Compositional Verification of Real-time Systems*

Edward Chang   Zohar Manna   Amir Pnueli

Dept. of Computer Science
Stanford University
Stanford, CA 94305

Dept. of Computer Science
Weizmann Institute of Science
Rehovot, Israel

Abstract

This paper presents a compositional proof system for the verification of real-time systems. Real-time systems are modeled as timed transition modules, which explicitly model interaction with the environment and may be combined using composition operators.

Composition rules are devised such that the correctness of a system may be determined from the correctness of its components. These proof rules are demonstrated on Fischer's mutual exclusion algorithm, for which mutual exclusion and bounded response are proven.

1 Introduction

This paper presents a compositional proof system for the verification of real-time systems. Real-time systems are a particular class of *reactive systems*, i.e., systems exhibiting ongoing interaction with the environment, distinguished by the observation that the correctness of a real-time system depends not only on the qualitative aspects of its behavior, but also on the quantitative timeliness of its behavior.

Real-time systems are modeled as *timed transition modules*, presented in section 2, in contrast to timed transition systems [5]. Timed transition modules explicitly model interaction with the environment and may be combined using composition operations, e.g., parallel composition.

The specification language *metric temporal logic* (MTL) [1] [8], interpreted over a discrete time domain, is extended in section 3 to include past operators. A complete axiomatization of MTL with past operators is presented in [2]. Several proof rules for global verification were suggested in [5] and shown to be complete for a restricted class of properties; new proof rules, accommodating past operators, are presented in section 5 and proved to be complete for verifying a broader class of properties.

A complex real-time system is typically modeled as a hierarchical composition of timed transition modules. Given a specification for such a system, the proposed verification methodology is to devise suitable specifications for each component such that the correctness of the system may be determined from the correctness of its components. The composition rules in section 6 are devised to support compositional verification. These proof rules are demonstrated on Fischer's mutual exclusion algorithm, for which mutual exclusion and bounded response are proven.

During the verification of some real-time property, it is often necessary to establish certain untimed properties of the system, i.e., properties that do not depend on timing constraints or the timed behavior of the system. The usual verification techniques [9] for untimed transition systems may be usefully applied to verify these properties.

The framework of timed transition modules presented in this paper applies when processes communicate through shared variables and/or message-passing [2]. Related work in formal methods for real-time message-passing systems includes [3] [6] [7].

2 Timed Transition Modules

The timed transition modules defined in this section are based on *timed transition systems*, as presented in [5]. The set $\mathcal{V}$ is an unspecified global vocabulary of system variables.

*This research was supported in part by the National Science Foundation under grant CCR-92-23226, by the Defense Advanced Research Projects Agency under contract NAG2-703, by the United States Air Force Office of Scientific Research under contract F49620-93-1-0139, and by the European Community ESPRIT Basic Research Action Project 6021 (REACT).

A *timed transition module* M consists of the following components:

- $V \subseteq \mathcal{V}$: A finite set of variables, partitioned into *private variables* V^p and *shared variables* V^s.
- $\mathcal{T}$: A finite set of *transitions*, i.e., assertions over V and a corresponding set of primed variables V'. A state s' is a *τ-successor* of s if

  $$\langle s, s' \rangle \;\models\; \tau \wedge \bigwedge_{u \notin V} (u' = u)$$

  where $\langle s, s' \rangle$ is the joint interpretation which interprets $x \in V$ as $s[x]$ and $x' \in V'$ as $s'[x]$. A transition τ is said to be *enabled* on a state s if there is some τ-successor s' of s; the *enabling condition* $En(\tau)$ characterizes the states on which τ is enabled.

  For every τ-successor s' of a state s, where $\tau \in \mathcal{T}$, there must be some $u \in V^p$ such that $s'[u] \neq s[u]$. Furthermore, each transition must be *self-disabling*, i.e., each τ must be disabled on every τ-successor of any state s. These conditions are satisfied, for instance, by a (private) control variable that changes, when each transition is taken, to a value which disables the transition.
- Θ: An *initial condition*, i.e., a satisfiable assertion over V characterizing the initial states of the transition module.
- l: A *lower bound* $l^\tau \in \mathbb{N}$ for each transition τ.
- u: An *upper bound* $u^\tau \in \mathbb{N}$ for each transition τ.

