In *Foundations of Software Technology and Theoretical Computer Science*,
LNCS, Bangalore, India, Dec. 1995.

Generalized Temporal Verification Diagrams*

I.A. Browne, Z. Manna, H.B. Sipma

Computer Science Department
Stanford University, Stanford, CA 94305
e-mail: {anca,manna,sipma}@cs.stanford.edu

Abstract. Verification diagrams are a succinct and intuitive way of representing proofs that reactive systems satisfy a given temporal property. We present a generalized verification diagram that allows representation of a proof of any property expressible by a temporal formula. We show that representation of a proof by generalized verification diagram is sound and complete.

1 Introduction

Verification diagrams are a succinct and intuitive way of representing proofs that reactive systems satisfy a given temporal property. Verification diagrams were first introduced for this purpose in [MP94] (see also [MP95]). In that paper, diagrams dedicated to particular classes of properties were presented, e.g., WAIT-FOR diagrams for precedence properties, CHAIN and RANKED diagrams for response properties. In this paper we present generalized verification diagrams, which allow the representation of a proof of any property expressible by a temporal formula.

The method of proof by verification diagram, called here *diagram verification*, is based on the representation of reactive systems by fair transition systems [MP91], summarized in Section 2, and on the representation of the specification by a temporal formula or a formula automaton (ω-automaton), explained in Section 3. A verification diagram, described in Section 4, is a formula automaton with additional components. The languages accepted by both formula automata and verification diagrams are sets of infinite sequences of states.

For a given reactive system P and temporal formula φ, diagram verification is accomplished by constructing a verification diagram (VD) Ψ and showing that Ψ faithfully represents all computations of the corresponding fair transition system (FTS) Φ and that it "satisfies" φ. To show that a VD Ψ satisfies a temporal formula φ we must show that the set of sequences accepted by Ψ, denoted by $\mathcal{L}(\Psi)$, is a subset of the set of sequences satisfying φ, denoted by $\mathcal{L}(\varphi)$, i.e., we must show

$$\mathcal{L}(\Psi) \subseteq \mathcal{L}(\varphi).$$

* This research was supported in part by the National Science Foundation under grant CCR-92-23226, by the Defense Advanced Research Projects Agency under NASA grant NAG2-892, by the United States Air Force Office of Scientific Research under grant F49620-93-1-0139, and by Department of the Army under grant DAAH04-95-1-0317.

A set of first-order verification conditions associated with the VD Ψ establishes that the set of computations of Φ, denoted by $\mathcal{L}(\Phi)$, is a subset of $\mathcal{L}(\Psi)$, i.e.,

$$\mathcal{L}(\Phi) \subseteq \mathcal{L}(\Psi).$$

Diagram verification consists of establishing the validity of a set of assertions (first-order formulas), and checking decidable graph-theoretical problems.

In Section 5 we show that if an FTS Φ satisfies a formula φ then there always exists a VD Ψ such that

$$\mathcal{L}(\Phi) \subseteq \mathcal{L}(\Psi) \subseteq \mathcal{L}(\varphi),$$

i.e., diagram verification is shown to be complete.

Throughout the paper the concepts are illustrated with a simple example.

In this paper, we present the results and proofs using temporal formulas as our specification language. The same results hold also for the larger class of properties that are specifiable by formula automata. However there is a difference in the complexity of the verification process.

2 Computational Model: Fair Transition Systems

The computational model used for reactive systems is that of a *fair transition system* (FTS), $\Phi = \langle V, \Theta, \mathcal{T}, \mathcal{J}, \mathcal{C} \rangle$, where V is a finite set of variables, Θ is an initial assertion, $\mathcal{T}$ is a finite set of transitions, $\mathcal{J} \subseteq \mathcal{T}$ contains the just (weakly fair) transitions and $\mathcal{C} \subseteq \mathcal{T}$ contains the compassionate (strongly fair) transitions. A *state* s is an interpretation of V, and Σ denotes the set of all states. A transition $\tau \in \mathcal{T}$ is a function $\tau : \ \Sigma \mapsto 2^{\Sigma}$, and each state in $\tau(s)$ is called a τ-successor of s. Each transition τ is represented by a transition relation $\rho_\tau(s, s')$, an assertion that expresses the relation between the values of V in s and the values of V (referred to by V') in any of its τ-successors s'. The enabledness of a transition τ is expressed by $En(\tau) : \ \exists s' \ . \ \rho_\tau(s, s')$.

A run of Φ is an infinite sequence of states such that the first state satisfies Θ and any two consecutive states satisfy a ρ_τ for some $\tau \in \mathcal{T}$. A computation of Φ is a run of Φ with the additional property that for each $\tau \in \mathcal{J}$ ($\tau \in \mathcal{C}$), it is not the case that τ is continually enabled beyond some point (infinitely many times enabled) but taken only finitely many times. The set of all runs of Φ is denoted by $\mathcal{L}_R(\Phi)$ and the set of all computations of Φ is denoted by $\mathcal{L}(\Phi)$.

Example Consider the program MTX (MAY-TERMINATE-X) shown in Figure 1. The **idle** statement does not modify any data variables so its only observable effect is when it terminates. It is not required to terminate however, which is represented in the transition system by excluding the transition associated with the **idle** statement from the justice and compassion set. The **request** and **release** statements are usually associated with a semaphore variable. Execution of **request** y decrements y by 1; it can only be executed if $y > 0$. Execution of **release** y increments y by 1.

The FTS Φ associated with this program is shown in Figure 2. The boolean variables ℓ_i and m_i indicate if control currently resides at the corresponding program location. The formula $pres(U)$, with $U \subseteq V$, states that the variables in U are not modified by the transition, i.e., $pres(U) : \ \bigwedge_{u \in U} (u' = u)$. ∎

$$
\begin{array}{c}
\textbf{local } \; x : \textbf{integer where } x \geq 0 \\
\textbf{local } \; y : \textbf{integer where } y = 1
\end{array}
$$

$$
P_1 :: \left[\begin{array}{l}
\ell_0\text{: } \textbf{loop forever do} \\
\quad \left[\begin{array}{l}
\ell_1\text{: } \textbf{idle} \\
\ell_2\text{: } \textbf{request } y \\
\ell_3\text{: } \textbf{idle} \\
\ell_4\text{: } \textbf{release } y
\end{array}\right]
\end{array}\right]
\quad || \quad
P_2 :: \left[\begin{array}{l}
m_0\text{: } \textbf{while } x > 0 \textbf{ do} \\
\quad \left[\begin{array}{l}
m_1\text{: } \textbf{request } y \\
m_2\text{: } \textbf{release } y \\
m_3\text{: } x := x - 1
\end{array}\right] \\
m_4\text{:}
\end{array}\right]
$$

