Statecharts Merging: Mathematical Underpinnings

Shiva Nejati

Department of Computer Science, University of Toronto,
Toronto, ON M5S 3G4, Canada.
Email: {shiva}@cs.toronto.edu

In this article, we prove that the merge algorithm in [6] is behaviour preserving. To do this, we describe a version of the algorithm over Modal Transition Systems (MTSs) and prove it to be behaviour preserving (Section 2).We then show that merges produced by this algorithm are equivalent to parameterized Statecharts merges computed by the original algorithm in [6] (Section 3).

1 Notation

This section provides background and fixes the notation.

Models. We describe our models as MTSs. MTSs are event-based transition systems with two types of transitions.

Definition 1 *[5] A* Modal Transition System (MTS) *is a tuple $M = (S, s_0, R^{must}, R^{may}, Act)$, where S is a finite set of states, $s_0 \in S$ is the initial state, $R^{must} \subseteq S \times Act \times S$ and $R^{may} \subseteq S \times Act \times S$ are respectively must and may transition relations, and Act is a set of transition labels.*

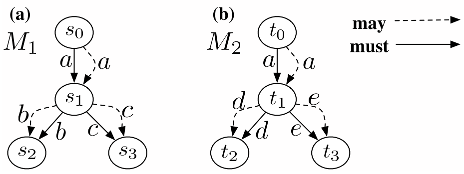

Fig. 1. Two initial MTSs.

Models shown in Figures 1 and 3 are examples of MTSs. Throughout this report, we choose to represent must transitions by solid arrows and may transition by dashed arrows. For both models in Figure 1, the set of must transitions is equal to that of may transitions. We refer to such MTSs as *complete* MTSs.

Temporal Logics. We describe the behaviours of MTSs using the temporal logic L_μ [4].

Definition 2 *[4] Let Var be a set of fixpoint variables, and Act be a set of transition labels. The* logic L_μ *is the set of formulas generated by the following grammar:*

$$\varphi ::= \mathsf{t} \mid Z \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \langle a\rangle\varphi \mid \mu Z \cdot \varphi(Z)$$

where $a \in Act$, $Z \in Var$, and $\varphi(Z)$ is syntactically monotone in Z.

The derived connectives are defined as follows:

$$\begin{array}{ll} \mathsf{f} & = \neg\mathsf{t} \\ \varphi_1 \vee \varphi_2 & = \neg(\neg\varphi_1 \wedge \neg\varphi_2) \\ [a]\varphi & = \neg\langle a\rangle\neg\varphi \\ \nu Z \cdot \varphi(Z) & = \neg\mu Z \cdot \neg\varphi(\neg Z) \end{array}$$

The semantics of L_μ over MTSs is given as follows.

Definition 3 *[3] Let M be an MTS, φ be an L_μ formula, and $e : Var \rightarrow \mathcal{P}(S)$ be an environment. We denote by $||\varphi||^M_\top e$ the set of states in M that satisfy φ, and by $||\varphi||^M_\bot e$ the set of states in M that refute φ. The sets $||\varphi||_\top e$ and $||\varphi||_\bot e$ are defined as follows,*

$$\begin{array}{ll|ll} ||\mathit{true}||_\top e & = \Sigma & ||\mathit{true}||_\bot e & = \emptyset \\ ||Z||_\top e & = e(Z) & ||Z||_\bot e & = \overline{e(Z)} \\ ||\neg\varphi||_\top e & = ||\varphi||_\bot e & ||\neg\varphi||_\bot e & = ||\varphi||_\top e \\ ||\varphi_1 \wedge \varphi_2||_\top e & = ||\varphi_1||_\top e \cap ||\varphi_2||_\top e & ||\varphi_1 \wedge \varphi_2||_\bot e & = ||\varphi_1||_\bot e \cup ||\varphi_2||_\bot e \\ ||\langle a\rangle\varphi||_\top e & = \{s \mid \exists s' \cdot R^{must}(s,a,s') \wedge s' \in ||\varphi||_\top e\} & ||\langle a\rangle\varphi||_\bot e & = \{s \mid \forall s' \cdot R^{may}(s,a,s') \Rightarrow s' \in ||\varphi||_\bot e \\ ||\mu Z \cdot \varphi||_\top e & = \bigcap\{S \subseteq \Sigma \mid ||\varphi||_\top e[Z \rightarrow S] \subseteq S\} & ||\mu Z \cdot \varphi||_\bot e & = \bigcup\{S \subseteq \Sigma \mid S \subseteq ||\varphi||_\bot e[Z \rightarrow S]\} \end{array}$$