2.1 The Environment Transition

The environment transition τ_E is intended to capture every possible behavior of the environment of M. Specifically, a state s' is a τ_E-successor of s if s and s' agree on the private variables of M, i.e.,

$$\langle s, s' \rangle \;\models\; \bigwedge_{u \in V^p} (u' = u)$$

2.2 Computations of a Transition Module

A *timed state sequence* ρ may be represented as a pair (σ, T), where $\sigma : s_0, s_1, \ldots$ is an infinite sequence of $\mathcal{V}$-states and $T = T_0, T_1, \ldots$ is a corresponding sequence of time values satisfying the following conditions:

- *Monotonicity:* For all $i \geq 0$:

  either $T_{i+1} = T_i$,
  or $T_{i+1} = T_i + 1$ and $s_{i+1} = s_i$.

  The case that time increases is referred to as a *tick step.*
- *Progress:* For all $i \geq 0$

  there is some $j > i$ such that $T_j > T_i$.

The adequacy of a digital clock model is considered in [4].

A timed state sequence ρ is defined to be a *computation* of M if it satisfies the following requirements:

- *Initiation:* The state s_0 is an initial state.
- *Consecution:* For every j, where $0 \leq j$:
  - either the state s_{j+1} is a τ-successor of the state s_j for some $\tau \in \mathcal{T}$, i.e., transition τ was *taken* at position j in σ,
  - or the state s_{j+1} is a τ_E-successor of state s_j, i.e., an *environment step* is taken at position j in σ.
- *Lower bound*

  No transition $\tau \in \mathcal{T}$ is taken unless it has been continually enabled at least l^τ time units.
- *Upper bound*

  No transition $\tau \in \mathcal{T}$ may be continually enabled for more than u^τ time units without being taken.

The set of computations of M is denoted $Comp(M)$.

2.3 Parallel Composition

Timed transition modules M_1 and M_2 may be composed if the private variables of each module are not variables of the other module. In particular:

- $V_1^p \cap V_2 = \emptyset$
- $V_2^p \cap V_1 = \emptyset$
- $\Theta_1 \wedge \Theta_2$ is satisfiable

The parallel composition of M_1 and M_2 yields the transition module M, defined as follows:

- $V \;=\; V_1 \cup V_2$

  $$\begin{aligned} V^p &= V_1^p \cup V_2^p \\ V^s &= V_1^s \cup V_2^s \end{aligned}$$
- $\mathcal{T} \;=\; \mathcal{T}_1 \;\cup\; \mathcal{T}_2$

- $\Theta = \Theta_1 \wedge \Theta_2$
- $l^\tau = l_1^\tau$ if $\tau \in \mathcal{T}_1$, otherwise l_2^τ.
- $u^\tau = u_1^\tau$ if $\tau \in \mathcal{T}_1$, otherwise u_2^τ.

The following composition theorem states that the computations of M, formed as the parallel composition of M_1 and M_2, consist of exactly the computations of both M_1 and M_2.

Theorem 2.1 $Comp(M) = Comp(M_1) \cap Comp(M_2)$.

Proof. Consider a computation of M_1 and M_2. In order to establish the consecution requirement for M, observe that for each $j \geq 0$ it is not possible that some τ_1 of M_1 and some τ_2 of M_2 are both taken at j. The remaining conditions are straightforward. ∎

3 Metric Temporal Logic

A *metric temporal formula* is constructed out of state formulas, i.e., first-order assertions, the boolean operators $\neg$ and $\vee$, and the following temporal operators:

$\bigcirc_{\sim c}$	— Next	$\mathcal{U}_{\sim c}$	— Until
$\ominus_{\sim c}$	— Previous	$\mathcal{S}_{\sim c}$	— Since

where $\sim$ is one of $\{<, =, >, \equiv_d\}$, $c \geq 0$, and $d \geq 2$. Note that $\equiv_d$ denotes congruence modulo d.