Fig. 1. Program MTX

$$
\begin{array}{l}
V = \{\ell_0, \ell_1, \ell_2, \ell_3, \ell_4\} \cup \{m_0, m_1, m_2, m_3, m_4\} \cup \{x, y\} \\
\Theta = \ell_0 \wedge m_0 \wedge x \geq 0 \wedge y = 1 \\
\mathcal{T} = \{\tau_{\ell_i} \mid i \in [0..4]\} \cup \{\tau_{m_i} \mid i \in [0..4]\} \cup \{\tau_I\} \text{ where} \\
\quad \rho_{\tau_{\ell_0}} = \ell_0 \wedge \neg \ell_0' \wedge \ell_1' \wedge pres(V - \{\ell_0, \ell_1\}) \\
\quad \rho_{\tau_{\ell_1}} = \ell_1 \wedge \neg \ell_1' \wedge \ell_2' \wedge pres(V - \{\ell_1, \ell_2\}) \\
\quad \rho_{\tau_{\ell_2}} = \ell_2 \wedge \neg \ell_2' \wedge \ell_3' \wedge y > 0 \wedge y' = y - 1 \wedge pres(V - \{\ell_2, \ell_3, y\}) \\
\quad \rho_{\tau_{\ell_3}} = \ell_3 \wedge \neg \ell_3' \wedge \ell_4' \wedge pres(V - \{\ell_3, \ell_4\}) \\
\quad \rho_{\tau_{\ell_4}} = \ell_4 \wedge \neg \ell_4' \wedge \ell_0' \wedge y' = y + 1 \wedge pres(V - \{\ell_0, \ell_4, y\}) \\
\quad \rho_{\tau_{m_0}} = m_0 \wedge \neg m_0' \wedge ((m_1' \wedge x > 0 \wedge m_4' = m_4) \vee \\
\qquad\qquad\qquad\qquad\qquad (m_4' \wedge x \leq 0 \wedge m_1' = m_1)) \\
\qquad\qquad \wedge\, pres(V - \{m_1, m_2, m_4\}) \\
\quad \rho_{\tau_{m_1}} = m_1 \wedge \neg m_1' \wedge m_2' \wedge y > 0 \wedge y' = y - 1 \\
\qquad\qquad \wedge\, pres(V - \{m_1, m_2, y\}) \\
\quad \rho_{\tau_{m_2}} = m_2 \wedge \neg m_2' \wedge m_3' \wedge y' = y + 1 \wedge pres(V - \{m_2, m_3, y\}) \\
\quad \rho_{\tau_{m_3}} = m_3 \wedge \neg m_3' \wedge m_0' \wedge x' = x - 1 \wedge pres(V - \{m_0, m_3, x\}) \\
\quad \rho_{\tau_I} = pres(V) \\
\mathcal{J} = \{\tau_{\ell_0}, \tau_{\ell_4}, \tau_{m_0}, \tau_{m_2}, \tau_{m_3}\} \\
\mathcal{C} = \{\tau_{\ell_2}, \tau_{m_1}\}
\end{array}
$$

Fig. 2. FTS Φ for program MAY-TERMINATE-X(MTX)

3 Specification Language

3.1 Temporal Logic

We use *linear-time temporal logic* as our specification language for reactive systems. A *temporal formula* is constructed out of assertions and the usual boolean and temporal operators.

An FTS Φ *satisfies* a temporal formula φ if all its computations satisfy φ.

Example Consider again the program MTX. The property that process P_2 will eventually terminate can be expressed by the temporal formula $\diamondsuit\, m_4$, i.e., eventually the boolean variable m_4 will be true. Unfortunately, the program does not satisfy this property as the process P_2 might get stuck at location m_1 while process P_1 is idle at location ℓ_3. A weaker formula that excludes this case is

$$\varphi : \ \Box \Diamond(\neg \ell_3) \rightarrow \Diamond m_4,$$

which expresses the property that if P_1 will not stay idle forever at location ℓ_3 then P_2 will eventually terminate. ▟

3.2 Formula Automata

A formula automaton (FA) is an ω-automaton (for a survey see [Tho90]) with Streett-like acceptance conditions on edges. The set of models of a property expressed by an FA is the set of state sequences that are accepted by the ω-automaton. It has been proven that any property expressible by a temporal formula can be expressed as an FA, however formula automata are strictly more expressive than quantifier-free temporal logic ([MW84]).

An FA $\mathcal{A} = \langle N, N_0, E, \mu, \mathcal{F}\rangle$ over a set of variables V has the components

- N: A finite set of nodes.
- $N_0 \subseteq N$: A set of initial nodes.
- $E \subseteq N \times N$: A set of edges connecting nodes.
- $\mu : N \mapsto F(V)$: A node-labeling function, where $F(V)$ denotes the set of all assertions over V.
- $\mathcal{F} \subseteq 2^{E \times E}$: An edge acceptance condition given by an acceptance list $\{(P_1, R_1), \ldots, (P_m, R_m)\}$, where $P_j \subseteq E$ are called the *persistent* edges, and $R_j \subseteq E$ are called the *recurrent* edges.

We say an infinite sequence of nodes $\pi : \ n_0, n_1, \ldots$ is a *path* of $\mathcal{A}$ if $n_0 \in N_0$, and for each $i \geq 0$, $(n_i, n_{i+1}) \in E$. We say that $e = (n_i, n_{i+1})$ occurs in π. For a path π,

- $inf_n(\pi)$ stands for $\{n \in N \mid n \text{ occurs infinitely often in } \pi\}$
- $inf_e(\pi)$ stands for $\{e \in E \mid e \text{ occurs infinitely often in } \pi\}$

We say a path π is *accepting* if it satisfies

$$\forall j \in [1..m] \,.\, inf_e(\pi) \subseteq P_j \ \vee \ (inf_e(\pi) \cap R_j \neq \emptyset),$$

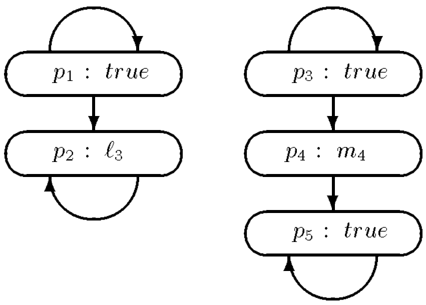
i.e., for each pair (P_j, R_j), some edge in R_j occurs infinitely often in π or all edges that occur infinitely often in π are in P_j.

For any infinite sequence of states $\sigma : \ s_0, s_1, \ldots$ we say that a path of $\mathcal{A}$, $\pi : \ n_0, n_1, \ldots,$ is a *trail* of σ in $\mathcal{A}$ if for all $i \geq 0$, $s_i \models \mu(n_i)$, i.e., every state s_i in the sequence satisfies the assertion associated with n_i. The trail of a finite sequence of states is defined similarly.

An infinite sequence of states σ is a *computation* of an FA $\mathcal{A}$ if it has an accepting trail in $\mathcal{A}$. The set of computations of $\mathcal{A}$ is denoted by $\mathcal{L}(\mathcal{A})$. We say that $\mathcal{A}$ is an FA of a temporal formula φ if

$$\mathcal{L}(\mathcal{A}) = \{\sigma \mid \sigma \models \varphi\}.$$

If $n \in N$ then $next(n)$ stands for $\{n_1 \in N \mid (n, n_1) \in E\}$, that is, $next(n)$ is the set of all successor nodes of n. For $N_1 \subseteq N$, $\mu(N_1)$ stands for $\bigvee_{n \in N_1} \mu(n)$.



initial nodes:

$N_0 = \{p_1, p_2, p_3, p_4\}$

acceptance list:

$\mathcal{F} = \{(P_1, R_1)\}$ where

$P_1 = \{(p_2, p_2), (p_5, p_5)\}$

$R_1 = \emptyset$

Fig. 3. Formula Automaton for $\varphi : \diamondsuit \square \ell_3 \ \vee \ \diamondsuit m_4$

Example An FA for $\varphi : \ \square \diamondsuit(\neg \ell_3) \rightarrow \diamondsuit m_4$, which can be rewritten as

$$\varphi : \ \diamondsuit \square \ell_3 \ \vee \ \diamondsuit m_4$$

is shown in Figure 3. It is straightforward to see that the subgraphs defined by p_1 and p_2 represent all sequences that satisfy the first disjunct, and the subgraph for p_3, p_4 and p_5 represent all sequences that satisfy the second disjunct.

