behaviour what such a specification allows is doing the a *immedilately*. This is the way, how immediate (urgent) actions are modelled in TMS.).

In EPSILON `:=:` is used for declarations binding the left hand side identifier to the right hand side, `;` denotes the must modality and `?` the may modality. `in(a)` and `out(a)` are used to represent an action and its corresponding co–action and restriction from actions is defined by `\L` where `L` is a list of actions.

The first of the four components, the *Crossing* is simply keeping track of whether there is a train in it, or not, and at any time when a train either enters or leaves, it gives an immediate signal `inside` or `outside` to the external observer.

The *Gate* is either open or closed. It is receiving signals `open` and `close` from the controller. After receiving a signal, say `close`, it takes for the gate some time in the range of $[0, X]$ to actually become closed. When the gate becomes closed (open), a corresponding external output signal (`down` or `up`) is signalled immedilately.

```
Spec1 :=:  down?inside?outside?up?Spec1

Spec2(D) :=:  Spec2a(D)/Admit_Urgency
Spec2a(D) :=:  down?ϵ(D);inside?outside?up?Spec2a(D)
Spec3(M,N) :=:  DownUp(M,N)/Uni([inside,outside])
Spec4(M,N) :=:  DownUp(M,N)/InOut/Admit_Urgency
DownUp(M,N) :=:  down?ϵ[M,N].up;DownUp(M,N)

InOut :=:  inside?outside?InOut
Admit_Urgency :=:  tau?Admit_Urgency

Spec5(P) :=:  Down(P)/Uni([inside,outside,up])
Down(P) :=:  down?P;Down(P)
```

Figure 4: Specifications.

The *Train* initially may send a signal to the controller about its approaching[13]. The train is then supposed to enter the crossing within the interval from `A` to `B`. Further on, it will necessarily leave the crossing no later than `C` time units after it entered.

In the initial state the *Controller* waits for the approaching of a train. If a train approaches he starts closing the gate no later than `U` time units after the approaching was signalled. Then he waits for `V` time units before opening the gate.

Figure 4 contains a few properties (specifications) against which the considered train crossing model can be analyzed.

First, a natural safety property for the train crossing to satisfy would be the occurrence of its external events in the order, as prescribed by `Spec1`. We express this fact in the theory by the weak time abstracted refinement between `TrainCrossing(X,A,B,C,U,V)` and `Spec1`. Using EPSILON it can be found out that, for instance,

$$\texttt{TrainCrossing(1,3,4,1,1,6)} \overset{\bullet}{\trianglelefteq} \texttt{Spec1}$$

Actually, `TrainCrossing(X,A,B,C,U,V)` will be a weak time abstracted refinement of `Spec1` whenever `U + X < A` and `B + C < V` (and for any particular values of the time parameters the fact of the refinement can be established by EPSILON; in fact, alongside with a symbolic description of the *contents* of the refinement, see Section 7). It is to be observed also that, though the specification `Spec1` is not explicitly mentioning time quantities at all, the correctness of the train crossing model against this specification is crucially dependent on the time quantities put in the description of various components of the model (intuitively, the internal timing properties of the model are precluding some *order* of the external events by requiring that some component is going always to produce its output *faster* than the other).

However, not all important properties of real time systems can be described solely in terms of the *ordering* of the system external event occurrences. In the case of the train crossing it might be very important to

[13] Note, that we do not require a train to approach the crossing. If no train will approach the crossing the whole system is inactive. In the theory this is reflected by *nil* $\triangleleft$ `TrainCrossing(X,A,B,C,U,V)` for any values of `X`, `A`, `B`, `C`, `U` and `V`.

require that always there will be a certain delay of, say, D time units between the gate becoming closed and the following moment when the train enters the crossing[14]. We express this fact in the TMS theory by TrainCrossing(X,A,B,C,U,V) being a weak timed refinement of Spec2(D), and we may find out (either reasoning theoretically, or just by applying EPSILON) that

$$\texttt{TrainCrossing(1,3,4,1,1,6)} \trianglelefteq \texttt{Spec2(1)}, \text{ but } \texttt{TrainCrossing(1,3,4,1,1,6)} \not\trianglelefteq \texttt{Spec2(2)}.$$

And indeed, there can be implementations of TrainCrossing(1,3,4,1,1,6) which do let less that 2 time units between the events down and inside[15].

Similarly, we could ask about the relationship between the time moments of the closing and the opening of the gate. For that purpose we define Spec3(M,N). Intuitively, Spec3(M,N) specifies that the opening of the gate is guaranteed to occur in the interval from M to N after it was lowered. Here we use another specification shorthand, namely, the very loose specification

$$Uni(L) \stackrel{def}{=} \prod_{a \in L \cup \{\tau\}} a_\diamond.Uni(L)$$

where Π denotes an n–ary parallel composition and $L \subseteq \mathcal{A}ct$. $Uni(L)$ is a "universal specification" in the sense that $S \lhd Uni(L)$ for any timed modal specification S with sort contained in L [16].

Whenever $\texttt{M} \leq 5$ and $\texttt{N} \geq 7$ it turns out, applying EPSILON, that

$$\texttt{TrainCrossing(1,3,4,1,1,6)} \trianglelefteq \texttt{Spec3(M,N)} \tag{4}$$

Hence, due to the strongest of these specifications, Spec3(5,7), the gate must be opened no later than 7 time units after it was closed. Moreover, it is impossible to tigthen the interval between the opening and the lowering of the gate, e.g. Spec3(5,6) is shown by EPSILON not be weak refined by TrainCrossing(1,3,4,1,1,6).

Actually, for the values of M and N mentioned above, we can prove that TrainCrossing(1,3,4,1,1,6) is a weak refinement of the even stronger property Spec4(M,N), that is, compared to Spec3(M,N) we furthermore require a specific ordering of the external events inside and outside.

Under the assumption that a proof of

$$\texttt{TrainCrossing(1,3,4,1,1,6)} \trianglelefteq \texttt{Spec4(M,N)}$$

had already been given, a direct proof of (4) would not be needed. As $\lhd$ is preserved by parallel composition, we can obtain the result in an alternative manner exploiting the compositionality. We prove that

[14] In case if there is very little time between these two events, think of a car which has entered the crossing just before the gate was closed, and has broken there. If there were enough time, it would be at least possible for people to leave the car, even better, if the car could be taken out mechanically, or the train could be stopped.

[15] A note is to be added about the specification component Admit_Urgency. When we are given a specification like $a_\diamond.S$ (or $a_\Box$, for that matter), it does not admit the implementation $a!S$, nor does it admit $\epsilon(d).a!S + a_\Box.S$ for any $d \geq 0$. This is because the specification is requiring (unlimited) delay ability from all its implementations (see the TMS delay semantics description in Table 2). The component Admit_Urgency, when added to the specification, contributes by discarding the delay ability requirement by the specification both in its initial and in any of its (operational) derivative states (Admit_Urgency only *allows* delays, without requiring them). As our example does contain immediate (urgent) actions (in fact, we have made all our external actions urgent), it can refine only specifications which does admit them. It is clear, however, that as a specification component the Admit_Urgency is harmless since allowing the implementations to have immediate actions is the only effect which it has.

[16] Thus $Uni(L)$ is the weakest specification with sort L. It does also admit urgent actions, in fact $\texttt{Admit\_Urgency} = Uni(\emptyset)$.

```
FastContr :=:  in(app);out(close);6;out(open);FastContr
SlowContr :=:  in(app);1;out(close);6;out(open);SlowContr
```

Figure 5: Implementations.

$$\texttt{InOut/Admit\_Urgency} \lhd \texttt{Uni([inside,outside])}$$

(which obviously holds) to immediately conclude Spec4(M,N) $\unlhd$ Spec3(M,N). The rest of the proof is due to the transitivity of $\unlhd$.

Another interesting property is the frequency of the lowering of the gate. More precisely, we want to determine the values of P for which TrainCrossing(1,3,4,1,1,6) is a weak refinement of Spec5(P). Intuitively, Spec5(P) specifies that the frequency between two consecutive closing of the gate must be at least P time units. At a first glance one would expect the frequency to be at least 6 time units because the controller must wait exactly 6 time units between initializing the lowering and opening of the gate. However, using EPSILON we can find that

$$\texttt{TrainCrossing(1,3,4,1,1,6)} \not\unlhd \texttt{Spec5(6)}$$

The reason for this is that it may take up to one time unit for the gate to close and later there is a possibility to open and afterwards close again immediately without performing any delay.

Instead, whenever P ≤ 5, we have

$$\texttt{TrainCrossing(1,3,4,1,1,6)} \quad \unlhd \quad \texttt{Spec5(P)} \tag{5}$$

We can either prove (5) directly in EPSILON or alternatively, for any P ≤ 5, we could prove instead that

$$\texttt{Spec3(P,7)} \quad \unlhd \quad \texttt{Spec5(P)} \tag{6}$$

and then take advantage of the transitivity of $\unlhd$. Due to compositionality, (6) holds since

$$\texttt{DownUp(P,7)} \quad \unlhd \quad \texttt{Down(P)/Uni([up])}$$

for any P ≤ 5.

Implementations of the loosely specified train crossing of Figure 3 may now be found simply by substituting each of the four components with some timed process (strongly or weakly) refining the component. Due to the partiality (looseness) of the specification of the components[17] each component will have several inequivalent implementations. For instance, as implementation of the controller one could choose one of the processes in Figure 5. Clearly FastContr and SlowContr are inequivalent and obviously both FastContr and SlowContr refines Controller(1,6).