4-Valued Semantics of L_μ over MTSs. We denote by **4** the 4-valued Belnap logic [2] with elements t, f, d, and m. The truth ordering $\leq$ of this logic is defined as $\mathsf{f} \leq \mathsf{m} \leq \mathsf{t}$, and $\mathsf{f} \leq \mathsf{d} \leq \mathsf{t}$. Negation is defined as $\neg\mathsf{t} = \mathsf{f}$, $\neg\mathsf{m} = \mathsf{m}$, and $\neg\mathsf{d} = \mathsf{d}$. An additional ordering $\preceq$ relates values based on the amount of information: $\mathsf{m} \preceq \mathsf{t} \preceq \mathsf{d}$ and $\mathsf{m} \preceq \mathsf{f} \preceq \mathsf{d}$, so that m represents the least amount of information and d the most. The logic **4** is shown in Figure 2.

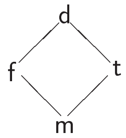

Fig. 2. Belnap logic.

The semantics of L_μ over an MTS M can be described as a 4-valued function $||.||^M : L_\mu \times S \rightarrow \mathbf{4}$. We write

$$||\varphi||^M(s) = \begin{cases} \mathsf{d} & \text{if } s \in ||\varphi||^M_\top \wedge s \in ||\varphi||^M_\bot \\ \mathsf{t} & \text{if } s \in ||\varphi||^M_\top \wedge s \notin ||\varphi||^M_\bot \\ \mathsf{f} & \text{if } s \in ||\varphi||^M_\bot \wedge s \notin ||\varphi||^M_\top \\ \mathsf{m} & \text{if } s \notin ||\varphi||^M_\top \wedge s \notin ||\varphi||^M_\bot \end{cases}$$

The value of φ in M, denoted $||\varphi||^M$ is defined as $||\varphi||^M(s_0)$, where s_0 is the initial state of M.

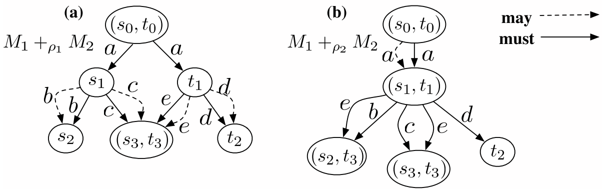

Fig. 3. Two alternative merges of MTSs in Figure 1.

For example, consider the property $\varphi = \langle a \rangle \mathsf{t}$ stating that the initial state has an outgoing transition labelled a. The value of φ is d in the model in Figure 3(a). This is because state (s_0, t_0) has outgoing must transitions labelled a which make φ true, but (s_0, t_0) does not have any outgoing may transition. Hence, the negation of φ, i.e., $[a]\mathsf{f}$, is true as well. That is, the initial state of the model in Figure 3(a) satisfies both φ and $\neg\varphi$, and thus, the value of φ becomes d in that model. The value of $\varphi = \langle a \rangle \mathsf{t}$ is true in the model in Figure 3(b) because its initial state has an outgoing may transition and an outgoing must transition, both labelled a. Thus, this model satisfies φ and refutes its negation.

For MTSs shown in Figure 1, no L_μ property would evaluate to m or d. This is because these MTSs are complete and for such MTSs it can be shown that every L_μ property evaluates either to true or to false [3].

Refinement. MTSs are compared using ordering relations known as *refinements relations* [5].

Definition 4 *[5, 3] Let $M_1 = (S_1, s_0, R_1^{must}, R_1^{may}, Act_1)$ and $M_2 = (S_2, t_0, R_2^{must}, R_2^{may}, Act_2)$ be MTSs. A refinement relation $\preceq \subseteq S_1 \times S_2$ is a relation where $s \preceq t$ iff*

1. $\forall s' \in S_1 \cdot \forall a \in Act_1 \cdot R^{must}(s, a, s') \Rightarrow \exists t' \in S_2 \cdot R^{must}(t, a, t') \wedge s' \preceq t'$,
2. $\forall t' \in S_2 \cdot \forall a \in Act_2 \cdot R^{may}(t, a, t') \Rightarrow \exists s' \in S_1 \cdot R^{may}(s, a, s') \wedge s' \preceq t'$.