Given a timed state sequence $\rho = (\sigma, T)$ and a metric temporal formula p, $(\rho, j) \models p$ denotes that p holds at position j in ρ. For a state formula p,

$(\rho, j) \models p \quad \Longleftrightarrow \quad s_j \models p$

That is, p is evaluated locally, using the interpretation given by s_j. The state s_j is a *p-state* if p holds on s_j.

$$
\begin{array}{lcl}
(\rho,j) \models \neg p & \Longleftrightarrow & (\rho,j) \not\models p \\
(\rho,j) \models p \vee q & \Longleftrightarrow & (\rho,j) \models p \text{ or } (\rho,j) \models q \\
(\rho,j) \models \bigcirc_{\sim c}\, p & \Longleftrightarrow & (\rho,j+1) \models p \\
 & & \quad \text{and } (T_{j+1} - T_j) \sim c \\
(\rho,j) \models p\, \mathcal{U}_{\sim c}\, q & \Longleftrightarrow & \text{for some } k,\ j \leq k, \\
 & & (\rho,k) \models q \text{ and } (T_k - T_j) \sim c, \\
 & & \text{and for every } i,\ j \leq i < k, \\
 & & (\rho,i) \models p \\
(\rho,j) \models \ominus_{\sim c}\, p & \Longleftrightarrow & j \geq 1 \text{ and } (\rho,j-1) \models p \\
 & & \quad \text{and } (T_j - T_{j-1}) \sim c \\
(\rho,j) \models p\, \mathcal{S}_{\sim c}\, q & \Longleftrightarrow & \text{for some } k,\ 0 \leq k \leq j, \\
 & & (\rho,k) \models q \text{ and } (T_j - T_k) \sim c, \\
 & & \text{and for every } i,\ k < i \leq j, \\
 & & (\rho,i) \models p
\end{array}
$$

An unbounded version $p\,\mathcal{U}\,q$ may be defined as $p\,\mathcal{U}_{=0}\,q \ \vee\ p\,\mathcal{U}_{>0}\,q$, and similarly for an unbounded version of $\mathcal{S}$. Additional temporal operators can be defined as follows:

$$
\begin{array}{lcll}
\Diamond_{\sim c}\, p & = & true\ \mathcal{U}_{\sim c}\, p & \text{– Eventually} \\
\Box_{\sim c}\, p & = & \neg \Diamond_{\sim c} \neg p & \text{– Henceforth} \\
p\, \mathcal{W}_{\sim c}\, q & = & \Box p \vee (p\, \mathcal{U}_{\sim c}\, q) & \text{– Waiting-for} \\
p \Rightarrow q & = & \Box(p \rightarrow q) & \text{– Entailment} \\
p \Leftrightarrow q & = & \Box(p \leftrightarrow q) & \text{– Congruence} \\
\diamondminus_{\sim c}\, p & = & true\ \mathcal{S}_{\sim c}\, p & \text{– Sometime in the past} \\
\boxminus_{\sim c}\, p & = & \neg \diamondminus_{\sim c} \neg p & \text{– Always in the past} \\
p\, \mathcal{B}_{\sim c}\, q & = & \boxminus p \vee (p\, \mathcal{S}_{\sim c}\, q) & \text{– Back-to}
\end{array}
$$

Similarly, unbounded versions of the above operators can be defined from the unbounded $\mathcal{U}$ and $\mathcal{S}$ operators. Note that, since time can change by at most 1 between two adjacent states, it is sufficient to allow $\sim c$ to be $= 0$ or $= 1$ for the immediate operators $\bigcirc_{\sim c}$ and $\ominus_{\sim c}$. A formula that does not include the *future operators* $\bigcirc_{\sim c}$, $\mathcal{U}_{\sim c}$, or operators derived from them is a *past formula.*

The *primed version* of a past formula p is defined to be a past formula p' such that p holds in the next state of a computation iff p' holds in the current state. For instance:

$$
\begin{array}{rcl}
(\ominus_{=0}\, p)' & = & p \wedge \neg Tick \\
(\ominus_{=1}\, p)' & = & p \wedge Tick \\
(p\, \mathcal{S}_{\leq c+1}\, q)' & = & q' \vee (p' \wedge Tick \wedge p\, \mathcal{S}_{\leq c}\, q) \\
 & & \vee\, (p' \wedge \neg Tick \wedge p\, \mathcal{S}_{\leq c+1}\, q)
\end{array}
$$

The formula *tick* is defined as follows:

$$tick: \quad \bigcirc_{=1}\, true$$

Thus *tick* holds at position j in a computation exactly when a tick step is taken, i.e., $T_{j+1} = T_j + 1$.

4 Characterizing the Environment

The environment transition of a timed transition module allows the environment to arbitrarily modify any variable not in V^p, i.e., any variable that is not a private variable of M. It is typically the case, however, that the environment of M behaves in a more restrained manner, and by characterizing the behavior of the actual environment it is possible to verify properties of M in a particular environment that do not hold for every environment. Let $taken(\tau_E)$ be a formula that holds at position i of a computation exactly when the environment transition τ_E is taken at position i.