Finally, let us emphasize the generality of correctness proofs carried out within the framework of TMS. Indeed, no matter which implementation of the Train Crossing we will decide upon, it will be ensured, as a consequence of the compositionality of verification in TMS, that any of those is guaranteed to satisfy all the properties above refined by TrainCrossing(1,3,4,1,1,6).

[17] Except for the Crossing of course.

6 Achieving Congruicity

In this section we examine the general lack of compositionality of the weak refinement $\trianglelefteq$ in more detail, and show two possible ways of coping with this deficiency. The first possibility is obtained through a syntactic restriction and the second possibility consists of a redefinition of weak refinement. It should be noted although that for most practical examples we belive that the weak refinement *is* preserved by parallel composition.

To see that $\trianglelefteq$ can in general not be preserved by parallel composition, consider the following example:

Let C be defined by

$$C \stackrel{def}{=} \tau_\Box.C + \tau_\Diamond.(\overline{b}_\Box + \tau_\Diamond.(\overline{a}_\Box + \epsilon(1).c_\Box))$$

Then $nil \trianglelefteq (C \,|\, \epsilon(1).a_\Box)\backslash a$ since

$$\begin{aligned}
&\{(nil, (C \,|\, \epsilon(1).a_\Box)\backslash a)\} \\
&\quad \cup\{(nil, ((\overline{a}_\Box + \epsilon(1).c_\Box) \,|\, \epsilon(d).a_\Box)\backslash a) \mid d < 1\} \\
&\quad \cup\{(nil, ((\overline{a}_\Box + \epsilon(e).c_\Box) \,|\, \epsilon(d).a_\Box)\backslash a) \mid d < e < 1\} \\
&\quad \cup\{(nil, ((\overline{a}_\Box + \epsilon(d).c_\Box) \,|\, a_\Box)\backslash a) \mid d < 1\} \\
&\quad \cup\{(nil, (nil \,|\, nil)\backslash a)\}
\end{aligned}$$

constitutes a refinement wrt. Φ_τ. Now, let $S \stackrel{def}{=} nil \,|\, b_\Box$ and let $T \stackrel{def}{=} ((C \,|\, \epsilon(1).a_\Box)\backslash a) \,|\, b_\Box$. [18] Then one may prove that $S \ntrianglelefteq T$. Intuitively, $S \ntrianglelefteq T$ because S allows a delay, say half a time unit, and this delay can only be matched by T in such a way that T cannot allow b or such that T after yet another half time unit requires c.

6.1 Syntactic Conditions

Following we present a syntactic condition under which the weak refinement $\trianglelefteq$ is preserved by parallel composition. It shall be noted, however, that the condition is far from being also *necessary*. Finding further even less restrictive conditions has not been included in the scope of this paper, as we believe that the work on those can benefit from the further case studies using TMS.

We first define the set of actions $Sort_m$ as follows:

Definition 6.1 *Define for any regular S the set $Sort_m \subseteq \Lambda$ as the least set satisfying the equations in Table 4.*

[18] Strictly speaking T is not a network according to the definition of networks in Section 3.2. However, it is immediate that T can easily be transformed to the TMS network $((C \,|\, \epsilon(1).a_\Box) \,|\, b_\Box)\backslash a$.

$$Sort_m(nil) = \emptyset$$
$$Sort_m(\epsilon(d).S) = Sort_m(S)$$
$$Sort_m(a_m.S) = \{a\} \cup Sort_m(S)$$
$$Sort_m(\tau_m.S) = Sort_m(S)$$
$$Sort_m(\mu_{m^c}.S) = Sort_m(S)$$
$$Sort_m(S+T) = Sort_m(S) \cup \mathcal{S}ort_m(d,T)$$
$$Sort_m(C) = Sort_m(S_C),\ C \stackrel{def}{=} S_C$$

Table 4: Definition of $Sort_m(S)$.

Then the restriction we impose on a network $S = (S_1 \mid \ldots \mid S_n)\backslash A$ is that S must not contain $\tau_\diamond$ and for all S_i

$$\alpha \in Sort_\diamond(S_i) \text{ implies} \\ \quad \forall j \neq i.\, \overline{\alpha} \notin Sort_\diamond(S_j) \cup Sort_\Box(S_j)$$

The restriction on networks implies that any network satisfying the restriction cannot allow without also requiring an internal transition.

It can be proven that for networks satisfying the syntactic restriction $\trianglelefteq$ preserves parallel composition.

6.2 Trajectory Step Refinement

In this section we define an alternative τ-abstracted refinement relation for TMS, which proves to be semantically "better behaved" than $\trianglelefteq$ (though not as elegantly definable).

First, given two TMS T and T', and $d \in \boldsymbol{R}_{>0}$ let us call any sequence $(\langle T_0, d_0\rangle, \ldots, \langle T_n, d_n\rangle)$ such that

- $T \stackrel{\varepsilon}{\Longrightarrow}_\diamond T_0$, $d_0 = 0$, $T_n = T'$, $d_n = d$ and
- $T_i \stackrel{\epsilon(\delta_i)}{\longrightarrow}_\diamond \stackrel{\varepsilon}{\Longrightarrow}_\diamond T_{i+1}$, with $\delta_i = d_{i+1} - d_i$ for all $i = 0, 1, \ldots, n-1$

a *step sequence* for $\langle T, d, T'\rangle$ (or simply a $\langle T, d, T'\rangle$ - step sequence). Further, let $S^{(d)}$ for a TMS S and $d \geq 0$ denote the TMS S' for which $S \stackrel{\epsilon(d)}{\longrightarrow}_\diamond S'$, if such S' exists (due to the time determinacy property (see Proposition 3.2) such S', if it exists, is unique).

Definition 6.2 *A binary relation $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ is a trajectory step refinement if $\langle S, T\rangle \in \mathcal{R}$ implies*

- $S \stackrel{a}{\longrightarrow}_\diamond S'$ *implies* $\exists T'.\, T \stackrel{a}{\Longrightarrow}_\diamond T'$ *and* $\langle S', T'\rangle \in \mathcal{R}$,
- $T \stackrel{a}{\longrightarrow}_\Box T'$ *implies* $\exists S'.\, S \stackrel{a}{\Longrightarrow}_\Box S'$ *and* $\langle S', T'\rangle \in \mathcal{R}$,
- $S \stackrel{\epsilon(d)}{\longrightarrow}_\diamond S'$ *implies the existence of a* $\langle T, d, T'\rangle$ *- step sequence* $(\langle T_0, d_0\rangle, \ldots, \langle T_n, d_n\rangle)$ *such that* $\langle S^{(d_i)}, T_i\rangle \in \mathcal{R}$ *for all* $i = 0, 1, \ldots, n$.

- $T \xrightarrow{\epsilon(d)}_{\Box} T'$ *implies* $\exists S'.\ S \xRightarrow{\epsilon(d)}_{\Box} S'$ *and* $\langle S', T' \rangle \in \mathcal{R}$.

We denote by $\lhd^T$ the largest trajectory step refinement. It is not difficult to establish that $\lhd^T$ is a preorder (i.e., $\lhd^T$ is reflexive and transitive) and that $\lhd^T \subseteq \unlhd$. Moreover, $\lhd^T$ is preserved by parallel composition (as well as all other TMS constructors except summation), the proof can be found in [God94].

Furthermore, one can prove rather easily that for TCCS processes (which are the implementations of the TMS specifications) the relation $\lhd^T$ coincides with the TCCS weak bisimulation[19] (as does also $\unlhd$).

So, TMS with the refinement relations $\lhd$ and $\lhd^T$ can be viewed as a conservative extension of Timed CCS with strong and weak bisimulation, being at the same time fully suitable for use in stepwise development process of real time systems.

Observe, though, that the definition of $\lhd^T$ does not follow the general abstracting refinement definition pattern used in Section 4.

As all other refinemnet relations considered in this paper, also $\lhd^T$ is decidable for (initially) integral timed TMS specifications. We outline the deciding algorithm in Section 7.5.

7 Algorithms for Refinement Checking

This section provides an outline of the algorithms for checking automatically whether two given (initially) integral timed modal specifications satisfy a given modal refinement relation (i.e. one of the relations $\lhd$, $\unlhd$, $\dot{\lhd}$, $\dot{\unlhd}$ and $\lhd^T$). These algorithms are the basis for the verification tool EPSILON, in which $\lhd$, $\unlhd$, $\dot{\lhd}$ and $\dot{\unlhd}$ have been implemented.

Since the definitions of the considered refinement relations depend essentially on exploiting infinite transition systems defining semantics of TMS (in fact, these transition systems are in a certain sense even "continuous", what is due to the density of the underlying time domain — non–negative reals), it is not possible to use them directly in the deciding algorithm. Instead, the algorithm uses a symbolic representation of TMS transition systems, based mainly on the *region graph technique* due to [AD90] and the following work in [Č92b] and [LW90] extending the region graph technique to work also for deciding timed and time abstracted *bisimulation equivalences* between timed processes.