We define $SF(\varphi)$ to be the set of all the atomic subformulas of φ. For example,

$$SF(\diamondsuit \square(\neg \ell_3) \rightarrow \diamondsuit m_4) \ = \ \{\ell_3, m_4\}.$$

A *φ-propositional state*, denoted by s_φ, assigns a truth-value $s_\varphi[q]$ to each $q \in SF(\varphi)$. We denote by Σ_φ the set of all such states. Clearly Σ_φ is finite with size $2^{|SF(\varphi)|}$. A *φ-propositional model* is an infinite sequence of φ-propositional states. *Satisfaction* of a formula φ over a φ-propositional model $\sigma_\varphi : s_{\varphi,0}, s_{\varphi,1}, \ldots$ is defined inductively, as before, where for an atomic subformula q, $s_\varphi \models q$ iff $s_\varphi[q] = \mathrm{T}$.

We define the propositional language of φ, denoted by $\mathcal{L}^p(\varphi)$, to be

$$\mathcal{L}^p(\varphi) = \{\sigma_\varphi \mid \sigma_\varphi \models \varphi\}.$$

The *propositional projection* of a state $s \in \Sigma$ is a φ-propositional state, denoted by s^p_φ , such that for every $q \in SF(\varphi)$, $s^p_\varphi \models q$ iff $s \models q$. Similarly, we say a φ-propositional model $\sigma^p_\varphi : s^p_{\varphi,0}, s^p_{\varphi,1}, \ldots$, is the *propositional projection* of a model $\sigma : s_0, s_1, \ldots$, if for every $i \geq 0$, $s^p_{\varphi,i}$ is the propositional projection of s_i.

If σ^p_φ is a propositional projection of σ then

$$\sigma^p_\varphi \in \mathcal{L}^p(\varphi) \quad \text{iff} \quad \sigma \in \mathcal{L}(\varphi).$$

Example A model of $\varphi : \ \square(x > 5) \vee \diamondsuit(y \neq 3)$ with atomic subformulas $p : \ x > 5$ and $q : \ y \neq 3$ is $\langle x : 7, y : 3\rangle, \langle x : 6, y : 3\rangle, \langle x : 7, y : 0\rangle, \ldots \ \in \ \mathcal{L}(\varphi)$ with propositional projection $\langle p : \mathrm{T}, q : \mathrm{F}\rangle, \langle p : \mathrm{T}, q : \mathrm{F}\rangle, \langle p : \mathrm{T}, q : \mathrm{T}\rangle, \ldots \ \in \mathcal{L}^p(\varphi)$.

4 Verification Diagrams

A verification diagram (VD) Ψ approximates the set of computations of an FTS Φ. It can be viewed as an FA with three additional components: an edge-labeling function to represent fairness, a set of ranking functions to justify the acceptance conditions, and a mapping relating the nodes to the formula φ to be proven.

For an FTS Φ, a (Φ, φ)-valid VD Ψ, viewed as an FA, has trails for all runs of Φ. The conditions associated with the ranking functions ensure that these trails are accepting trails. The fairness conditions associated with the edge-labeling function ensure that every computation of Φ has a fair trail in Ψ. These conditions together ensure that $\mathcal{L}(\Phi) \subseteq \mathcal{L}(\Psi)$.

The conditions associated with the mapping function provide the other necessary inclusion, namely $\mathcal{L}(\Psi) \subseteq \mathcal{L}(\varphi)$.

4.1 Definition

We define a *(generalized) verification diagram* $\Psi = \langle N, N_0, E, \mu, \eta, \mathcal{F}, \Delta, f \rangle$ over an FTS Φ and formula φ to be a labeled directed graph with N, N_0, E, μ and $\mathcal{F}$ defined as in the FA, and η, Δ and f defined as follows:

- $\eta : E \mapsto 2^{\mathcal{T}}$: An edge-labeling function. Each edge is labeled by a set of transitions of Φ.
- $\Delta \subseteq \{\delta \mid \delta : \Sigma \mapsto \mathcal{D}\}$: A set of ranking functions, mapping states to elements of a well-founded domain $\mathcal{D}$. For each pair (P_j, R_j) of the acceptance list, and for each node $n \in N$, Δ contains a ranking function $\delta_{j,n}$.
- $f : N \mapsto PF(SF(\varphi))$. A mapping from nodes to propositional formulas over the atomic subformulas of φ.

The definitions and notation introduced in Section 3 for the FA apply to the VD. Here we define some additional notions related to the three extra components of the VD.

If $n \in N$, $\tau \in \mathcal{T}$ then $\tau(n)$ stands for $\{n_1 \in next(n) \mid \tau \in \eta(n, n_1)\}$, i.e., $\tau(n)$ is the set of all successor nodes of n that are reachable via an edge labeled by τ.

We say an infinite path π is *fair* if it satisfies

1. $\forall \tau \in \mathcal{J} : \quad (\forall n \in inf_n(\pi))\ (\tau(n) \neq \emptyset) \quad \rightarrow \quad (\exists e \in inf_e(\pi).\tau \in \eta(e))$ and
2. $\forall \tau \in \mathcal{C} : \quad (\exists n \in inf_n(\pi))\ (\tau(n) \neq \emptyset) \quad \rightarrow \quad (\exists e \in inf_e(\pi).\tau \in \eta(e))$.

The first condition says that if a node on which a compassionate transition τ is enabled is visited infinitely many times, then there must be at least one edge labeled by τ that appears infinitely often as well, i.e., τ must be taken infinitely often. The second condition says that if a just transition τ is enabled on all nodes that appear infinitely often, then τ must be taken infinitely often.

An infinite sequence of states σ is a *run* of Ψ if there exists an accepting trail of σ in Ψ. An infinite sequence of states σ is a *computation* of Ψ if there exists a fair and accepting trail of σ in Ψ. The *languages* defined by Ψ are the following:

$$\mathcal{L}_R(\Psi) = \{\sigma \mid \sigma \text{ is a run of } \Psi\}$$

$$\mathcal{L}(\Psi) = \{\sigma \mid \sigma \text{ is a computation of } \Psi\}.$$

A computation $\sigma : s_{\varphi,0}, s_{\varphi,1}, \ldots$ is a φ-propositional model of Ψ if there exists a fair and accepting path $\pi : n_0, n_1, \ldots$ in Ψ such that for all $i \geq 0$

$$s_{\varphi,i} \models f(n_i).$$

The set of all φ-propositional models of a VD Ψ is denoted by $\mathcal{L}^p(\Psi)$.