An *environment restriction* Env is a formula of the form:

$$Env: \quad taken(\tau_E) \Rightarrow \underbrace{\bigwedge_{i=1}^{m}(\varphi_i \rightarrow \psi_i')}_{\mathcal{E}}$$

where φ_i and ψ_i are past formulas. The conjunction $\mathcal{E}$ states that if φ_i holds at a position j where an environment step is taken, then ψ_i must hold at the next position $j+1$. In other words, $\mathcal{E}$ constrains the next-state relation that must be satisfied by every environment step.

5 Global Verification

The following axiom is introduced for the new auxiliary variable, i.e., non-system variable, $Tick$:

$$\texttt{TICK}: \quad Tick \Leftrightarrow tick$$

The bounded monotonicity condition for computations of a timed transition module corresponds to the following axiom:

$$\texttt{BMON}: \quad Tick \Rightarrow \bigwedge_{u \in \mathcal{V}} u' = u$$

The upper bound condition corresponds to the following axiom:

$$\texttt{UPPER}: \quad \Box \Diamond_{\leq u^\tau} (\neg En(\tau) \vee first)$$

where the formula $first : \neg \ominus true$ holds precisely at the first state of a computation.

The *verification condition* of a transition τ with respect to precondition p and postcondition q, where p and q are past formulas, is defined as follows:

$$\{p\}\tau\{q\}: \quad \begin{array}{l} p \wedge \tau \wedge \bigwedge_{u \notin V}(u' = u) \\ \quad \wedge \neg Tick \wedge \boxminus_{<l^\tau} En(\tau) \quad \Rightarrow \quad q' \end{array}$$

For a set of transitions $\mathcal{T}$, $\{p\}\mathcal{T}\{q\}$ is taken to be the conjunction of verification conditions $\{p\}\tau\{q\}$ for each $\tau \in \mathcal{T}$.

Given the environment restriction $taken(\tau_E) \Rightarrow \mathcal{E}$, the *environment verification condition* is given by:

$$\{p\}\mathcal{E}\{q\}: \quad \begin{array}{l} p \wedge \mathcal{E} \wedge \bigwedge_{u \in V^p}(u' = u) \wedge \neg Tick \\ \quad \Rightarrow q' \end{array}$$

Finally, the *clock verification condition* is given by:

$$\{p\}clock\{q\}: \quad (p \wedge Tick) \Rightarrow q'$$

Then the following proof rule is used to establish safety properties of a timed transition module M:

R − SAFE

For a past formula φ:

S1. $\Theta \rightarrow \varphi$

S2. $\varphi \Rightarrow p$

S3. $\{\varphi\}\ \mathcal{T}\ \{\varphi\}$

S4. $\{\varphi\}\ \mathcal{E}\ \{\varphi\}$

S5. $\{\varphi\}\ clock\ \{\varphi\}$

$M \models Env \rightarrow \Box p$

Theorem 5.1 *Rule* R-SAFE *is sound and complete for the (global) verification of safety properties of a timed transition module* M*.*

Example: Consider Fischer's mutual exclusion algorithm, presented in Figure 1. Assume uniform time bounds L and U for each statement, except the non-critical regions ℓ_1 and m_1, which both have lower and upper bounds $3L$ and ∞, respectively. Further assume that $2L > U$. Ignoring process M_2 for a moment, it is possible to establish

$$M_1 \models Env \rightarrow \Box \underbrace{at\_\ell_{7,8} \rightarrow x = 1}_{p}$$

where $at\_\ell_{7,8}$, for instance, states that control is either at ℓ_7 or ℓ_8, and the environment assumption Env is given as:

Env : $taken(\tau_E) \Rightarrow \mathcal{E}_1^1 \wedge \mathcal{E}_1^2$

$\mathcal{E}_1^1$: $x' = x \vee x' = 0 \vee x' = 2$

stating that the environment of M_1 either does not change x, or changes x to 0 or 2.

$\mathcal{E}_1^2$: $\boxminus_{\leq U}\ x = 1 \rightarrow x' = 1$

stating that, if $x = 1$ has held through at least the last U time units, then the environment does not change x from 1.