We say M_2 refines M_1 and write $M_1 \preceq M_2$, if there is a refinement $\preceq$ such that $s_0 \preceq t_0$, where s_0 and t_0 are the initial states of M_1 and M_2 respectively.

Note that in this report, we use the notation $\preceq$ to denote both refinement ordering over MTSs and knowledge ordering over the elements of the 4-valued logic. It is known that refinement preserves temporal properties [5, 3], i.e., if $M_1 \preceq M_2$ then $\forall \varphi \in L_\mu \cdot ||\varphi||^{M_1} \preceq ||\varphi||^{M_2}$. In Definition 4, we do not require that $\preceq$ be the largest refinement relation. For the largest refinement relation, the following can be proven [5, 3]

$$M_1 \preceq M_2 \Leftrightarrow \forall \varphi \in L_\mu \cdot ||\varphi||^{M_1} \preceq ||\varphi||^{M_2}$$

That is, the largest refinement relation is logically characterized by L_μ. In this report, we only require that a refinement relation be property preserving and not make use of the logical characterization of refinement. Therefore, we do not need a refinement relation to be the largest such relation.

2 Computing Common Refinements

In this section, we describe an algorithm for computing common refinements. The input to this algorithm is a pair of models M_1 and M_2 and a relation $\rho \subseteq S_1 \times S_2$. We assume that ρ includes (s_0, t_0). If not, we add to M_1 and M_2 new initial states s_0' and t_0' with transitions to old ones. We then simply add the tuple (s_0', t_0') to ρ. The output of the algorithm is a merged model denoted $M_1 +_\rho M_2 = (S_\rho, (s_0, t_0), R_\rho^{must}, R_\rho^{may}, Act_1 \cup Act_2)$ and computed as follows:

Set of States (S_ρ): $S_\rho = \rho \cup \{s \mid \nexists t \in S_2 \cdot (s,t) \in \rho\} \cup \{t \mid \nexists s \in S_1 \cdot (s,t) \in \rho\}$
Set of must transitions (R_ρ^{must}): For every $(s, a, s') \in R_1^{must}$ (the case for $(t, a, t') \in R_2^{must}$ is symmetric):
- **a(1)** $\forall t, t' \in S_2 \cdot (s,t) \in \rho \wedge (s', t') \in \rho \Rightarrow ((s,t), a, (s', t')) \in R_\rho^{must}$
- **a(2)** $\nexists t' \in S_2 \cdot (s', t') \in \rho \Rightarrow \forall t \in S_2 \cdot (s,t) \in \rho \Rightarrow ((s,t), a, s') \in R_\rho^{must}$
- **a(3)** $\nexists t \in S_2 \cdot (s,t) \in \rho \Rightarrow \forall t' \in S_2 \cdot (s', t') \in \rho \Rightarrow (s, a, (s', t')) \in R_\rho^{must}$
- **a(4)** $\nexists t \in S_2 \cdot (s,t) \in \rho \wedge \nexists t' \in S_2 \cdot (s', t') \in \rho \Rightarrow (s, a, s') \in R_\rho^{must}$

Set of may transitions (R_ρ^{may}):
- **b(1)** For every $(s, a, s') \in R_1^{may}$ and $(t, a, t') \in R_2^{may}$, if $(s,t) \in \rho \wedge (s', t') \in \rho$, then $((s,t), a, (s', t')) \in R_\rho^{may}$.
- **b(2)** For every state s in M_1 that is not mapped by ρ, and for every transition $(s, a, s') \in R_1^{may}$ (the case for M_2 is symmetric):
  1 $\forall t' \in S_2 \cdot (s', t') \in \rho \Rightarrow (s, a, (s', t')) \in R_\rho^{may}$
  2 $\nexists t' \in S_2 \cdot (s', t') \in \rho \Rightarrow (s, a, s') \in R_\rho^{may}$

Intuitively, the state space of $M_1 +_\rho M_2$ includes all pairs of states in ρ and all states in M_1 and M_2 that are not mapped by ρ. Every must transition in M_1 or M_2 is lifted to corresponding must transitions in $M_1 +_\rho M_2$ (by rules **a(1) through a(4)**). May transitions that are shared between M_1 and M_2 are lifted to may transitions in $M_1 +_\rho M_2$ (by rule **b(1)**) . May transitions that emanate from states unmapped by ρ are also lifted to may transition in $M_1 +_\rho M_2$ (by rule **b(2)**).