The algorithms described in this section are theoretically rather direct generalizations of those described in [Č92b] and [LW90] for deciding corresponding (timed or time-abstracted) bisimulation equivalences, especially in the case of "timed" refinements (the strong and weak ones). The presentation of the refinement deciding algorithms in this section differs from the previous theoretical work mainly in

- being more concrete, with more detailed explanation of the used data structures, closely tied up with the basic design decisions implemented in the verification tool EPSILON;
- making use of explicitly compositional process algebraic style syntax of TMSs (this should be contrasted with the automata based model of Parallel Timer Processes for which the bisimulation equivalence problem was shown decidable in [Č92b]);
- providing uniform treatment in deciding time abstracted and time sensitive refinements.

[19] The proof relies on maximal progress and time determinism properties, which hold for TCCS (they hold also for TMS specifications).

Also a slightly novel point is the symbolic deciding procedure for trajectory step refinement in Section 7.5, what has not been considered before.

Yet for motivating the "correctness" of various parts of the developed algorithms we from time to time in the following refer to the basic theoretical papers [AD90], [Č92b] and [LW90].

7.1 Principal Schema of the Algorithms

Given two initially integral timed modal specifications S and T to be checked under some refinement relation $\mathtt{ref}$ (i.e. $\lhd$, $\overset{\bullet}{\lhd}$, $\unlhd$, $\overset{\bullet}{\unlhd}$ of $\lhd^T$), the algorithm for deciding whether S is refined by T (i.e. if $S \,\mathtt{ref}\, T$) can be thought as deciding some appropriate property over a certain *finite* symbolic transition system[20]

$$\mathcal{T}_{S,T} = \langle \mathcal{X}_{S,T}, \mathcal{L}^\bullet, \longrightarrow, X_0 \rangle, \text{ where}$$

- $\mathcal{X}_{S,T}$ is a set of *symbolic states* (each symbolic state $X \in \mathcal{X}_{S,T}$ is a representation of a certain (usually infinite) set of pairs $\langle S', T' \rangle$ where S' is a state of S and T' is state of T);
- $\mathcal{L}^\bullet$ is a set of symbolic *labels* (constructed from the actions $a \in \mathcal{A}ct$ and some special actions used for representing delays);
- $\longrightarrow \subseteq \mathcal{X}_{S,T} \times \mathcal{L}^\bullet \times \mathcal{X}_{S,T}$ is a symbolic transition relation; and
- $X_0 \in \mathcal{X}_{S,T}$ is the initial state.

The property to be checked over $\mathcal{T}_{S,T}$ in each case is expressed as the existence of a state set $\mathcal{X} \subseteq \mathcal{X}_{S,T}$ which is both containing X_0 and satisfying a certain closeness property, such as, for instance, *"for every $X \in \mathcal{X}$ whenever $X \xrightarrow{a} X'$ for $a \in \mathcal{L}^\bullet$, then $X' \xrightarrow{a'} X''$ for some $X'' \in \mathcal{X}$"*.

More precisely, for every refinement relation $\mathtt{ref}$ a certain monotone functional $\mathcal{F}^{\mathtt{ref}}$ on the powerset of $\mathcal{X}_{S,T}$ is defined ($\mathcal{F}^{\mathtt{ref}}$ is mapping subsets of $\mathcal{X}_{S,T}$ to subsets of $\mathcal{X}_{S,T}$), and the question to be answered about $\mathcal{T}_{S,T}$ (in order to decide, if $S \,\mathtt{ref}\, T$) is formulated as

"Does there exist a set $\mathcal{X} \subseteq \mathcal{X}_{S,T}$ such that $X_0 \in \mathcal{X}$ and $\mathcal{X} \subseteq \mathcal{F}^{\mathtt{ref}}(\mathcal{X})$?"

Equivalently, one can say that the algorithm is examining, whether X_0 is contained in the *greatest fixpoint* of $\mathcal{F}^{\mathtt{ref}}$.

Having a monotone functional $\mathcal{F}$ over the finite symbolic transition system $\mathcal{T}_{S,T}$ there are a variety of ways how to check, if a certain state X_0 of the system is contained in the greatest fixpoint of $\mathcal{F}$. The tool Epsilon is based on the (efficient) local checking technique due to [Lar92], thus in constructing a state set $\mathcal{X}$ such that $\mathcal{X} \in \mathcal{F}(\mathcal{X})$ we take initially $\mathcal{X}$ to contain only X_0 and add elements as needed in order to obtain a set closed under the functional $\mathcal{F}$. Though the local checking technique cannot be implemented without a limited amount of "backtracking" (since it is not possible to guess a priori which state should be included into $\mathcal{X}$, if there are alternatives), the "storage" of already examined unsuccessful alternatives, as advocated in [Lar92], allows to avoid unnecessary recomputations. This gives for the considered local checking technique a polynomial worst-case time complexity (in size of the symbolic transition system). However, the main advantage of this technique is that it needs to explore only the *reachable* part of the symbolic system, which in "good" cases is considerably smaller than the system in whole. For more details of the method we refer to [Lar92].

[20] In fact, the transition systems $\mathcal{T}_{S,T}$ designed for solving $S \,\mathtt{ref}\, T$ are different for different refinement relations.

7.2 Main Symbolic Structures

In this section we define the main symbolic structures used in the refinement deciding algorithms, relevant for both deciding time abstracted and time sensitive refinements. After that the refinement deciding algorithms themselves (together with a few additional necessary constructs) are considered in Section 7.3 and Section 7.4 for time abstracted and time sensitive ($\lhd$ and $\unlhd$) refinements, respectively. The algorithm for deciding the trajectory step refinement $\lhd^T$ is given in Section 7.5.

7.2.1 Representation of Regular TMS

We define the set of modal action prefixes $\mathcal{A}ct^m = \{a_\Diamond, a_\Box | a \in \mathcal{A}ct\}$ and let $a_\bullet$ possibly with some upper indices range over $\mathcal{A}ct^m$.

We say that a RTMS (= regular TMS) S_1 is in *Normal Expression Form* ("NEF", for short), if it is defined by an equation system

$$
\begin{aligned}
S_1 &\stackrel{def}{=} \epsilon(d^{11}).a_\bullet^{11}.S_{11} + \ldots + \epsilon(d^{1k_1}).a_\bullet^{1k_1}.S_{1k_1}, \\
\ldots &\quad \ldots \quad \ldots \\
S_r &\stackrel{def}{=} \epsilon(d^{r1}).a_\bullet^{r1}.S_{r1} + \ldots + \epsilon(d^{rk_r}).a_\bullet^{rk_r}.S_{rk_r},
\end{aligned}
$$

where for every $i \le r$ and $j \le k_i$ we have $d^{ij} \in \boldsymbol{N}$ (possibly $d^{ij} = 0$), $a_\bullet^{ij} \in \mathcal{A}ct^m$ and $S_{ij} = S_u$ for some $u \le r$ (i.e. every S_{ij} appears as a left hand side of one of the expressions above).

In the following description of the refinement relation deciding algorithms we assume that every RTMS contained as a component in the given specifications S or T is given in a NEF. This assumption may be made without any loss of generality as any RTMS may be turned into NEF by rather straightforward equivalence preserving transformations.

For T being a RTMS, defined by the NEF equation

$$T \stackrel{def}{=} \epsilon(d^1).a_\bullet^1.T_1 + \ldots + \epsilon(d^k).a_\bullet^k.T_k$$

and $d \ge 0$ being a non–negative *real* number we define a d-shift of T to be[21]

$$T^{(d)} = \epsilon(d^1 \dot{-} d).a_\bullet^1.T_1 + \ldots + \epsilon(d^k \dot{-} d).a_\bullet^k.T_k.$$

Observe that, if $T \xrightarrow{\epsilon(d)}_m T'$ for some T', then $T' = T^{(d)}$[22].

Let us further on denote the set of all initially integral timed regular TMS (RTMS) by $\mathcal{I}$. Moreover, whenever a RTMS T can be obtained from some IT RTMS $T_0 \in \mathcal{I}$ just by some time-shift $T = T_0^{(d)}$, where $d \in \boldsymbol{R}_{+0}$, we call it *well timed* (and sometimes abbreviate this as w.t.). So,

$$T = \epsilon(2.63).a_\Box.T_1 + \epsilon(4.63).b_\Diamond.T_2 + c_\Box.T_3$$

is well timed (take, for instance,

[21] For x, y being nonnegative reals, $x \dot{-} y = max\{x - y, 0\}$.

[22] However, the shift operation is defined purely syntactically, and it does assert nothing about the actual delay possibility $T \xrightarrow{\epsilon(d)}_m T'$ for some T'. This delay can be impossible due to the *maximal progress* requirements, for instance, when $T = \epsilon(0).\tau_\Box.S + \epsilon(1).a_\bullet.S'$.

$$T_0 = \epsilon(3).a_{\Box}.T_1 + \epsilon(5).b_{\Diamond}.T_2 + c_{\Box}.T_3 \text{ and } d = 0.37),$$

while $S = \epsilon(2.63).a_{\Box}.T_1 + \epsilon(2.08).b_{\Diamond}.T_2$ obviously is not. Clearly, any operational derivative of an initially integral timed regular TMS will be well timed.

We define for any well timed RTMS T its *integral part* $T_0 = int(T) \in \mathcal{I}$ and *fractional part* $d = fr(T) \in [0,1[$ by $T = T_0^{(d)}$ (since we require $T_0 \in \mathcal{I}$ and $d \in [0,1[$, such a decomposition of any w.t. T into its integral and fractional parts is unique).