4.2 Verification Conditions

To show that a VD Ψ faithfully represents all computations of an FTS Φ, and that all its computations satisfy the property φ, the following verification conditions associated with Ψ must be established:

- *Initiation*: At least one initial node satisfies the initial condition of Φ:

$$\Theta \rightarrow \mu(N_0).$$

- *Consecution*: Any τ-successor of a state satisfying $\mu(n)$ satisfies the label of some successor node of n: For every node $n \in N$, and every transition $\tau \in \mathcal{T}$,

$$\mu(n)(s) \ \wedge\ \rho_\tau(s, s') \ \rightarrow\ \mu(next(n))(s')$$

- *Acceptance*: For each of the pairs (P_j, R_j) of the acceptance list and for any $e = (n_1, n_2) \in E$ and $\tau \in \mathcal{T}$, when taking τ from an arbitrary state s, if $e \notin R_j$ then δ_j does not increase, and if $e \notin (P_j \cup R_j)$ then δ_j decreases:
  (A1) If $e \in P_j - R_j$ then

$$\rho_\tau(s, s') \ \wedge\ \mu(n_1)(s) \ \wedge \mu(n_2)(s') \ \rightarrow\ \delta_{j,n_1}(s) \succeq \delta_{j,n_2}(s').$$

  (A2) If $e \in \overline{R_j \cup P_j}$ then

$$\rho_\tau(s, s') \ \wedge\ \mu(n_1)(s) \ \wedge\ \mu(n_2)(s') \ \rightarrow\ \delta_{j,n_1}(s) \succ \delta_{j,n_2}(s').$$

  Note that, due to the presence of the idling transition, this requirement implies that self-loops must be accepting, i.e.,

$$\text{if } (n, n) \in E \text{ then } (n, n) \in P_j \cup R_j.$$

  This is in accordance with our earlier statement that every run has an accepting trail: a finite computation padded with idling transitions is a run.
- *Fairness*: For each $e = (n_1, n_2) \in E$ and $\tau \in \eta(e)$
  (F1) τ is guaranteed to be enabled:

$$\mu(n_1)(s) \rightarrow En(\tau).$$

  (F2) Any τ-successor of a state satisfying $\mu(n_1)$ satisfies the label of some node in $\tau(n_1)$:

$$\mu(n_1)(s) \ \wedge\ \rho_\tau(s, s') \ \rightarrow\ \mu(\tau(n_1))(s').$$

(F3) If τ is compassionate, then, for any other node in the same connected component with n_1, either τ is guaranteed to be disabled or τ labels some outgoing edge from n_1: if $\tau \in \mathcal{C}$, $n_3 \in N$ is such that $\tau(n_3) = \emptyset$ and n_3 is in the maximal strongly connected component (MSCS) that contains n_1, then $\mu(n_3)(s) \rightarrow \neg En(\tau)$.

– *Satisfaction:*

(S1) for all $n \in N$, if $s \models \mu(n)$ then $s^p_\varphi \models f(n)$, and

(S2) $\mathcal{L}^p(\Psi) \subseteq \mathcal{L}^p(\varphi)$

Definition 1 A verification diagram Ψ over an FTS Φ is (Φ,φ)-*valid* if all the verification conditions associated with Ψ are Φ-valid.

Lemma 1. If Ψ is a (Φ,φ)-valid verification diagram then (i) $\mathcal{L}_R(\Phi) \subseteq \mathcal{L}_R(\Psi)$, (ii) $\mathcal{L}(\Phi) \subseteq \mathcal{L}(\Psi)$, and (iii) $\mathcal{L}(\Psi) \subseteq \mathcal{L}(\varphi)$.

Proof:

(i) Consider an arbitrary run $\sigma: s_0, s_1, \ldots$ of the FTS Φ. A straightforward induction proof shows that the *Initiation* and the *Consecution* conditions imply that for every finite prefix $s_0, \ldots, s_k$ of σ, there exists a finite path $n_0, \ldots, n_k$ of the VD such that for all $0 \le i \le k$, $s_i \models \mu(n_i)$. Therefore σ has a trail π in Ψ.

To show that π is accepting, suppose to the contrary that there exists $j \in [1\ldots m]$ such that $\mathit{inf}_e(\pi) \cap R_j = \emptyset$ and $\mathit{inf}_e(\pi) \not\subseteq P_j$. Then there exists $\ell \ge 0$ such that for all $i \ge \ell$, $(n_i, n_{i+1}) \in \overline{R_j}$, and there exist infinitely many i's such that $(n_i, n_{i+1}) \in \overline{P_j \cup R_j}$. But then there exists, by the *Acceptance* conditions, an infinite sequence $\delta_{j,n_\ell} \succeq \delta_{j,n_{\ell+1}} \succeq \ldots$ that is strictly decreasing infinitely many times, contradicting the well-foundedness of $\mathcal{D}$.

(ii) We have to show that if σ is a computation then σ has a fair accepting trail in Ψ. By (i), σ has an accepting trail in Ψ, and the proof of (i) shows that any trail of a prefix of σ can be completed to a trail of σ.

Let π be a trail of σ such that for any other trail π' of σ, if $\mathit{inf}_n(\pi')$ can be reached from $\mathit{inf}_n(\pi)$ then $\mathit{inf}_n(\pi')$ is in the same connected component as $\mathit{inf}_n(\pi)$. We show how to construct another trail of σ that is fair (under (i) we have already shown that any trail of σ is accepting). Let $k \ge 0$ be such that for any $i \ge k$, $n_i \in \mathit{inf}_n(\pi)$. We want to show that for any $l \ge k$, there exists $m \ge l$ and a finite trail $n_0, \ldots, n_k, n'_{k+1}, \ldots, n'_l, \ldots, n'_m$ of σ, such that for any $\tau \in \mathcal{J}$ ($\tau \in \mathcal{C}$), $n_l, \ldots, n'_m$ gratifies τ; i.e., the segment is fair: it contains an edge labeled by τ, or $\tau(n) = \emptyset$ holds for some (for every) node n in the segment.

We first gratify the transitions in $\mathcal{J}$. Notice that once a transition is gratified on a segment, it will stay gratified on any segment that includes that segment. Suppose that we have already gratified some transitions in the segment $n'_l, \ldots, n'_i$ and τ has not been considered yet. If $\tau(n'_i) = \emptyset$ then τ is gratified; if $\tau(n'_i) \neq \emptyset$ then, by F2, it must be the case, that there exist a node n'_{i+i} such that $\tau \in \eta(n'_i, n'_{i+1})$ and $s_{i+1} \models n'_{i+1}$.

Next, we gratify the transitions in $\mathcal{C}$ such that they remain gratified on any longer segment. We have two cases:

- In the original trail π, for every $n \in \mathit{inf}_n(\pi)$ we have $\tau(n) = \emptyset$. Then any other trail of σ that starts with $n_0, \ldots, n_k$ has the same property. Indeed, let π' be a trail of σ of the form $\pi' : \ n_0, \ldots, n_k, n'_{k+1}, \ldots$. Then $\mathit{inf}_n(\pi')$ is reachable from $\mathit{inf}_n(\pi)$ and therefore it must be in the same connected component. This implies, by F3, that there is no node n'_i with $i \geq k$ such that $\tau(n') \neq \emptyset$. Therefore any segment of any trail of σ that starts with $n_0, \ldots, n_k$ gratifies τ.
- In the original trail π, there exists a node $n \in \mathit{inf}_n(\pi)$ such that $\tau(n) \neq \emptyset$. Thus, by F1, τ is infinitely often enabled on σ. As any trail π' of the form $\pi' : \ n_0, \ldots, n_k, n'_{k+1}, \ldots$ stays in the same connected component as $\mathit{inf}_n(\pi)$ it follows, by F3, that any time we take a τ transition on σ after k, it must be the case that the respective node has an outgoing edge labeled τ. Then, by F2, we must be able to take that edge and thus gratify τ.

It follows, by induction, that we can construct a trail $\pi' : \ n_0, \ldots, n_{k_1}, \ldots, n_{k_i}, \ldots, n_{k_{i+1}-1}, n_{k_{i+1}}, \ldots$ of σ such that each segment $n_{k_i}, \ldots, n_{k_{i+1}-1}$ gratifies all the transitions in $\mathcal{C} \cup \mathcal{J}$. Therefore π' is fair.