The strengthening assertion φ in rule R-SAFE may be taken to be:

$$\varphi: \quad \begin{array}{l} (at\_\ell_5 \wedge x = 1) \rightarrow (x = 1)\,\mathcal{B}\ at\_\ell_4 \\ \wedge\ (at\_\ell_6 \wedge x = 1) \rightarrow \\ \qquad (x = 1)\,\mathcal{B} \boxminus_{<L} (at\_\ell_5 \wedge x = 1) \\ \wedge\ at\_\ell_{7,8} \rightarrow \boxminus_{<2L}\ x = 1 \end{array}$$

$$
M_1 :: \left[\begin{array}{l} \ell_0\colon\ \textbf{loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\colon\ \textbf{noncritical} \\ \ell_2\colon\ \textbf{while } x \neq 1 \textbf{ do} \\ \quad \left[\begin{array}{l} \ell_3\colon\ \textbf{await } x = 0 \\ \ell_4\colon\ x := 1 \\ \ell_5\colon\ \textbf{skip} \\ \ell_6\colon\ \textbf{if } x = 1 \textbf{ then} \\ \quad \ell_7\colon\ \textbf{critical} \end{array}\right] \\ \ell_8\colon\ x := 0 \end{array}\right] \end{array}\right] \;\|\; M_2 :: \left[\begin{array}{l} m_0\colon\ \textbf{loop forever do} \\ \quad \left[\begin{array}{l} m_1\colon\ \textbf{noncritical} \\ m_2\colon\ \textbf{while } x \neq 2 \textbf{ do} \\ \quad \left[\begin{array}{l} m_3\colon\ \textbf{await } x = 0 \\ m_4\colon\ x := 2 \\ m_5\colon\ \textbf{skip} \\ m_6\colon\ \textbf{if } x = 2 \textbf{ then} \\ \quad m_7\colon\ \textbf{critical} \end{array}\right] \\ m_8\colon\ x := 0 \end{array}\right] \end{array}\right]
$$

Figure 1: Fischer's mutual exclusion algorithm.

Rule `R-SAFE` is complete for proving safety properties. In particular, this includes bounded response formulas of the form $p \Rightarrow \Diamond_{\leq U}\, q$ and minimal separation formulas of the form $p \Rightarrow q\, \mathcal{W}_{>L}\, r$, because, according to the following proposition, every bounded response or minimal separation formula is equivalent to some formula $\Box\, \varphi$, for a past formula φ.

Proposition 5.2

(a) $p \Rightarrow \Diamond_{\leq U}\, q \quad \sim \quad \Box\neg((\neg q)\, \widehat{\mathcal{S}}_{>U}\, (p \wedge \neg q))$

(b) $p \Rightarrow q\, \mathcal{W}_{>L}\, r \quad \sim \quad \Box((\neg q \rightarrow \neg p)\, \mathcal{B}\, (r \wedge \boxminus_{\leq L} \neg p))$

The strict since operator $\widehat{\mathcal{S}}$ in part (a) of the proposition can be defined in terms of $\mathcal{S}$:

$$p\, \widehat{\mathcal{S}}_{>U}\, q \quad \equiv \quad \ominus_{=0}\, (p\, \mathcal{S}_{>U}\, q) \vee \ominus_{=1}\, (p\, \mathcal{S}_{>U-1}\, q)$$

If $\widehat{\mathcal{S}}$ is replaced by $\mathcal{S}$, the proposition fails, as seen by considering the following computation (where $U = 1$ and $q = \ominus\, r$):

$\langle p\colon \mathrm{T}, r\colon \mathrm{F}, T\colon 0\rangle$, $\langle p\colon \mathrm{F}, r\colon \mathrm{F}, T\colon 0\rangle$, $\langle p\colon \mathrm{F}, r\colon \mathrm{F}, T\colon 1\rangle$, $\langle p\colon \mathrm{F}, r\colon \mathrm{T}, T\colon 1\rangle$, $\langle p\colon \mathrm{F}, r\colon \mathrm{T}, T\colon 2\rangle$, ...

Even though, by proposition 5.2, rule `R-SAFE` can be used to establish bounded response properties, it is usually more convenient to use the following rule.