We shall find it convenient to have the notions of integral and fractional parts of RTMS also generalized to tuples (or, vectors) of RTMS. So, for $\overline{U} = \langle U_1, \ldots, U_n \rangle$ being a tuple (= a vector) of RTMSs, we define

$$\overline{int}(\overline{U}) = \langle int(U_1), \ldots, int(U_n) \rangle \text{ and } \overline{fr}(\overline{U}) = \langle fr(U_1), \ldots, fr(U_n) \rangle.$$

We call a RTMS T *delay sensitive* iff $T^{(d)}$ differs from T for some $d > 0$. Equivalently, T is delay sensitive, iff its NEF main (top level) defining equation contains at least one non-zero delay prefix[23]. It is not difficult to show that a w.t. RTMS is delay sensitive if and only if its integral part is.

For $S \in \mathcal{I}$ being defined in NEF over the variable set Var we define the set of RTMSs *possibly reachable* from S to be $re(S) = \{V^{(d)} | V \in Var,\, d \in \boldsymbol{R}_{+0}\}$. Observe that for any RTMS S the set of IT RTMS $S_0 \in re(S) \cap \mathcal{I}$ is finite.

Finally, for a composite TMS $S = (V_1 | \ldots | V_k) \setminus L$ we let [24].

$$re(S) = \{(U_1 | \ldots | U_k) \setminus L \mid U_i \in re(V_i), \forall i\}.$$

A TMS $(U_1 | \ldots | U_k) \setminus L$ is called well timed iff all its components U_i are.

7.2.2 Symbolic Characterization of RTMS Tuples

In this section we introduce some basic ingredients of the "region graph" construction (see e.g. [AD90]) serving as the main part of the symbolic representations of TMS specifications under analysis both in the time abstracted and time sensitive cases.

First, for any n-tuple of non-negative real numbers $\overline{x} = \langle x_1, x_2, \ldots, x_n \rangle \in \boldsymbol{R}^n_{+0}$ let us define its *symbolic value* $c(\overline{x})$ to be the ordering $\leq_\bullet$ of the $n+1$–element set $M_n = \{1, 2, \ldots, n\} \cup \{*\}$ such that

- for $i, j \in \{1, 2, \ldots, n\}$ $\;i \leq_\bullet j$ iff $x_i \leq x_j$, and
- for $i \in \{1, 2, \ldots, n\}$ $\;i \leq_\bullet *$ iff $x_i = 0$, as well as
- $* \leq_\bullet i$ for all $i \in M_n$.

We define $=_\bullet$ to be $\leq_\bullet \cap \leq_\bullet^{-1}$, and $i <_\bullet j$ whenever $i \leq_\bullet j$ and not $i =_\bullet j$.

It is easy to see that the set of all symbolic values $c(\overline{x})$ for $\overline{x} \in \boldsymbol{R}^n_{+0}$ precisely coincides with the set of all orderings $\leq'_\bullet$ of M_n which are *well formed* in a sense they are

- total (meaning $i \leq'_\bullet j$ or $j \leq'_\bullet i$ for all $i, j \in M_n$),

[23] So, $\tau_{\Box}.T_1 + \epsilon(2.15).a_{\Box}.T_2$ is delay sensitive (though the actual "semantical" delays of it are forbidden due to its first summand). On the contrary, $a_{\Box}.T_1 + b_{\Diamond}.T_2$ is not delay sensitive, since it does not change by delaying.

[24] Observe that for a composite S the set $re(S)$ most likely is an over-estimation of the set of TMSs reachable from S.

- transitive, and
- having $* \leq'_\bullet i$ for all $i \in M_n$.

We represent any well formed ordering $\leq_\bullet$ (and, so, any symbolic value $c(\overline{x})$) as a list $[s_1, s_2, \ldots, s_u]$, where $\{s_1, s_2, \ldots, s_u\}$ is a partitioning of M_n into non-empty sets such that

- $i =_\bullet j$ iff i and j belong to the same subset s_r of M_n, and
- for $l < r \leq u$ whenever $i \in s_l$ and $j \in s_r$, then $i <_\bullet j$ $(i, j \in M_n)$[25].

Further, let $\overline{U} = \langle U_1, U_2, \ldots, U_n \rangle$ be an n-tuple of w.t. RTMS. We define the *symbolic characterization* of $\overline{U}$ to be $\sigma(\overline{U}) = \langle \overline{int}(\overline{U}), c(\overline{fr}(\overline{U}))\rangle$ (observe that in the vector $\overline{fr}(\overline{U})$ every element is a nonnegative real, moreover it lies within the interval $[0, 1[$).

For a fixed $n \in \boldsymbol{N}$ we let $\Omega(n)$ denote the set of all pairs $\sigma = \langle \overline{V}, c\rangle$ where $\overline{V}$ is an n-tuple of IT RTMS and c is a well formed ordering of M_n. Again, one can easily show that $\Omega(n)$ precisely coincides with the set of all symbolic characterizations $\sigma(\overline{U})$ for $\overline{U}$ being an n-tuple of w.t. RTMS.

We call an element $\sigma = \langle \overline{V}, c\rangle \in \Omega(n)$ *boundary* iff for some $i \in M_n \setminus \{*\}$ simultaneously $i =_\bullet *$ and V_i is delay sensitive[26] [27]. A symbolic value $\sigma \in \Omega(n)$ which is not boundary is called *interior*.

Up to now we have defined a "static" symbolic characterization of arbitrary n-tuple $\overline{U}$ of RTMS. The next point is to introduce "transitions" between symbolic values in $\Omega(n)$, which we claim "represent" the actual delay transitions performed synchronously by all components of $\overline{U}$ (see Lemma 7.1 below). To begin with, let $next(T)$ for an integral timed RTMS $T \in \mathcal{I}$ denote $T^{(1)}$ (clearly, always $next(T) \in \mathcal{I}$).

Let $\sigma = \langle \overline{V}, c\rangle$ with $c = [s_1, s_2, \ldots, s_u]$. We define $succ(\sigma) = \langle \overline{V}', c'\rangle$:

- for σ being boundary
  - $\overline{V}' = \overline{V}$ (i.e. no changes are occurring in the integral part vector), and
  - $c' = [s_1^A, s_1^B, s_2, \ldots, s_u]$, where s_1^A and s_1^B is the partitioning of s_1 where $i \in s_1^B$ whenever V_i is delay sensitive, and $i \in s_1^A$ otherwise ($*$ always stays in s_1^A);
- for interior σ,
  - if $u > 1$, then $c' = [s_1 \cup s_u, s_2, \ldots, s_{u-1}]$. As to V', let $V_i' = next(V_i)$, if $i \in s_u$, and $V_i' = V_i$ otherwise;
  - if $u = 1$, then $succ(\sigma) = \sigma$ (this corresponds to the case when all components of $\overline{V}$ have already exhausted all their delay prefixes).

Observe that for this definition of $succ$ we will have $succ(\sigma)$ well formed (= being a symbolic characterization of some $\overline{U}$) whenever σ is. Moreover, one can show the following result. We let $\overline{U}^{(d)} = \langle U_1^{(d)}, \ldots, U_n^{(d)}\rangle$.

[25] For instance, if $\overline{x} = \langle 5.1,\ 4.2,\ 0.7,\ 0,\ 4.2,\ 5.1\rangle$, then $c(\overline{x}) = [\{*, 4\}, \{3\}, \{2, 5\}, \{1, 6\}]$.

[26] Here and further on we use an obvious convention that $\overline{V} = \langle V_1, \ldots, V_n\rangle$, $\overline{V}' = \langle V_1', \ldots, V_n'\rangle$, etc. for an appropriate n.

[27] The intuition of a boundary symbolic value $\sigma = \sigma(\overline{U})$ is that no matter how small shift (delay) will be performed simultaneously by all U_js, the resulting tuple of RTMSs will have another corresponding symbolic value - due to the component U_i which has integral initial delay prefixes (indicated by $fr(U_i) = 0$) and at least one of them not zero (indicated by the delay sensitiveness of U_i).

Lemma 7.1 *If $\sigma = \sigma(\overline{U})$, then $succ(\sigma) = \sigma(\overline{U}^{(d)})$ for some $d > 0$. Reverse, for every $d \geq 0$ we have $\sigma(\overline{U}^{(d)}) = succ^m(\sigma(\overline{U}))$ for some $m \geq 0$.*

Proof sketch: For a RTMS vector $\overline{U}$ such that $\sigma(\overline{U}) = \langle \overline{V}, c\rangle$ we define $\delta(\overline{U}) = 1 - fr(U_i)$, where i is one of indices in the last component of the list c. One can show that $succ(\sigma(\overline{U})) = \sigma(\overline{U}^{(\delta)})$, where

- $\delta = \delta(\overline{U})$ whenever $\sigma(\overline{U})$ is interior, and
- $\delta \in]0, \delta(\overline{U})[$, if $\sigma(\overline{U})$ is boundary.