(iii) $\sigma \in \mathcal{L}(\Psi)$ implies, by S1, $\sigma^p_\varphi \in \mathcal{L}^p$ which implies, by S2, $\sigma^p_\varphi \in \mathcal{L}^p(\varphi)$ and therefore $\sigma \in \mathcal{L}(\varphi)$. □

Proposition 2. (Soundness) Let Φ be an FTS and φ a temporal formula. If there exists a (Φ, φ)-valid VD then Φ satisfies φ.

Proof: It follows directly from lemma 1 that $\mathcal{L}(\Phi) \subseteq \mathcal{L}(\varphi)$. □

Example Consider the FTS Φ (Figure 2) of program MTX. A VD Ψ that summarizes the proof of the property

$$\varphi : \ \Box \Diamond (\neg \ell_3) \rightarrow \Diamond m_4$$

over Φ is shown in Figure 4. We sketch a proof that Ψ is (Φ, φ)-valid.

We need to check the five conditions mentioned before.

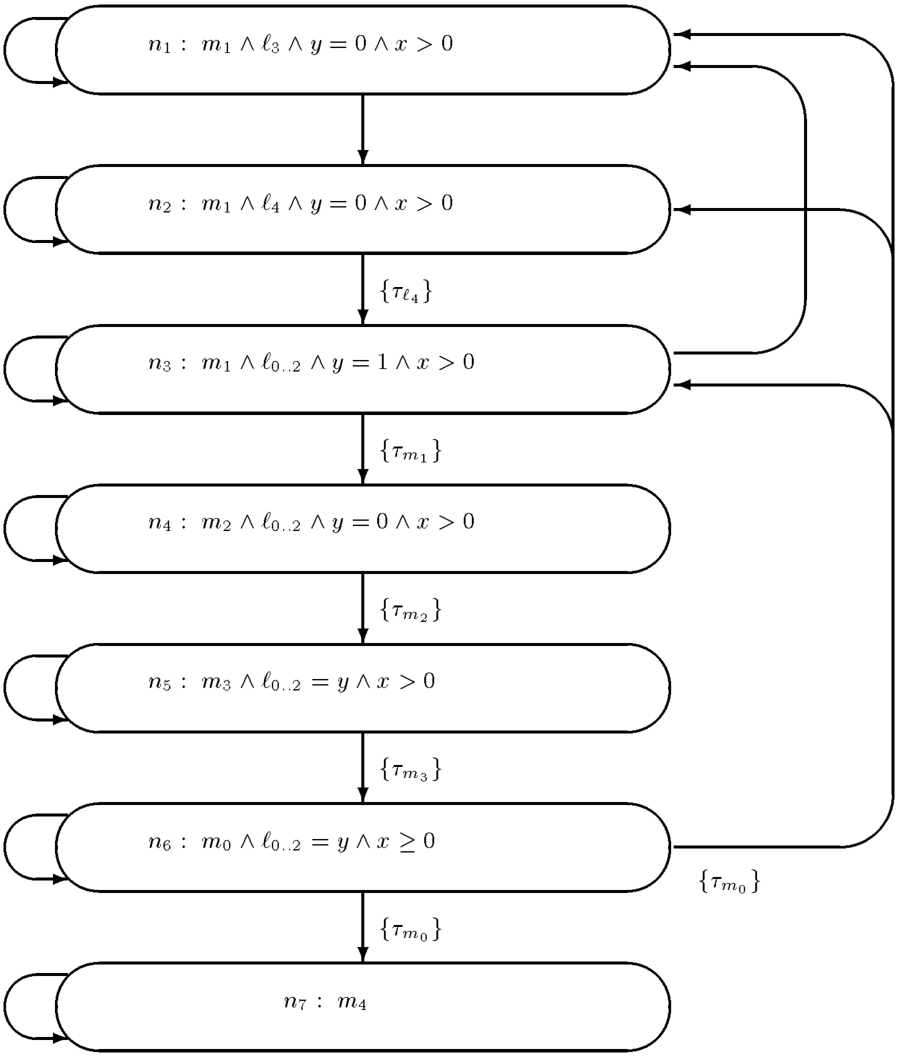
- *Initiation*:

$$\underbrace{\ell_0 \ \wedge \ m_0 \ \wedge \ x \geq 0 \ \wedge \ y = 1}_{\Theta} \ \rightarrow \ \underbrace{m_0 \ \wedge \ \ell_{0..2} = y \ \wedge \ x \geq 0}_{\mu(n_6)}$$

Thus the initiation condition holds.
- *Consecution*: We have to check the consecution condition for all nodes and all transitions. As an example we show the verification condition for node n_2. The verification conditions for the other nodes are similar.
The only transitions enabled in n_2 are τ_I and τ_{ℓ_4}, since m_1 is disabled due to $y = 0$. Obviously ρ_{τ_I} preserves $\mu(n_2)$ and for τ_{ℓ_4} we have

$$\underbrace{m_1 \wedge \ell_4 \wedge y = 0 \wedge x > 0}_{\mu(n_2)(s)} \ \wedge \ \underbrace{\ell_4 \wedge \neg \ell'_4 \wedge \ell'_0 \wedge y' = y + 1 \wedge pres(\overline{\{\ell_0, \ell_4, y\}})}_{\rho_{\tau_{\ell_4}}}$$
$$\rightarrow \ \underbrace{\ldots}_{\mu(n_2)(s')} \ \vee \ \underbrace{m'_1 \wedge \ell'_{0\ldots2} \wedge y' = 1 \wedge x' > 0}_{\mu(n_3)(s')}$$

which is obviously valid.



$N_0 = \{n_6\}$
$\mathcal{F} = \{(P_1, R_1)\}$, where $P_1 = E - \{(n_5, n_6)\}$ and $R_1 = \emptyset$
$\delta_1 : \{n_1, \dots, n_7\} \times \Sigma \to \aleph$

$$\delta_1(n, s) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$f : \begin{cases} f(n_1) = \{\ell_3\} \\ f(n_2) = f(n_3) = f(n_4) = f(n_5) = f(n_6) = \emptyset \\ f(n_7) = \{m_4\} \end{cases}$$

Fig. 4. VD for program MTX and property $\square \Diamond (\neg \ell_3) \to \Diamond m_4$

- *Fairness:* The fairness condition requires us to check for node n_2 that

(F1) $\underbrace{m_1 \wedge \ell_4 \wedge y = 0 \wedge x > 0}_{\mu(n_2)} \;\rightarrow\; \underbrace{\ell_4}_{En(\ell_4)}$ and

(F2) $\underbrace{m_1 \wedge \ell_4 \wedge y = 0 \wedge x > 0}_{\mu(n_2)(s)} \wedge \underbrace{\ell_4 \wedge \neg\ell_4' \wedge \ell_0' \wedge y' = y + 1 \wedge pres(\overline{\{\ell_0, \ell_4, y\}})}_{\rho_{\tau_{\ell_4}}}$