> `R − RESP`
> For past formulas $\varphi[i]$, where $i = 0, \ldots, U$:
>
> R1. $p \;\Rightarrow\; (q \vee \varphi[U])$
> R2. $\{\varphi[i]\}\ \mathcal{T}\ \{q \vee \varphi[i]\}$
> R3. $\{\varphi[i]\}\ \mathcal{E}\ \{\varphi[i]\}$
> R4. $\{\varphi[i+1]\}\ \mathit{clock}\ \{q \vee \varphi[i]\}$
> R5. $\varphi[0] \wedge \mathit{Tick} \;\Rightarrow\; q$
>
> $M \;\vDash\; \mathit{Env} \;\rightarrow (p \Rightarrow \Diamond_{\leq U}\, q)$

Premise R5 may be replaced by

$$\text{R5.} \quad \varphi[i] \;\Rightarrow\; \mathit{En}(\tau_h)$$

to obtain a weaker but often useful rule, where U is then the upper bound of the "helpful transition" $\tau_h \in \mathcal{T}$.

6 Compositional Verification

The following proof rule may be used for compositional verification, where M is obtained as the parallel composition of transition modules $M_1, \ldots, M_N$.

> `PAR`
> For formulas $\varphi_1, \ldots, \varphi_N$ and environment restrictions $\mathit{Env}_1, \ldots, \mathit{Env}_N$:
>
> P1. $M_i \;\vDash\; \mathit{Env}_i \;\rightarrow\; \varphi_i$
> P2. $M_i \;\vDash\; \mathit{Env}_i \;\rightarrow (\mathit{taken}(M_i) \Rightarrow \bigwedge_{j \neq i} \mathcal{E}_j)$
> P3. $\vDash\; \mathcal{E} \rightarrow \bigwedge_i \mathcal{E}_i$
> P4. $\vDash\; (\bigwedge_i \varphi_i) \rightarrow \varphi$
>
> $M \;\vDash\; \mathit{Env} \;\rightarrow\; \varphi$

Rule `PAR` as stated above is not complete. In particular, it is sometimes necessary to introduce auxiliary variables in the specification of each process to represent the hidden state of each process' environment. A stronger version of rule `PAR` can be shown to be complete, where the upper bounds of transitions in M_i are encoded in φ_i [2].

Example: The critical safety property for Fischer's mutual exclusion algorithm is expressed:

$$\mathit{Env} \;\rightarrow\; \Box\neg(\mathit{at\_}\ell_7 \wedge \mathit{at\_}m_7)$$

where the environment restriction Env states that the environment does not modify x. Rule `PAR` may be applied, choosing Env_1 as before, Env_2 chosen symmetrically, and:

$$\begin{array}{ll} \varphi_1: & at\_\ell_{7,8} \Rightarrow x = 1 \\ \varphi_2: & at\_m_{7,8} \Rightarrow x = 2 \end{array}$$

The first premise P1 has been established previously for M_1, and can be established symmetrically for M_2. The third and fourth premises clearly hold. To prove premise P2 for M_2, i.e., that the transitions of M_2 do not violate the environment restriction of M_1, it suffices to consider only the transitions of M_2 that modify x.

For transition τ_{m_4}, first observe that

$$at\_m_4 \Rightarrow \diamondminus_{\leq U} x = 0$$

is valid for M_2. It follows that whenever τ_{m_4} is taken, the antecedent $\boxminus_{\leq U}\ x = 1$ of $\mathcal{E}_1^2$ must be false. Furthermore, τ_{m_4} sets $x = 2$, so $\mathcal{E}_1^1$ is also satisfied. Therefore $\mathcal{E}_1 = \mathcal{E}_1^1 \wedge \mathcal{E}_1^2$ holds whenever τ_{m_4} is taken.

For transition τ_{m_8}, which sets $x = 0$, observe that $at\_m_8 \Rightarrow x = 2$ follows from φ_2, so whenever τ_{m_8} is taken, the antecedent $\boxminus_{\leq U}\ x = 1$ of $\mathcal{E}_1^2$ must be false. Again, $\mathcal{E}_1$ is satisfied whenever τ_{m_8} is taken, establishing premise P2 for M_2. The corresponding condition for M_1, stating that the transitions of M_1 satisfy the environment restriction of M_2, can be established symmetrically.