This observation together with time determinacy and time continuity properties of RTMSs, taking into account also that for any $d \in \boldsymbol{R}_{+0}$ the symbolic value set $\{\sigma(\overline{U}^{(d')}) | 0 \leq d' \leq d\}$ is finite, leads to the proof of the lemma in both directions.$\Box$

We illustrate the introduced constructions on a simple example. Let

$$\overline{U} = \langle \epsilon(3.53).a_\Box.S_1,\ \epsilon(0.04).b_\Box.S_2,\ \epsilon(1.53).c_\Box.S_3,\ \epsilon(3.00).d_\Box.S_4,\ \epsilon(0).e_\Box.S_5\rangle.$$

Then $\sigma(\overline{U}) = \langle \overline{V}, c\rangle$, where $c = [\{*, 4, 5\}, \{1, 3\}, \{2\}]$[28] and $\overline{V}$ is the vector of integral timed RTMS

$$\overline{V} = \langle \epsilon(4).a_\Box.S_1,\ \epsilon(1).b_\Box.S_2,\ \epsilon(2).c_\Box.S_3,\ \epsilon(3).d_\Box.S_4,\ \epsilon(0).e_\Box.S_5\rangle$$

Observe that $V_4 = \epsilon(3).d_\Box.S_4$ is delay sensitive, whereas $V_5 = \epsilon(0).e_\Box.S_5$ is not. Since $4 =_\bullet *$, we conclude that σ is boundary. So, $succ(\sigma(\overline{U})) = \langle \overline{V}', c'\rangle$ for $\overline{V}' = \overline{V}$ and $c' = [\{*, 5\}, \{4\}, \{1, 3\} < \{2\}]$. Synchronous delay of any length $d \in]0, 0.04[$ of all components of $\overline{U}$ would lead to a RTMS vector $\overline{U}'$ whose symbolic characterization $\sigma(\overline{U}') = succ(\sigma(\overline{U}))$. As can easily be seen, $\sigma(\overline{U}')$ is interior.

7.3 Time Abstracted Refinements

In this section we show some details of the algorithms for deciding the time abstracted refinement relations $\overset{\bullet}{\lhd}$ and $\overset{\bullet}{\unlhd}$ between initially integral timed specifications.

Let $S' = (U_1 | \ldots | U_k) \setminus L$ be an arbitrary well timed TMS[29]. We define the symbolic representation of S' to be the triple $\langle\langle S'\rangle\rangle = \langle \overline{V}, c, L\rangle$ where $\langle \overline{V}, c\rangle = \sigma(\overline{U})$ is the symbolic characterization of the RTMS vector $\overline{U} = \langle U_1, \ldots, U_k\rangle$.

For a given IT TMS S we define the set of its *symbolic states* to be $\mathcal{X}_S = \{\langle\langle S_0\rangle\rangle \mid S_0 \in re(S)\}$. It can be easily seen that $\mathcal{X}_S$ is finite for every S.

One can observe that two TMSs S_1 and S_2 which have the same symbolic representation (i.e. $\langle\langle S_1\rangle\rangle = \langle\langle S_2\rangle\rangle$) have also identical "time abstracted" behaviours (whenever S_1 may/must be able to do a real action becoming S_1', this can be simulated by the same real action of S_2 becoming S_2' such that $\langle\langle S_2'\rangle\rangle = \langle\langle S_1'\rangle\rangle$, also if $S_1 \xrightarrow{\epsilon(d)}_m S_1'$, then $S_2 \xrightarrow{\epsilon(d')}_m S_2'$ for some d', S_2', again having $\langle\langle S_2'\rangle\rangle = \langle\langle S_1'\rangle\rangle$). One can find this result proven for a similar case of timed processes (in place of TMSs) in [LW90].

[28] Indeed, the corresponding fractional part vector is $\langle 0.47, 0.96, 0.47, 0, 0\rangle$. One needs to get slightly used to the fact that the RTMS itself has smaller waiting time than its integral part. This decision was taken to ensure that a RTMS is delay sensitive if and only if its integral part is.

[29] Recall that well timedness means consisting of RTMSs each of which can be obtained from IT RTMS by some delay.

We exploit this observation of (abstracted) behaviour coincidence of symbolically equally represented TMSs by reflecting these "equal" behaviours on the level of symbolic representations, what will lead directly to the needed refinement deciding algorithms.

More precisely, we define for $a \in \mathcal{A}ct$ and $m \in \{\Diamond, \Box\}$ the transitions $\xrightarrow{a}_m$ and $\xrightarrow{WT}_m$ between the symbolic states of IT TMS in the following (abstract) way

$$\frac{S' \xrightarrow{a}_m S''}{\langle\!\langle S' \rangle\!\rangle \xrightarrow{a}_m \langle\!\langle S'' \rangle\!\rangle}, \quad \frac{\langle\!\langle S'' \rangle\!\rangle = succ_m \langle\!\langle S' \rangle\!\rangle}{\langle\!\langle S' \rangle\!\rangle \xrightarrow{WT}_m \langle\!\langle S'' \rangle\!\rangle},$$

where $succ_m$ is a natural generalization of $succ$ from RTMS vector symbolic characterizations to symbolic states of TMSs, taking into account also maximal progress requirements. More precisely,

- $succ_m(\langle \overline{V}, c, L \rangle) = \langle \overline{V}', c', L \rangle$ for $\langle \overline{V}', c' \rangle = succ(\langle \overline{V}, c \rangle)$, if $(V_1 | \dots | V_k) \setminus L$ can not do a τ_{m^c} transition[30],
- $succ_m(\langle \overline{V}, c, L \rangle)$ is undefined otherwise.

The real (non-delay) transitions between symbolic states are computed as follows (it is not difficult to see that these computations correspond to the more abstract definition given above):

- whenever $a \in \mathcal{A}ct$, then $\langle \overline{V}, c, L \rangle \xrightarrow{a}_m \langle \overline{V}', c', L \rangle$ provided $a \notin L$ and for some i
  - $V_i \xrightarrow{a}_m V_i'$ (in the definition of V_i an appropriate summand must be contained);
  - $V_j' = V_j$ for all $j \neq i$ (all other components remain unchanged);
  - c' is obtained from c by moving the index i from its equivalence class to the class containing $*$ (followed by cleaning the list from empty elements - subsets of M_n).
- In addition, $\langle \overline{V}, c, L \rangle \xrightarrow{\tau}_m \langle \overline{V}', c', L \rangle$ whenever for some $i \neq j$ and some $\alpha \in \Lambda$
  - $V_i \xrightarrow{\alpha}_m V_i'$ and $V_j \xrightarrow{\bar{\alpha}}_m V_j'$;
  - $V_s' = V_s$ for all s such that $s \neq i$ and $s \neq j$;
  - c' is obtained from c by moving the indices i and j from their equivalence classes to the class containing $*$ (followed by cleaning the list from empty elements).

Let $A \xrightarrow{WT^*}_m B$ mean $A(\xrightarrow{WT}_m)^* B$ (i.e. the symbolic state B can be reached from A by zero or more $succ_m$ operations. We also introduce "time abstracted" transitions of symbolic states by defining $A \xrightarrow{a}{}^{W}_{m} B$ to stand for $A \xrightarrow{WT^*}_m \xrightarrow{a}_m \xrightarrow{WT^*}_m B$.

Observing Lemma 7.1 and "equivalence" of TMSs corresponding to one symbolic state, it is not difficult to justify the correctness of the following deciding procedures for $\dot{\triangleleft}$ and $\dot{\trianglelefteq}$.

For initially integral timed TMS S and T to check, if $S \,\dot{\triangleleft}\, T$, we define first $\mathcal{X}_{S,T} = \mathcal{X}_S \times \mathcal{X}_T$. For any $\mathcal{X} \subseteq \mathcal{X}_{S,T}$ we let $\mathcal{F}^{\dot{\triangleleft}}(\mathcal{X})$ to consist of all those pairs $\langle A, B \rangle$ for which

- whenever $A \xrightarrow{a}_\Diamond A'$ then $B \xrightarrow{a}{}^{W}_{\Diamond} B'$ for some B' with $\langle A', B' \rangle \in \mathcal{X}$,

[30] $\Box^c = \Diamond$, $\Diamond^c = \Box$.

- whenever $B \xrightarrow{a}_{\Box} B'$ then $A \xrightarrow{a}{}^{W}_{\Box} A'$ for some A' with $\langle A', B'\rangle \in \mathcal{X}$,
- whenever $A \xrightarrow{WT}_{\Diamond} A'$ then $B \xrightarrow{WT^*}_{\Diamond} B'$ for some B' with $\langle A', B'\rangle \in \mathcal{X}$,
- whenever $B \xrightarrow{WT}_{\Box} B'$ then $A \xrightarrow{WT^*}_{\Box} A'$ for some A' with $\langle A', B'\rangle \in \mathcal{X}$.

We have $S \overset{\bullet}{\lhd} T$ if and only if for some $\mathcal{X}$ both $\langle\langle\!\langle S\rangle\!\rangle, \langle\!\langle T\rangle\!\rangle\rangle \in \mathcal{X}$ and $\mathcal{X} \subseteq \mathcal{F}^{\overset{\bullet}{\lhd}}(\mathcal{X})$ (i.e. $\langle\langle\!\langle S\rangle\!\rangle, \langle\!\langle T\rangle\!\rangle\rangle$ is contained in the greatest fixpoint of $\mathcal{F}^{\overset{\bullet}{\lhd}}$).