$\rightarrow \underbrace{m_1' \wedge \ell_{0\ldots2}' \wedge y' = 1 \wedge x' > 0}_{\mu(n_3)(s')}$

which are easily seen to be valid.
The fairness verification conditions associated with the nodes n_4, n_5, and n_6 can be checked similarly.
For the fairness verification condition associated with node n_3 and transition m_1 we also have to check condition F3, since $m_1 \in \mathcal{C}$. The nodes in the same MSCS as n_3 are $\{n_1, n_2, n_4, n_5, n_6\}$. It is straightforward to check that on none of these nodes m_1 is enabled.
- *Acceptance*: Notice that m_3 is the only transition that modifies x. Therefore A1 holds. For A2 we have $\overline{P_1 \cup R_1} = \{(n_5, n_6)\}$, and the only transition that leads from n_5 to n_6 is m_3, which decrements x, so A2 also holds.
- *Satisfaction*: We have to show that conditions S1 and S2 hold. Obviously condition S1 holds for f.
It is clear from Figure 3 that

$$\mathcal{L}^p(\varphi) = \langle -, - \rangle^* \langle \mathrm{T}, - \rangle^\omega \;\cup\; \langle -, - \rangle^* \langle -, \mathrm{T} \rangle \langle -, - \rangle^\omega$$

where $\langle v_1, v_2 \rangle$ interprets $\langle \ell_3, m_4 \rangle$ and $-$ denotes a don't-care value.
The propositional language accepted by Ψ is

$$\mathcal{L}^p(\Psi) = \underbrace{\langle -, - \rangle}_{n_1 \ldots n_6}{}^{*} \underbrace{\langle \mathrm{T}, - \rangle}_{n_1}{}^{\omega} \;\cup\; \underbrace{\langle -, - \rangle}_{n_1 \ldots n_6}{}^{*} \underbrace{\langle -, \mathrm{T} \rangle}_{n_7}{}^{\omega}$$

which is a subset of $\mathcal{L}^p(\varphi)$.
Note that the strongly connected components $\{n_1, n_2, n_3\}$ and $\{n_2\}$, $\{n_3\}$, $\{n_4\}$, $\{n_5\}$ and $\{n_6\}$ are excluded from the infinity set by the fairness requirement: e.g., in $\{n_1, n_2, n_3\}$ the transition $m_1 \in \mathcal{C}$ is infinitely often enabled but never taken. The strongly connected components $\{n_1, n_2, n_3, n_4, n_5, n_6\}$, $\{n_2, n_3, n_4, n_5, n_6\}$ and $\{n_3, n_4, n_5, n_6\}$ are excluded by the acceptance condition, which states that edge (n_5, n_6) cannot occur infinitely often. ■

5 Completeness

A verification diagram is a complete representation of a proof that a reactive system satisfies a temporal formula. The following definitions and results are needed for the proof.

Definitions. An FA $\mathcal{A}$ is *deterministic* if any sequence of states σ has at most one trail of σ in $\mathcal{A}$, i.e., for any $n_1, n_2 \in N$, if $n_1, n_2 \in N_0$ or for some $n_3 \in N$, $n_1, n_2 \in next(n_3)$ then $\neg(\mu(n_1) \wedge \mu(n_2))$. An FA $\mathcal{A}$ is *complete* if any sequence of states σ has at least one trail of σ in $\mathcal{A}$, i.e., $\mu(N_0) \wedge \bigwedge_{n \in N} \mu(next(n))$ is valid.

Lemma 3. Let Φ be an FTS and $\mathcal{A}$ an FA. Then there exists a mapping from nodes to assertions $acc : N \mapsto F(V)$ such that $s \models acc(n)$ iff there exists a computation segment $s_0(\models\Theta), \ldots, s_k{=}s$ of Φ and a path segment $n_0(\in N_0), \ldots, n_k{=}n$ of $\mathcal{A}$ such that for any $i \in [0..k]$, $s_i \models \mu(n_i)$.

Justification. Such an assertion can always be constructed in an assertion language that is sufficiently expressive to encode finite sequences.

Theorem 4. [LPS81] Let $\sqsupset$ be a well-founded ordering over a set S. Then there exists a function into the ordinals, $\delta : S \to \mathcal{O}$ such that

(W1) $s \sqsupset s'$ implies $\delta(s) > \delta(s')$.
(W2) If $s \sqsupset s''$ implies $s' \sqsupset s''$ for every $s'' \in S$ then $\delta(s) \leq \delta(s')$.

Lemma 5. Let Φ be an FTS and $\mathcal{A}$ a deterministic FA with acceptance list $\{(P_1, R_1), \ldots, (P_m, R_m)\}$ such that $\mathcal{L}_R(\Phi) \subseteq \mathcal{L}(\mathcal{A})$ and for every node n in $\mathcal{A}$ and $s \models \mu(n)$, there exists a computation segment $s_0(\models\Theta), \ldots, s_k{=}s$ of Φ and a path $n_0(\in N_0), \ldots, n_k{=}n$ of $\mathcal{A}$ such that for any $i \in [0..k]$, $s_i \models \mu(n_i)$

Then there exist well-founded domains $(\mathcal{D}_j)_{j\in[1..m]}$ and functions $(\delta_j : N \times \Sigma \to \mathcal{D}_j)_{j\in[1..m]}$ such that for any edge $e = (n, n')$ in $\mathcal{A}$ and for any $\tau \in \mathcal{T}$
(a) if $e \in P_j - R_j$ then $\rho_\tau(s, s') \wedge \mu(n)(s) \wedge \mu(n')(s') \;\to\; \delta_j(n, s) \geq \delta_j(n', s')$
(b) if $e \in \overline{R_j \cup P_j}$ then $\rho_\tau(s, s') \wedge \mu(n)(s) \wedge \mu(n')(s') \;\to\; \delta_j(n, s) > \delta_j(n', s')$.

Proof. Let $\sqsupset_j$ be the order on $N \times \Sigma(V)$ given by $(n, s) \sqsupset_j (n', s')$ iff there is an "unfair" computation leading from (n, s) to (n', s'), that is there exists a computation segment $s_0 = s, \ldots, s_k = s'$ of Φ and a path segment $n_0 = n, \ldots, n_k = n'$ in $\mathcal{A}$ such that $s_i \models \mu(n_i)$, $\{(n_0, n_1), (n_1, n_2) \ldots, (n_{k-1}, n_k)\} \not\subseteq P_j$ and $\{(n_0, n_1), (n_1, n_2), \ldots, (n_{k-1}, n_k)\} \cap R_j = \emptyset$.
$\sqsupseteq_j$ **is a partial order**. It is clearly reflexive and transitive. In order to show that it is also antisymmetric, suppose that $(n, s) \sqsupseteq_j (n', s')$, $(n', s') \sqsupseteq_j (n, s)$ and $(n, s) \neq (n', s')$. There exists a path segment $n_0(\in N_0), \ldots, n_k = n$ and a computation segment $s_0(\models\Theta), \ldots s_k = s$ such that for all $i \in [0..k]$, $s_i \models \mu(n_i)$. As $(n, s) \sqsupset_j (n', s')$, there exists a path segment $n_k, \ldots, n_l{=}n'$ of $\mathcal{A}$ and a computation segment $s_k, \ldots, s_l{=}s'$ of Φ such that for all $i \in [k..l]$, $s_i \models \mu(n_i)$. Similarly there exists a path segment $n_l, \ldots, n_m = n$ of $\mathcal{A}$ and a computation segment $s_l, \ldots, s_m{=}s$ of Φ such that for all $i \in [l..m]$, $s_i \models \mu(n_i)$. But then we have a run $s_0, \ldots, (s_k, \ldots, s_l, \ldots, s_{m-1})^\omega$ of Φ, and a path $n_0, \ldots, (n_k, \ldots, n_l, \ldots, n_{m-1})^\omega$ in $\mathcal{A}$ that is not accepting, such that for any $i \in [0..m-1]$, $s_i \models n_i$. This contradicts the assumption that every run of Φ is accepted by $\mathcal{A}$. Thus it must be the case that $(n, s) = (n', s')$.
$\sqsupset_j$ **is well-founded**. Indeed, suppose that $(n_1, s_1) \sqsupset_i (n_2, s_2) \sqsupset_j \ldots$ Then, as before, we could construct a run of Φ (that goes through $s_1, s_2, \ldots$) that is not a computation of $\mathcal{A}$.