Figure 3 illustrates two different merges of models M_1 and M_2 in Figure 1. The relation inducing the merge in Figure 3(a) is $\rho_1 = \{(s_0, t_0), (s_3, t_3)\}$ and the one inducing the merge in Figure 3(b) is $\rho_2 = \{(s_0, t_0), (s_1, t_1), (s_3, t_3), (s_2, t_3)\}$. The following theorem shows that the merge of M_1 and M_2 computed by the above algorithm corresponds to a common refinement of M_1 and M_2.

Theorem 1. *Let M_1 and M_2 be MTSs, and let $\rho \subseteq S_1 \times S_2$. Then,*

1. $M_1 \preceq M_1 +_\rho M_2$
2. $M_2 \preceq M_1 +_\rho M_2$

Proof:
To prove $M_1 \preceq M_1 +_\rho M_2$, we show that the relation

$$\sigma_1 = \{(s,s) \mid s \in S_1 \wedge s \in S_\rho\} \cup \{(s, (s,t)) \mid s \in S_1 \wedge (s,t) \in S_\rho\}$$

is a refinement relation between M_1 and $M_1 +_\rho M_2$.

Before we give the proof, we provide an inductive definition, equivalent to Definition 4, for the refinement relation $\preceq$.

Definition 5 *We define a sequence of refinement relations* $\preceq^0$, $\preceq^1$, ... *on* $S_1 \times S_2$ *as follows:*

- $\preceq^0 = S_1 \times S_2$
- $s \preceq^{n+1} t$ *iff*
  1. $\forall s' \in S_1 \cdot \forall a \in Act_1 \cdot R_1^{must}(s, a, s') \;\Rightarrow\; \exists t' \in S_2 \cdot R_2^{must}(t, a, t') \;\wedge\; s' \preceq^n t'$
  2. $\forall t' \in S_2 \cdot \forall a \in Act_2 \cdot R_2^{may}(t, a, t') \;\Rightarrow\; \exists s' \in S_1 \cdot R_1^{may}(s, a, s') \;\wedge\; s' \preceq^n t'$

  The largest refinement relation is defined as $\bigcap_{i \geq 0} \preceq^i$.

Note that since M_1 and M_2 are finite structures, the sequence $\preceq^0$, $\preceq^1$, ... is finite as well.

To show that σ_1 is a refinement, we prove that σ_1 is a subset of the largest refinement relation, i.e., $\sigma_1 \subseteq \bigcap_{i \geq 0} \preceq^i$. The proof follows by induction on i:

Base case. $\sigma_1 \subseteq \preceq^0$. Follows from the definition of σ_1 and the fact that $\preceq^0 = S_1 \times S_\rho$.
Inductive case. Suppose $\sigma_1 \subseteq \preceq^i$. We prove that $\sigma_1 \subseteq \preceq^{i+1}$.

By Definition 5, we need to show for every $(s, r) \in \sigma_1$,
1. $\forall s' \in S_1 \cdot \forall a \in Act_1 \cdot R_1^{must}(s, a, s') \;\Rightarrow\; \exists r' \in S_\rho \cdot R_\rho^{must}(r, a, r') \;\wedge\; s' \preceq^i r'$
2. $\forall r' \in S_\rho \cdot \forall a \in Act_1 \cup Act_2 \cdot R_\rho^{may}(r, a, r') \;\Rightarrow\; \exists s' \in S_1 \cdot R_1^{may}(s, a, s') \;\wedge\; s' \preceq^i r'$
Here is the proof,

1.