In the weak case we distinguish observable actions (denoted by α) and the internal computations τ and let $\overset{\varepsilon}{\Longrightarrow}_m$ stand for $(\xrightarrow{\tau}_m \cup \xrightarrow{WT}_m)^*$. It can be easily shown (similarly, as in the strong case) that $S \overset{\bullet}{\unlhd} T$ if and only if $\langle\langle\!\langle S\rangle\!\rangle, \langle\!\langle T\rangle\!\rangle\rangle \in \mathcal{X} \subseteq \mathcal{X}_S \times \mathcal{X}_T$ for some $\mathcal{X}$ satisfying for every $\langle A, B\rangle \in \mathcal{X}$ the following conditions

- whenever $A \xrightarrow{\tau}_{\Diamond} A'$ or $A \xrightarrow{WT}_{\Diamond} A'$ then $B \overset{\varepsilon}{\Longrightarrow}_{\Diamond} B'$ for some B' with $\langle A', B'\rangle \in \mathcal{X}$,
- whenever $A \xrightarrow{\alpha}_{\Diamond} A'$ then $B \overset{\varepsilon}{\Longrightarrow}_{\Diamond}\xrightarrow{\alpha}_{\Diamond}\overset{\varepsilon}{\Longrightarrow}_{\Diamond} B'$ for some B' with $\langle A', B'\rangle \in \mathcal{X}$,
- whenever $B \xrightarrow{\tau}_{\Box} B'$ or $B \xrightarrow{WT}_{\Box} B'$ then $A \overset{\varepsilon}{\Longrightarrow}_{\Box} A'$ for some A' with $\langle A', B'\rangle \in \mathcal{X}$,
- whenever $B \xrightarrow{\alpha}_{\Box} B'$ then $A \overset{\varepsilon}{\Longrightarrow}_{\Box}\xrightarrow{\alpha}_{\Box}\overset{\varepsilon}{\Longrightarrow}_{\Box} A'$ for some A' with $\langle A', B'\rangle \in \mathcal{X}$.

7.4 Timed Refinements

To decide (strong or weak) time–sensitive refinements between two initially integral timed specifications S and T it is not in general possible to build finite partitionings of the state sets of S and T in which each element contains only "equivalent" states as we saw for the time–abstracted refinements (observe that any two TMSs[31] which have a slightest difference in their initial delay prefixes usually are semantically different).

However, following the ideas of [Č92b], one can attribute a symbolic characterization (similar in its nature to the symbolic state considered before in the time abstracted case) to every *pair* of well timed TMSs, so achieving the property that, whenever $\langle\!\langle S', T'\rangle\!\rangle = \langle\!\langle S'', T''\rangle\!\rangle$, then either both $S' \lhd T'$ and $S'' \lhd T''$, or none of these refinements hold[32]. We follow this line in the following technical description of algorithms deciding timed refinement relations.

For $S_0 = (U_1|\ldots|U_k) \setminus L$ and $T_0 = (U_{k+1}|\ldots|U_n) \setminus M$ being well timed TMS we define a symbolic representation (a symbolic state) $\langle\!\langle S_0, T_0\rangle\!\rangle$ of the pair $\langle S_0, T_0\rangle$ to be a 5-tuple $\langle \overline{V}, c, L, M, k\rangle$, where $\langle \overline{V}, c\rangle = \sigma(\overline{U})$ for $\overline{U} = \langle U_1, \ldots, U_k, U_{k+1}, \ldots, U_n\rangle$ being a concatenation of component vectors $\langle U_1, \ldots, U_k\rangle$ and $\langle U_{k+1}, \ldots, U_n\rangle$ for both TMSs S_0 and T_0[33].

For S and T being IT TMSs we define the symbolic state set $\mathcal{X}_{S,T}$ for characterizing their "joint behaviour" to be $\{\langle\!\langle S_0, T_0\rangle\!\rangle | S_0 \in re(S), T_0 \in re(T)\}$. One can show that $\mathcal{X}_{S,T}$ is finite for any S, T.

Similarly, as in time abstracted case, we define symbolic transitions between symbolic states. Here, however, we must distinguish between transitions which are done by "left" (i.e. "belonging" to S_0)

[31] One can consider also timed processes without modalities to observe the same effect.

[32] In other words, $S' \lhd T'$ implies $S'' \lhd T''$. The same property is true also for $\unlhd$, i.e. whenever $\langle\!\langle S', T'\rangle\!\rangle = \langle\!\langle S'', T''\rangle\!\rangle$, then $S' \unlhd T'$ implies $S'' \unlhd T''$.

[33] The number k is carried into the symbolic state to separate the components of the joint vector into those which "belong" to S_0 and which to T_0.

components from those done by " right" components (i.e. "belonging" to T_0). We do that by introducing an additional index 1 or 2 to the transition relation.

Assuming $m \in \{\Diamond, \Box\}$, $a \in \mathcal{A}ct$, we have first:

$$\frac{S \xrightarrow{a}_m S'}{\langle\!\langle S,T\rangle\!\rangle \xrightarrow{a}_{m,1} \langle\!\langle S',T\rangle\!\rangle} \quad \frac{T \xrightarrow{a}_m T'}{\langle\!\langle S,T\rangle\!\rangle \xrightarrow{a}_{m,2} \langle\!\langle S,T'\rangle\!\rangle}.$$

We say that $\langle\!\langle R_1,R_2\rangle\!\rangle \xrightarrow{WT}_{m,i}$ (without specifying the target of the transition) provided $R_i \xrightarrow{\epsilon(d)}_m$ for some $d>0$ (due to the maximal progress and transition liveness properties the necessary and sufficient condition for such delays is the absence of $R_i \xrightarrow{\tau}_{m^c}$ transition possibility).

For $X = \langle \overline{V}, c, L, M, k\rangle$ such that $X \xrightarrow{WT}_{m,1}$ and $X \xrightarrow{WT}_{m,2}$ we define its "m-delay successor" to be $succ_m(X) = \langle \overline{V}', c', L, M, k\rangle$, where $\langle \overline{V}', c'\rangle = succ(\langle \overline{V}, c\rangle)$ (see Section 7.2.2 for definition of time successor $succ$ for a symbolic characterization of a vector of RTMS)[34].

Given 5-tuple representations of the symbolic states, the computation of the transition relations $\xrightarrow{a}_{m,i}$ between them in accordance with the given abstract definition is rather straightforward, so we show only an example:

- For $a \in \mathcal{A}ct$, if $a \neq \tau$, then $\langle \overline{V}, c, L, M, k\rangle \xrightarrow{a}_{m,1} \langle \overline{V}', c', L', M', k'\rangle$ iff
  - $L' = L$, $M' = M$ and $k' = k$, as well as $a \notin L$;
  - $V_i \xrightarrow{a}_m V_i'$ for some $i \leq k$; $V_j' = V_j$ for all $j \neq i$;
  - c' is obtained from c by moving i to the equivalence class containing $*$ (followed by cleaning the list from empty elements).

In the case of symbolic time transitions we perform the check if $X \xrightarrow{WT}_{m,i}$ by checking the negation of $X \xrightarrow{\tau}_{m^c,i}$.

The following algorithm of deciding (strong) timed refinement between IT TMS can be proven correct following very closely the arguments from [Č92b].

Given $\mathcal{X} \subseteq \mathcal{X}_{S,T}$ we define $\mathcal{F}^{\triangleleft}_{S,T}(\mathcal{X})$ to consist of those and only those $X \in \mathcal{X}_{S,T}$ for which

- $X \xrightarrow{a}_{\Diamond,1} X_1$ implies $X_1 \xrightarrow{a}_{\Diamond,2} X_2$ for $X_2 \in \mathcal{X}$,
- $X \xrightarrow{a}_{\Box,2} X_1$ implies $X_1 \xrightarrow{a}_{\Box,1} X_2$ for $X_2 \in \mathcal{X}$,
- whenever $X \xrightarrow{WT}_{\Diamond,1}$ then $succ_\Diamond(X) \in \mathcal{X}$,
- whenever $X \xrightarrow{WT}_{\Box,2}$ then $succ_\Box(X) \in \mathcal{X}$.

We conclude that $S \triangleleft T$ iff $\langle\!\langle S,T\rangle\!\rangle$ is contained in the greatest fixpoint of $\mathcal{F}^{\triangleleft}_{S,T}$.

In order to decide the weak refinement we let $X \xRightarrow{\tau}_{m,i} X'$ mean $X(\xrightarrow{\tau}_{m,i})^* X'$ and $X \xRightarrow{\alpha}_{m,i} X'$ mean $X \xRightarrow{\tau}_{m,i} \xrightarrow{\alpha}_{m,i} \xRightarrow{\tau}_{m,i} X'$. Let also a symbolic state $X = \langle \overline{V}, c, L, M, k\rangle$ be called boundary or interior in accordance with the pair $\langle \overline{V}, c\rangle$.

Let us recall that the definition of the weak (τ-abstracted) refinement in Section 4 in case of $S \trianglelefteq T$ allows to match a delay transition $S \xrightarrow{\epsilon(d)}_\Diamond S'$ in general by a chain of transitions

[34]Observe that for $X = \langle\!\langle S_0,T_0\rangle\!\rangle$ $succ_m(X)$, if defined, will be $\langle\!\langle S_0',T_0'\rangle\!\rangle$, where S_0' and T_0' are obtained by *synchronous* delaying of S_0 and T_0 for some time amount $d>0$.

$$T(\xrightarrow{\tau}_\diamond)^* \xrightarrow{\epsilon(d_1)}_\diamond (\xrightarrow{\tau}_\diamond)^* \cdots \xrightarrow{\epsilon(d_k)}_\diamond (\xrightarrow{\tau}_\diamond)^* T'$$

so that $d_1 + \ldots + d_k = d$ and $S' \trianglelefteq T'$ ($T \xrightarrow{\epsilon(d)}_\Box T'$ is to be matched by a similar appropriate chain from S). It turns out that in the deciding procedure below we need to consider symbolic transitions following the same general pattern, at least for matching the $\xrightarrow{\epsilon(d)}_\diamond$ transitions[35].