By Theorem 4, there exists a function $\delta_j : S \to \mathcal{O}$ that satisfies W1 and W2. We have to show that δ_j satisfies (a) and (b). Let $n, n' \in N$ and $\tau \in \mathcal{T}$.

(a) Suppose $(n, n') \in P_j - R_j$ and $\rho_\tau(s, s') \wedge \mu(n)(s) \wedge \mu(n)(s')$. For any (n'', s''), if $(n', s') \sqsupset_j (n'', s'')$ then also $(n, s) \sqsupset_j (n'', s'')$. Indeed, suppose that

$(n', s') \sqsupset_j (n'', s'')$. Then there exists a path segment $n_1 = n', \ldots, n_k = n''$ of $\mathcal{A}$ and a computation segment $s_1 = s', \ldots, s_k = s''$ of Φ such that for any $i \in [1..k]$, $s_i \models \mu(n_i)$. It follows that $n, n_1, \ldots, n_k$ is a path segment of $\mathcal{A}$, $s, s_1, \ldots, s_k$ is a computation segment of Φ, $s \models \mu(n)$, and for any $i \in [1..k]$, $s_i \models \mu(n_i)$. Clearly, $\{(n, n_1), (n_1, n_2), \ldots, (n_{k-1}, n_k)\}$ is not accepted by (P_j, R_j) and thus we have shown that $(n, s) \sqsupset_j (n', s')$. This implies, by W2, that $\delta(n, s) \geq \delta(n', s')$.

(b) Suppose $(n, n') \in \overline{R_j \cup P_j}$ and $\rho_\tau(s, s') \wedge \mu(n)(s) \wedge \mu(n)(s')$. Then $(n, s) \sqsupset_j (n', s')$ which implies, by W1, that $\delta(n, s) > \delta(n' s')$. □

Proposition 6. (Completeness) Let Φ be an FTS that satisfies a temporal formula φ. Then there exists a (Φ, φ)-valid VD Ψ. Moreover all the verification conditions of Ψ are valid.

Proof. (1) *Earmarked transitions.* Let $(\tau_1, \ldots, \tau_m)$ be an ordering of all the transitions in $\mathcal{T}$ an let $t \notin V$ be a variable that ranges over $\{1, \ldots, m\}$. Then we can construct an FTS $\Phi^t = \langle V \cup \{t\}, \Theta, \mathcal{T}^t, \mathcal{C}^t, \mathcal{J}^t \rangle$ obtained from Φ by "earmarking" each transition with a different value for t. More precisely, $\mathcal{T}^t = \{\tau_i^t \mid \rho_{\tau_i^t} = \rho_{\tau_i} \wedge t' = i\}$, $\mathcal{C}^t = \{\tau_i^t \mid \tau_i \in \mathcal{C}\}$ and $\mathcal{J}^t = \{\tau_i^t \mid \tau_i \in \mathcal{J}\}$. Φ^t is used to construct an FA $\mathcal{A}_p$ that accepts all the runs in Φ^t and an edge-labeling function that exactly translates the fairness of Φ^t. The variable t is then eliminated to obtain an FA $\mathcal{A}_f$ that accepts all the runs of Φ and an edge-labeling function with all the desired properties. The ranking functions are defined on a different FA $\mathcal{A}_r$ that accepts all the runs of Φ and also admits ranking functions satisfying the *Acceptance* conditions. Finally, the two automata are combined into a VD.

(2) *Characterization of fairness in* Ψ^t. Let $\mathcal{A}_{\mathcal{T}^t}$ be the FA with nodes $N^t = N_0^t = 2^{\mathcal{T}^t} \times \mathcal{T}^t$, edges $E^t = \{((T, \tau), (T', \tau')) \mid \tau' \in T\}$, node-labeling function defined by $\mu^t(T, \tau_i^t) = \bigwedge_{\tau \in T} En(\tau) \wedge \bigwedge_{\tau \notin T} \neg En(\tau) \wedge t = i$ and the fairness condition $\mathcal{F}^t = \{(R_\tau, P_\tau) \mid \tau \in \mathcal{C}^t \cup \mathcal{J}^t\}$ where

$$\text{if } \tau \in \mathcal{J}^t \text{ then } \begin{cases} P_\tau = \emptyset \\ R_\tau = \{((T_1, \tau_1), (T_2, \tau_2)) \mid \tau \notin T_1\} \cup \{((T_1, \tau_1), (T_2, \tau))\} \end{cases}$$

$$\text{if } \tau \in \mathcal{C}^t \text{ then } \begin{cases} P_\tau = \{((T_1, \tau_1), (T_2, \tau_2)) \mid \tau \notin T_1\} \\ R_\tau = \{((T_1, \tau_1), (T_2, \tau))\} \end{cases}$$

Let η^t be an edge-labeling defined by $\eta^t((T, \tau), (T', \tau')) = \begin{cases} \tau' & \text{if } \tau' \in \mathcal{C}^t \cup \mathcal{J}^t \\ \emptyset & \text{otherwise} \end{cases}$

Then $\mathcal{A}_{\mathcal{T}^t}$ has the following properties: it is deterministic; any run of Φ^t has a trail in $\mathcal{A}_{\mathcal{T}^t}$; a run of Φ^t is a computation of Φ^t iff its trail in $\mathcal{A}_{\mathcal{T}^t}$ is η^t-fair; a path in $\mathcal{A}_{\mathcal{T}^t}$ is η^t-fair iff it is accepting; the *Fairness* conditions hold.

(3) *Underlying* FA *for a candidate* VD. Any computation of such an FA, has to be a model of φ and any run of Φ is either a computation of this FA or it is non-fair (in which case it is eliminated by an edge-labeling function). A first step towards achieving these conditions is to consider a deterministic FA $\mathcal{A}_u$, such that $\mathcal{L}(\mathcal{A}_u) = \mathcal{L}(\varphi) \cup \overline{\mathcal{L}(\mathcal{A}_{\mathcal{T}^t})}$. ($\mathcal{A}_u$ can be constructed from $\mathcal{A}_{\mathcal{T}^t}$ and an FA for φ by standard ω-automata constructions.) Let $\mathcal{A}_s$ be the FA obtained from $\mathcal{A}_u$ by strengthening its labeling to the labeling given by Lemma 3.

Then $\mathcal{A}_s$ has the following properties: $\mathcal{L}_R(\Phi^t) = \mathcal{L}(\mathcal{A}_s)$; $\mu^s(n)(s)$ iff there exists a computation segment $s_0(\models \Theta^t), s_1, \ldots, s_k = s$ of Φ^t and a path segment

$n_0(\in N_0), n_1, \ldots, n_k{=}n$ of $\mathcal{A}_s$ such that $\forall i \in [0..k].\ s_i \models \mu^s(n_i)$; the *Consecution* condition holds. We can always assume that for any node n there exists a constant t_n such that $\mu(n) \rightarrow t = t_n$;

(4) *Edge-labeling.* In order to be able to use the edge-labeling on $\mathcal{A}_{\mathcal{T}^t}$ we take $\mathcal{A}_p$ to be the cross product of $\mathcal{A}_s$ and $\mathcal{A}_{\mathcal{T}^t}$ with the accepting condition induced by $\mathcal{A}_s$. The edge-labeling function on $\mathcal{A}_p$ is defined to be the one induced by η^t.