$\forall s' \in S_1 \cdot \forall a \in Act_1 \cdot R_1^{must}(s, a, s')$

$\Rightarrow$ (Case 1: if rule **a(1)** applies)
$\exists t \in S_2 \cdot r = (s, t) \wedge \exists (s', t') \in S_\rho \cdot R_\rho^{must}((s, t), a, (s', t')) \wedge (s', (s', t')) \in \sigma_1$
$\Rightarrow$ (by the inductive hypothesis, and let $r' = (s', t')$)
$\exists r' \in S_\rho \cdot R_\rho^{must}(r, a, r') \wedge s' \preceq^i r'$

$\Rightarrow$ (Case 2: If rule **a(2)** applies)
$\exists t \in S_2 \cdot r = (s, t) \wedge \nexists t' \in S_2 \cdot (s', t') \in \rho \wedge R_\rho^{must}((s, t), a, s')$
$\Rightarrow$ (by definition of S_ρ and σ_1)
$\exists t \in S_2 \cdot r = (s, t) \wedge \exists s' \in S_\rho \cdot R_\rho^{must}((s, t), a, s') \wedge (s', s') \in \sigma_1$
$\Rightarrow$ (by the inductive hypothesis, and let $r' = s'$)
$\exists r' \in S_\rho \cdot R_\rho^{must}(r, a, r') \wedge s' \preceq^i r'$

$\Rightarrow$ (Case 3: If rule **a(3)** applies)
$r = s \wedge \exists t' \in S_2 \cdot (s', t') \in \rho \wedge R_\rho^{must}(s, a, (s', t'))$
$\Rightarrow$ (by definition of S_ρ and σ_1)
$\exists (s', t') \in S_\rho \cdot R_\rho^{must}(s, a, (s', t')) \wedge (s', (s', t')) \in \sigma_1$
$\Rightarrow$ (by the inductive hypothesis, and let $r' = (s', t')$)
$\exists r' \in S_\rho \cdot R_\rho^{must}(r, a, r') \wedge s' \preceq^i (s', t')$

$\Rightarrow$ (Case 4: If rule **a(4)** applies)
$r = s \wedge \nexists t' \in S_2 \cdot (s', t') \in \rho \wedge R_\rho^{must}(s, a, s')$
$\Rightarrow$ (by definition of S_ρ and σ_1)
$\exists s' \in S_\rho \cdot R_\rho^{must}(s, a, s') \wedge (s', s') \in \sigma_1$
$\Rightarrow$ (by the inductive hypothesis, and let $r' = s'$)
$\exists r' \in S_\rho \cdot R_\rho^{must}(r, a, r') \wedge s' \preceq^i r'$

2.

$\forall r' \in S_\rho \cdot \forall a \in Act_1 \cup Act_2 \cdot R_\rho^{may}(r, a, r')$

$\Rightarrow$ (Case 1: If rule **b(1)** applies)
$\exists (s, t), (s', t') \in \rho \cdot r = (s, t) \wedge r' = (s', t') \wedge R_1^{may}(s, a', s')$
$\Rightarrow$ (by definition of ρ and σ_1)
$\exists s' \in S_1 \cdot R_1^{may}(s, a, s') \wedge (s', (s', t')) \in \sigma_1$
$\Rightarrow$ (by the inductive hypothesis)
$\exists s' \in S_1 \cdot R_1^{may}(s, a, s') \wedge s' \preceq^i r'$

$\Rightarrow$ (Case 2: If rule **b(2)-1** applies)
$r = s \wedge \exists (s', t') \in \rho \cdot r' = (s', t') \wedge R_1^{may}(s, a', s')$
$\Rightarrow$ (by definition of ρ and σ_1)
$\exists s' \in S_1 \cdot R_1^{may}(s, a, s') \wedge (s', (s', t')) \in \sigma_1$
$\Rightarrow$ (by the inductive hypothesis)
$\exists s' \in S_1 \cdot R_1^{may}(s, a, s') \wedge s' \preceq^i r'$
$\Rightarrow$ (Case 3: If rule **b(2)-2** applies)
$r = s \wedge \nexists t' \in S_2 \cdot (s', t') \in \rho \wedge R_1^{may}(s, a', s')$
$\Rightarrow$ (by definition of σ_1)
$\exists s' \in S_1 \cdot R_1^{may}(s, a, s') \wedge (s', s') \in \sigma_1$
$\Rightarrow$ (by the inductive hypothesis, and let $r' = s'$)
$\exists s' \in S_1 \cdot R_1^{may}(s, a, s') \wedge s' \preceq^i r'$

The above proves that $\sigma \subseteq \bigcap_{i \geq 0} \preceq^i$. Since σ_1 also relates s_0, i.e., the initial state of M_1, to (s_0, t_0), i.e., the initial state of $M_1 +_\rho M_2$, σ_1 is indeed a refinement relation between M_1 and $M_1 +_\rho M_2$. Note that σ_1 might not be the largest refinement relation, but any refinement relation that includes the initial states of its underlying models can preserve their temporal properties.