To describe the search for timed matching transitions on the symbolic level, first, if $succ_\diamond(X)$ is defined, we let $X \xrightarrow{0} X'$ and $X \xrightarrow{1} X''$, where

- $X' = succ_\diamond(X)$ and $X'' = succ_\diamond(succ_\diamond(X))$ for X being boundary, and
- $X' = X$ and $X'' = succ_\diamond(X)$ for interior X.

One can say that $\xrightarrow{0}$ is "waiting to become interior" and $\xrightarrow{1}$ is "waiting to become boundary". Semantically, if $\delta = \delta(S,T)$ is the difference between 1 and the largest delay fractional part in S, T, then $\langle\langle S,T\rangle\rangle \xrightarrow{0} \langle\langle S',T'\rangle\rangle$ and $\langle\langle S,T\rangle\rangle \xrightarrow{1} \langle\langle S'',T''\rangle\rangle$, where

- $S \xrightarrow{\delta}_\diamond S''$ and $T \xrightarrow{\delta}_\diamond T''$, as well as
- $S \xrightarrow{\delta'}_\diamond S'$ and $T \xrightarrow{\delta'}_\diamond T'$ with δ' chosen to be any number within $]0,\delta[$.

Intuitively $\xrightarrow{1}$ can be also explained as a symbolic move corresponding to the simultaneous waiting of TMSs until the first new timer fractional part becomes 0 (for what two $succ_\diamond$ moves are necessary, if the pair of initial TMSs belonged to a boundary region).

Further, let $\xRightarrow{0*} = (\xRightarrow{\tau}_{\diamond,2} \cup \xrightarrow{0})^*$. Observe that this relation comprises interleaving of "right" symbolic $\tau_\diamond$-moves (we need to introduce this construction for only one direction) and elementary delay transitions. One can show, for instance, that $\langle\langle S_0,T_0\rangle\rangle \xRightarrow{0*} X$ implies that for some $d \in [0, \delta(S_0,T_0)[$ [36] both $S_0 \xrightarrow{\epsilon(d)}_\diamond S'$ and $T_0 \xRightarrow{\epsilon(d)}_\diamond T'$ so that $X = \langle\langle S',T'\rangle\rangle$ (observe the same d in both transition chains).

Following the ideas of [Č92b] and [Č92a] one can prove that $S \trianglelefteq T$ if and only if there exists $\mathcal{X} \subseteq \mathcal{X}_{S,T}$ such that both $\langle\langle S,T\rangle\rangle \in \mathcal{X}$ and for every $X \in \mathcal{X}$:

- whenever $X \xrightarrow{a}_{\diamond,1} X_1$ then $X_1 \xRightarrow{a}_{\diamond,2} X_2$ for $X_2 \in \mathcal{X}$,
- whenever $X \xrightarrow{a}_{\Box,2} X_1$ then $X_1 \xRightarrow{a}_{\Box,1} X_2$ for $X_2 \in \mathcal{X}$,
- whenever $X \xrightarrow{WT}_{\Box,2}$ then $X \xRightarrow{\tau}_{\Box,1} X_1$ for some X_1 such that $succ_\Box(X_1) \xRightarrow{\tau}_{\Box,1} X_2$ for $X_2 \in \mathcal{X}$,
- whenever $X \xrightarrow{WT}_{\diamond,1}$ then both

[35] Remarkably, the $\longrightarrow_\Box$ transition system of TMS satisfies the classical maximal progress property ([Wan90]), saying that for no TMS S it would be the case that both $S \xrightarrow{\tau}_\Box S'$ and $S \xrightarrow{\epsilon(d)}_\Box S''$ for any d, S', S''. This implies that, if $S \xrightarrow{\epsilon(d)}_\Box$ for some $d > 0$, then for sufficiently small d's we have $S \xRightarrow{\epsilon(d')}_\Box S'$ if and only if $S \xrightarrow{\epsilon(d')}_\Box S'$ (i.e. during some initial period the delay of S can not be interrupted by any internal transitions). One can show that this initial period is at least as long as the difference between 1 and the largest S component fractional part. Using this observation the "matching region" construction in the $\Box$-timed cases can be taken to be substantially simpler (see the deciding procedure below). For $\longrightarrow_\diamond$ transition system the classical maximal progress property is not satisfied, for instance, on $\tau_\diamond.S$ for any S.

[36] In fact, d can be taken 0, if there were no $\xrightarrow{0}$ transitions actually occurring within this $\xRightarrow{0*}$, otherwise d can be taken to be any value from $]0, \delta(S_0,T_0)[$. It can be noted that the following transitions could be possible (actually, only in somewhat degenerate cases) also for larger values of d.

- $X \overset{0*}{\Longrightarrow} \overset{0}{\longrightarrow} \overset{\tau}{\Longrightarrow}_{\diamond,2} X_1$ for some $X_1 \in \mathcal{X}$ [37] and
- $X \overset{0*}{\Longrightarrow} \overset{1}{\longrightarrow} \overset{\tau}{\Longrightarrow}_{\diamond,2} X_2$ for $X_2 \in \mathcal{X}$ [38].

The most complicated $X \overset{WT}{\longrightarrow}_{\diamond,1}$ case in the above definition for a pair of TMSs $\langle S_0, T_0 \rangle \in X \in \mathcal{X}$ makes immediate guarantees that whenever $S_0 \overset{\epsilon(d)}{\longrightarrow}_{\diamond} S'$ for some $d \leq \delta(S_0, T_0)$, then $T_0 \overset{\epsilon(d)}{\Longrightarrow}_{\diamond} T'$ with $X' = \langle\!\langle S', T' \rangle\!\rangle \in \mathcal{X}$ [39]. The necessary extension to the case of arbitrary $d \in \boldsymbol{R}_{+0}$ is then done by induction over time moments when fractional parts of specification components become 0 [40] (precisely as in the another timed case of $X \overset{WT}{\longrightarrow}_{\Box,2}$). For more details of the correctness proof we refer to [Č92b, Č92a].

7.5 Trajectory Step Refinement

For the procedure deciding, if $S \lhd^T T$, we keep the notation introduced to describe symbolic deciding of $\trianglelefteq$. What is changing here, is just the symbolic clause for matching the $\overset{\epsilon(d)}{\longrightarrow}_{\diamond}$ moves of S, precisely to reflect the requirement taken from the definition of $S \lhd^T T$ that all regions corresponding to the points in the step sequence of $T \overset{\epsilon(d)}{\Longrightarrow}_{\diamond} T'$ should be "matching".

So, it can be proven that $S \lhd^T T$ if and only if there exists $\mathcal{X} \subseteq \mathcal{X}_{S,T}$ such that both $\langle\!\langle S, T \rangle\!\rangle \in \mathcal{X}$ and for every $X \in \mathcal{X}$:

[37] Observe that this case reduces to triviality, if X is interior and $X \overset{WT}{\longrightarrow}_{\diamond,2}$, by letting $X_1 = X$.

[38] In fact this is a slight simplification. Actually one needs to make sure (for the given algorithm to work properly) that during the $\overset{0*}{\Longrightarrow}$ transitions the last component of "fractional part list" of X is not lost. This is implemented by adding into this last class an extra dummy element (so X becomes some X^+) to be removed later (when X^+ has become X_2^+, we keep just X_2). In theory this construction corresponds to an extra clock, and is elaborated for a very similar situation in [Č92a].

[39] Recall that $\delta(S_0, T_0)$ is the distance from the largest fractional part of components of either S_0 or T_0 to 1.

[40] It can be shown that there can not be more than a finite amount of those within any finite time interval.

- whenever $X \xrightarrow{a}_{\Diamond,1} X_1$ then $X_1 \overset{a}{\Longrightarrow}_{\Diamond,2} X_2$ for $X_2 \in \mathcal{X}$,
- whenever $X \xrightarrow{a}_{\Box,2} X_1$ then $X_1 \overset{a}{\Longrightarrow}_{\Box,1} X_2$ for $X_2 \in \mathcal{X}$,
- whenever $X \xrightarrow{WT}_{\Box,2}$ then $X \overset{\tau}{\Longrightarrow}_{\Box,1} X_1$ for some X_1 such that $succ_{\Box}(X_1) \overset{\tau}{\Longrightarrow}_{\Box,1} X_2$ for $X_2 \in \mathcal{X}$,
- whenever $X \xrightarrow{WT}_{\Diamond,1}$ then both
  - $X \overset{\tau}{\Longrightarrow}_{\Diamond,2} X_0 \xrightarrow{0}\overset{\tau}{\Longrightarrow}_{\Diamond,2} X_1$ so that $X_i \in \mathcal{X}$ for $i = 0, 1$, and
  - $X \overset{\tau}{\Longrightarrow}_{\Diamond,2} X_0' \xrightarrow{0}\overset{\tau}{\Longrightarrow}_{\Diamond,2} X_1' \ldots \xrightarrow{0}\overset{\tau}{\Longrightarrow}_{\Diamond,2} X_m' \xrightarrow{1}\overset{\tau}{\Longrightarrow}_{\Diamond,2} X_{m+1}'$ so that $X_i' \in \mathcal{X}$ for all $i = 0, 1, \ldots, m+1$, $m \geq 0$ [41].