Then $\mathcal{A}_p$ has the following properties: it is deterministic; the *Fairness* conditions hold; $\mathcal{L}_R(\Phi^t) \subseteq \mathcal{L}(\mathcal{A}_p)$; for any node n there exists a constant t_n such that $\mu(n) \rightarrow t = t_n$; $\mathcal{L}(\Phi^t) \subseteq \{\sigma \in \mathcal{L}(\mathcal{A}_p) \mid \sigma$ has a fair and accepting trail in $\mathcal{A}_p\}$; $\{\sigma \in \mathcal{L}(\mathcal{A}_p) \mid \sigma$ has a fair and accepting trail in $\mathcal{A}_p\} \subseteq \mathcal{L}(\varphi)$.

(5) *Removal of t.* Let $\mathcal{A}_f$ be the FA over V with the same structure as $\mathcal{A}_p$ but with the node-labeling function defined by $\mu^f(n) = \mu^p(n)[t/t_n]$ where t_n is such that $\mu^p(n) \rightarrow t = t_n$. Let η^f be the function induced by η^p. Then $\mathcal{A}_f$ has the following properties: $\mathcal{L}_R(\Phi) \subseteq \mathcal{L}(\mathcal{A}_f)$; the *Consecution* and *Fairness* conditions hold; $\mathcal{L}(\Phi) \subseteq \{\sigma \in \mathcal{L}(\mathcal{A}_f) \mid \sigma$ has a fair and accepting trail in $\mathcal{A}_f\}$; $\{\sigma \in \mathcal{L}(\mathcal{A}_f) \mid \sigma$ has a fair and accepting trail in $\mathcal{A}_f\} \subseteq \mathcal{L}(\varphi)$.

(6) *Ranking functions.* Let $\mathcal{A}_{df}$ be a deterministic FA that accepts the same language as $\mathcal{A}_f$ and let $\mathcal{A}_r$ be an FA that has the same structure as $\mathcal{A}_{df}$ except for the labeling function, which is given by the formula *acc* from Lemma 3. Then $\mathcal{A}_r$ has the following properties: $\mathcal{L}_R(\Phi) = \mathcal{L}(\mathcal{A}_r)$; $\mu^r(n)(s)$ iff there exists a computation segment $s_0(\models \Theta), s_1, \ldots, s_k = s$ of Φ and a path segment $n_0(\in N_0), n_1, \ldots, n_k{=}n$ of $\mathcal{A}_r$ such that $\forall i \in [0..k].\ s_i \models \mu^r(n_i)$; the *Consecution* and *Acceptance* conditions hold.

(7) *The verification diagram.* Let Ψ be the VD obtained by taking the cross product of $\mathcal{A}_f$ and $\mathcal{A}_r$ with the edge-labeling function induced by η_f, the ranking functions induced by $\delta_{n,i}$ defined under (6) and the mapping function defined by $f(n) = \bigwedge_{q \in SF(\varphi)\,:\,\mu(n) \rightarrow q} q \ \wedge\ \bigwedge_{q \in SF(\varphi)\,:\,\mu(n) \rightarrow \neg q} \neg q$. We can always assume that the labels of the nodes of Ψ are satisfiable (otherwise we can remove them).

Ψ is (Φ,φ)-valid. *Initiation:* Both $\mathcal{A}_f$ and $\mathcal{A}_r$ accept any run of Φ and therefore the product has a trail for any run in Φ, which implies that any initial node has to satisfy the label of some initial node of Ψ. *Consecution* results from the fact that both $\mathcal{A}_f$ and $\mathcal{A}_r$ have this property. *Acceptance* follows from the *Acceptance* condition for $\mathcal{A}_r$. *Fairness* holds because both automata satisfy the *Consecution* condition and $\mathcal{A}_f$ satisfies the *Fairness* conditions. The (S1) *Satisfaction* condition is true by the definition of f. To prove (S2), let $\sigma_\varphi \in \mathcal{L}^p(\Psi)$ and let π be its trail in Ψ. As the label of each node on the trail is satisfiable, there exists a sequence $\sigma \in \mathcal{L}(\Psi)$ with trail π. Then $\sigma \in \mathcal{L}_R(\Phi)$. Let σ^t be a sequence of states over $V \cup \{t\}$ defined by the trail π and σ. Then σ^t is a run of Φ^t with a fair trail in $\mathcal{A}_{\mathcal{T}^t}$. Therefore σ^t is a computation of Φ^t, which implies $\sigma^t \models \varphi$ and as a consequence $\sigma \models \varphi$. This implies, by the definition of f, $\sigma_\varphi \models \varphi$.

6 Discussion

The verification diagram as presented can be used as a proof rule. It reduces the proof of a property over a fair transition system to a set of proofs of first-order validities and regular ω-automata operations, such as language inclusion.

As regards complexity, a proof by verification diagram consists of checking the validity of the *Initiation* condition, $|N| \cdot |\mathcal{T}|$ *Consecution* conditions, $\mathcal{O}(|\mathcal{F}| \cdot |\mathcal{T}| \cdot |E|)$ *Acceptance* conditions, $\mathcal{O}(|\mathcal{T}| \cdot (|E| + |N|)$ *Fairness* conditions, and $\mathcal{O}(|N|)$ *Satisfaction* conditions. In addition, it requires finding the maximal strongly connected components of the verification diagram (F3), and the verification of the inclusion of two propositional languages (S2).

The generalized verification diagrams presented here, generalize the verification diagrams presented in [MP94]. A verification diagram is an instance of a generalized verification diagram, specialized to the property to be proven, that is, in the verification diagram some of the verification conditions of the corresponding generalized verification diagram have been translated into requirements on structure and node labeling, specific to the property.

The advantage of the verification diagram is that, in general, structural requirements are easier to check than the corresponding verification conditions, in particular language inclusion; in addition the predetermined structure provides guidance in constructing the proof. The advantage of the generalized verification diagram is that it is universally applicable to any property specifiable by a formula automaton, whereas the verification diagram is limited to the properties for which the structural requirements have been determined. When automating checking of the diagram, a single algorithm suffices for generalized verification diagrams, whereas verification diagrams require multiple algorithms, one for each property. Generalized verification diagrams can also be used to systematically derive verification diagrams for specific properties.

Acknowledgements

We thank Anuchit Anuchitanukul, Luca de Alfaro and Tomás Uribe for their careful reading of the paper and their helpful suggestions.

References

[LPS81] D. Lehmann, A. Pnueli, and J. Stavi. Impartiality, justice and fairness: The ethics of concurrent termination. In *Proc. 8th Int. Colloq. Aut. Lang. Prog.*, volume 115 of *LNCS*, pages 264–277. Springer-Verlag, 1981.

[MP91] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, New York, 1991.

[MP94] Z. Manna and A. Pnueli. Temporal verification diagrams. In *Proc. Int. Symp. on Theoretical Aspects of Computer Software*, volume 789 of *LNCS*, pages 726–765. Springer-Verlag, 1994.

[MP95] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, New York, 1995.

[MW84] Z. Manna and P. Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Trans. Prog. Lang. Sys.*, 6(1):68–93, Jan. 1984.

[Tho90] Wolfgang Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133–191. Elsevier Science Publishers (North-Holland), 1990.