6.1 Bounded Response for Fischer's Algorithm

The following bounded response property states that process M_1 is assured access to its critical section in $12U$ time units:

$$(at\_\ell_2 \wedge x \neq 1) \Rightarrow \Diamond_{\leq 12U}\ at\_\ell_7$$

A stronger environment restriction Env_1 is needed for M_1 to establish this property, chosen as follows:

$$taken(\tau_{E_1}) \Rightarrow \mathcal{E}_1^1 \wedge \mathcal{E}_1^2 \wedge \mathcal{E}_1^3 \wedge \mathcal{E}_1^4 \wedge \mathcal{E}_1^5$$

where each conjunct $\mathcal{E}_1^i$ is described below:

- $\mathcal{E}_1^1: \quad x' = 0 \vee x' = 2 \vee x' = x$

  states that the environment can change x only to 0 or 2.

- $\mathcal{E}_1^2: \quad \boxminus_{\leq U}\ x = 1 \rightarrow x' = 1$

  states that, if $x = 1$ for at least U time units, then the environment does not change x from 1. This implies that M_1 has the right to enter its critical section.

- $\mathcal{E}_1^3: \quad x = 2 \wedge \diamondminus_{<4L}\ x \neq 2 \rightarrow x' = 2$

  states that the environment does not prematurely withdraw a request for its critical section, i.e., cannot change x from 2 in less than $4L$ time units.

- $\mathcal{E}_1^4: \quad x = 1 \wedge (x = 1)\,\mathcal{S}\,(x = 2) \rightarrow x' = 1$

  states that, if the environment has lost the competition to enter its critical section, characterized by the change from $x = 2$ to $x = 1$, then the environment cannot attempt to enter the critical section again (until x is reset by M_1).

- $\mathcal{E}_1^5: \quad (x = 1)\,\mathcal{B}\,(\diamondminus_{<6L}\ \boxminus_{<4L}\ x = 2) \rightarrow x' = x$

  states that the environment cannot attempt to reenter its critical section if M_1 attempts to enter its critical section within $6L$ time units after the environment last left, as characterized by $x = 2$ having held for at least $4L$ time units.

Figure 2 outlines a proof of the desired bounded response property. Each node, labeled by φ, with thin edges to nodes labeled by $\psi_1, \ldots, \psi_n$, represents the claim that

$$\varphi \Rightarrow \Diamond_{\leq U} \bigvee_{i=1}^{n} \psi_n$$

is valid for M_1 under environment restriction Env_1, which can be proved using rule `R-RESP`.

The thick edge, representing the lemma that

$$\begin{array}{l} at\_\ell_3 \wedge \boxminus_{<2U}\ x = 2 \\ \quad \Rightarrow \Diamond_{\leq 2U}\ (at\_\ell_3 \wedge x = 0 \wedge \diamondminus_{\leq U}\ \boxminus_{<4L}\ x = 2) \end{array}$$

is valid for M under environment restriction Env, can be proved using rule `PAR` where φ_1 is taken to be:

$$\begin{array}{l} at\_\ell_3 \wedge \boxminus_{<2U}\ x = 2 \\ \quad \Rightarrow at\_\ell_3\,\mathcal{W}\,(at\_\ell_3 \wedge x = 0 \wedge \diamondminus_{\leq U}\ \boxminus_{<4L}\ x = 2) \end{array}$$

and φ_2 is taken to be:

$$(\boxminus_{<2U}\ x = 2) \Rightarrow \Diamond_{\leq 2U}\ (x = 0 \wedge \diamondminus_{\leq U}\ \boxminus_{<4L}\ x = 2)$$

Together with the previous claims for M_1, this establishes bounded response for process M_1. Bounded response for M_2 can be established similarly.