We shall not be concerned here with discussions on the efficiency of checking $\lhd^T$, however, let us make the obvious observation that both parts of the matching requirement in the last clause (when $X \xrightarrow{WT}_{\Diamond,1}$) could usually be easily fulfilled by a single region sequence, namely $X \overset{\tau}{\Longrightarrow}_{\Diamond,2} X_0' \xrightarrow{0}\overset{\tau}{\Longrightarrow}_{\Diamond,2} X_1' \ldots \xrightarrow{0}\overset{\tau}{\Longrightarrow}_{\Diamond,2} X_m' \xrightarrow{1}\overset{\tau}{\Longrightarrow}_{\Diamond,2} X_{m+1}'$, where $m \geq 1$.

8 Conclusions

In this paper we have presented the theory of TMS together with the automatic refinement checking tool Epsilon. The theory TMS is an extension of real–time process calculi with the specific aim of allowing *loose* or *partial* specifications. Looseness of specifications was achieved by introducing two modalities to transitions of specifications: a *may* and a *must* modality. As a natural consequence of having two kinds of transitions we generalized in a direct way various notions of process equivalences (strong equivalence, τ–abstracted equivalence, time–abstracted equivalence and τ– and time–abstracted equivalence) and introduced corresponding *refinement orderings* between specifications.

We showed how the notion of refinement allows for *generality* in the theory, that is, a single correctness proof in TMS may capture a whole (possibly) infinite family of correctness proofs in process calculi. Further, it turned out that the theory is *compositional* to a large extent; in particular the non time–abstracted refinements are preserved (at least) by parallel composition. Together, generality and compositionality makes TMS a useful formalism for *stepwise refinement* development. In a stepwise refinement development of a system the initial specification is rather abstract and permits a wide range of implementations. In each refinement step the set of possible implementation is reduced while still maintaining correctness with respect to the initial specification. Eventually an implementation is reached.

We consider (and this may be stated in a formal manner) TMS to be a theory in between timed process algebras and timed logics. TMS possesses both the compositional strength of algebras and is like logics a theory for expressing loose specifications of real–time systems. However, in contrast to (real–time) logics where compositional verification so far has not been offered, TMS supports compositional verification.

TMS and its automatic refinement checking tool has been successfully applied to a not completely trivial example of a train crossing. In particular we outlined how compositionality can be taken advantage of during refinement checking.

Epsilon, however is a prototype tool and we have ideas of how to improve the time and space performance of the tool. One possible idea is to investigate combinations of the kernel of Epsilon (that is, the

[41] Here we should also be concerned with keeping the last component of X fractional part list, precisely as in similar situation when deciding $\unlhd$.

local model checking technique of [Lar92]) with the state space partitioning algorithms that has proven successful for verification of real–timed systems (e.g. the temporal logic model checking partitioning algorithm presented in [ACH+92]). Other performance improving ideas include utilizing algebraic laws while performing the correctness checking.

The idea of introducing looseness by allowing two modalities on transitions is completely orthogonal to process algebra. Hence, another topic of research interest is that of introducing looseness to other kinds of real–time theories, for instance timed graphs [Dil89].

References

[ACD90] Rajeev Alur, Costas Courcoubetis, and David Dill. Model–checking for real–time systems. In *Proceedings of the Fifth IEEE Symposium on Logic in Computer Science*, 1990.

[ACH+92] R. Alur, C. Courcoubetis, N. Halbwachs, D. Dill, and H. Wong-Toi. Minimization of timed transition systems. In *CONCUR'92*, volume 630 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[AD90] R. Alur and D. Dill. Automata for modelling real–time systems. In *ICALP'90*, volume 443 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[AD91] R. Alur and D. Dill. The theory of timed automata. In *Real–Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[BB89] J.C.M. Baeten and J.A. Bergstra. Real time process algebra. Technical Report P8916, University of Amsterdam, 1989.

[BL90] G. Boudol and K.G. Larsen. Graphical versus logical specifications. In *Proceedings of CAAP'90*, volume 431 of *Lecture Notes in Computer Science*, 1990.

[BLS92] A. Børjesson, K.G. Larsen, and A. Skou. Generality in design and compositional verification using tav. In *Proceedings of FORTE'92*, 1992.

[Dil89] D.L. Dill. Timing assumptions and verification of finite–state concurrent systems. *Lecture Notes in Computer Science*, 407, 1989.

[GLZ89] J.C. Godskesen, K.G. Larsen, and M. Zeeberg. TAV (tools for automatic verification) — users manual. Technical report, University of Aalborg, Denmark, 1989.

[God94] Jens Chr. Godskesen. *Timed Modal Specifications — A Theory for Verification of Real–Time Concurrent Systems.* Ph.d. thesis, University of Aalborg, Denmark, 1994.

[HL89] H. Hüttel and K.G. Larsen. The use of static constructs in a modal process logic. In *Proceedings of Logic at Botik'89*, volume 363 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.

[Lar90] K.G. Larsen. Modal specifications. In *Proceedings of Workshop on Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, 1990.

[Lar92] K.G. Larsen. Efficient local correctness checking. In *Proceedings of CAV'92*, volume 663 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[LT88] K.G. Larsen and B. Thomsen. A modal process logic. In *Proceedings LICS'88*, 1988.

[LW90] K.G. Larsen and Y. Wang. Time abstracted bisimulation: Implicit specifications and decidability. In *Proceedings of MFPS'93*, Lecture Notes in Computer Science. Springer-Verlag, 1990.

[Mil89] Robin Milner. *Communication and Concurrency.* Series in Computer Science. Prentice–Hall International, 1989.

[NSY91] X. Nicollin, J. Sifakis, and S. Yovine. From ATP to timed graphs and hybrid systems. In *Real–Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[NSY92] X. Nicollin, J. Sifakis, and S. Yovine. Compiling real–time specifications into extended automata. *IEEE TSE Special Issue on Real–Time Systems*, September 1992.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *5th GI Conference*, volume 104 of *Lecture Notes in Computer Science*, 1981.

[SV89] R. De Simone and D. Vergamini. Aboard AUTO. Technical Report 111, INRIA, Sofia–Antipolis, 1989.

[Č92a] K. Čerāns. *Algorithmic Problems in Analysis of Real Time System Specifications.* Dr. sc. comp. thesis, University of Latvia, Riga, 1992.

[Č92b] K. Čerāns. Decidability of bisimulation equivalences for processes with parallel timers. In *Proceedings of CAV'92*, volume 663 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[Wan90] Y. Wang. Real–time behaviour of asynchronous agents. In *Proceedings of CONCUR'90*, volume 458 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

Recent BRICS Report Series Publications

RS-97-11 **Kārlis Čerāns, Jens Chr. Godskesen, and Kim G. Larsen.** ***Timed Modal Specification — Theory and Tools*****. April 1997. 32 pp.**

RS-97-10 **Thomas Troels Hildebrandt and Vladimiro Sassone.** ***Transition Systems with Independence and Multi-Arcs*****. April 1997. 20 pp. Appears in Peled, Pratt and Holzmann, editors,** ***DIMACS Workshop on Partial Order Methods in Verification*****, POMIV '96, pages 273–288.**

RS-97-9 **Jesper G. Henriksen and P. S. Thiagarajan.** ***A Product Version of Dynamic Linear Time Temporal Logic*****. April 1997. 18 pp. To appear in** ***Concurrency Theory: 7th International Conference*****, CONCUR '97 Proceedings, LNCS, 1997.**

RS-97-8 **Jesper G. Henriksen and P. S. Thiagarajan.** ***Dynamic Linear Time Temporal Logic*****. April 1997. 33 pp.**

RS-97-7 **John Hatcliff and Olivier Danvy.** ***Thunks and the λ-Calculus (Extended Version)*****. March 1997. 55 pp. Extended version of article to appear in the** ***Journal of Functional Programming*****.**

RS-97-6 **Olivier Danvy and Ulrik P. Schultz.** ***Lambda-Dropping: Transforming Recursive Equations into Programs with Block Structure*****. March 1997. 53 pp. Extended version of an article to appear in the 1997 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation (PEPM '97), Amsterdam, The Netherlands, June 1997.**

RS-97-5 **Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke.** ***First-Order Logic with Two Variables and Unary Temporal Logic*****. March 1997. 18 pp. To appear in** ***Twelfth Annual IEEE Symposium on Logic in Computer Science*****, LICS '97 Proceedings.**

RS-97-4 **Richard Blute, Josée Desharnais, Abbas Edalat, and Prakash Panangaden.** ***Bisimulation for Labelled Markov Processes*****. March 1997. 48 pp. To appear in** ***Twelfth Annual IEEE Symposium on Logic in Computer Science*****, LICS '97 Proceedings.**

RS-97-3 **Carsten Butz and Ieke Moerdijk.** ***A Definability Theorem for First Order Logic*****. March 1997. 10 pp.**

RS-97-2 **David A. Schmidt.** ***Abstract Interpretation in the Operational Semantics Hierarchy*****. March 1997. 33 pp.**