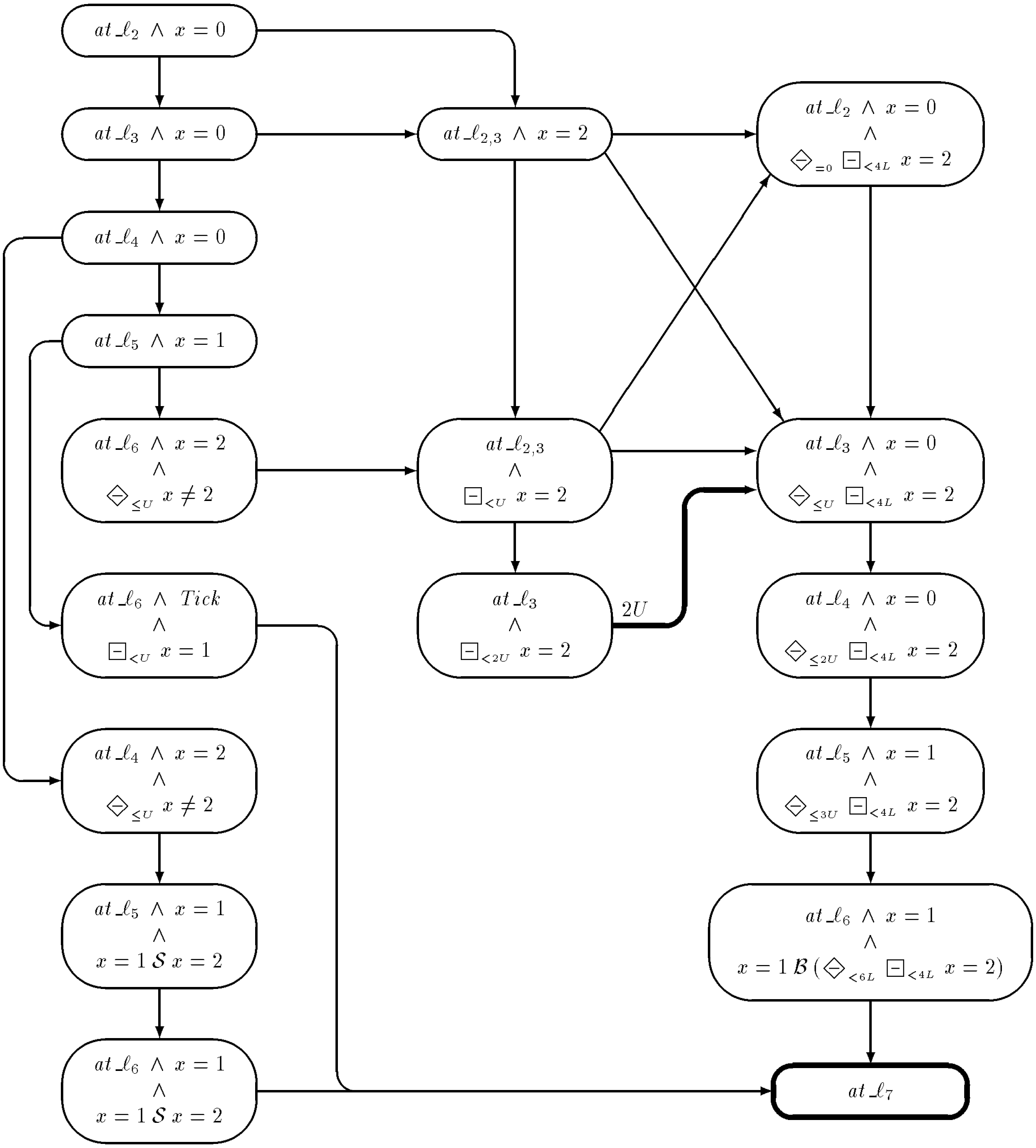

Figure 2: A proof outline.

References

[1] R. Alur and T.A. Henzinger. Real-time logics: Complexity and expressiveness. In *Proc. 5th IEEE Symp. Logic in Comp. Sci.*, pages 390–401, 1990.

[2] E. Chang. *Compositional Verification of Reactive and Real-time Systems.* PhD thesis, Stanford University, 1994.

[3] J. Davies. *Specification and Proof in Real-Time CSP.* Cambridge University Press, 1993.

[4] T. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In W. Kuich, editor, *Proc. 19th Int. Colloq. Aut. Lang. Prog.*, number 623 in Lec. Notes in Comp. Sci, pages 545–558. Springer-Verlag, 1992.

[5] T.A. Henzinger, Z. Manna, and A. Pnueli. Temporal proof methodologies for timed transition systems. *Inf. and Cont.*, 1993.

[6] J. Hooman. *Specification and Compositional Verification of Real-time Systems.* Number 558 in Lec. Notes in Comp. Sci. Springer-Verlag, 1991.

[7] J. Hooman and J. Widom. A temporal-logic-based compositional proof system for real-time message passing. In E. Odijk, M. Rem, and J.-C. Syre, editors, *PARLE 89: Parallel Architectures and Languages Europe*, number 366 in Lec. Notes in Comp. Sci, pages 424–441. Springer-Verlag, 1989.

[8] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time Systems*, 2(4):255–299, 1990.

[9] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems.* Springer-Verlag, 1994.