To prove $M_2 \preceq M_1 +_\rho M_2$, we show that the relation

$$\sigma_2 = \{(t, t) \mid t \in S_\rho \wedge t \in S_2\} \cup \{(t, (s, t)) \mid t \in S_2 \wedge (s, t) \in S_\rho\}$$

is a refinement relation between M_2 and $M_1 +_\rho M_2$. The proof is symmetric to the above proof. $\square$

It can be seen that the merges in Figure 3 are common refinements of the models in Figure 1. More precisely,

- $M_1 \preceq M_1 +_{\rho_1} M_2$ where the refinement relation is $\{(s_0, (s_0, t_0)), (s_1, s_1), (s_2, s_2), (s_3, (s_3, t_3))\}$.
- $M_2 \preceq M_1 +_{\rho_1} M_2$ where the refinement relation is $\{(t_0, (s_0, t_0)), (t_1, t_1), (t_3, (s_3, t_3)), (t_2, t_2)\}$.
- $M_1 \preceq M_1 +_{\rho_2} M_2$ where the refinement relation is $\{(s_0, (s_0, t_0)), (s_1, (s_1, t_1)), (s_2, (s_2, t_3)), (s_3, (s_3, t_3))\}$.
- $M_2 \preceq M_1 +_{\rho_2} M_2$ where the refinement relation is $\{(t_0, (s_0, t_0)), (t_1, (s_1, t_1)), (t_2, t_2), (t_3, (s_3, t_3)), (t_3, (s_2, t_3))\}$.

3 Parameterization

In this section, we show that: when input MTSs are complete, their merges can be described as parameterized state-machines. A parameterized state-machine is a *labelled transition system (LTS)* whose transitions are labelled by an event and (optionally) a condition. For example, models shown in Figure 4 are parameterized state-machines.

Merges of complete MTSs have an interesting feature which allows us to translate them to parameterized state-machines: For these models, it can be shown that their set of may transitions is a subset of their set of must transitions (for example, see the merges in Figure 3). Intuitively, this is because these merges include *all* the must transitions of the input models, but only *some* of their may transitions.

We now show how these merges can be described as parameterized state-machines. Let $M = (S, s_0, R^{may}, R^{must}, Act)$ be an MTS where $R^{may} \subseteq R^{must}$, and let M'

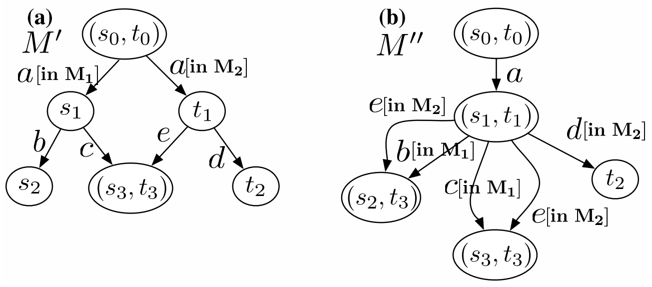

Fig. 4. Parameterized merges of MTSs in Figure 1.

be its corresponding parameterized state-machine that we want to construct. The state-space and the initial state of M' are exactly the same as those of M. Every transition (s, a, s') in $R^{may} \cap R^{must}$ is lifted verbatim to an (unguarded) transition in M'. Every transition (s, a, s') in $R^{must} \setminus R^{may}$ becomes a guarded transition in M' where the guard condition denotes the origin of that transition. If the transition is lifted from M_1 (resp. M_2), the guard condition is $[\mathbf{in\ M_1}]$ (resp. $[\mathbf{in\ M_2}]$). For example, the parameterized versions of merges $M_1 +_{\rho_1} M_2$ and $M_1 +_{\rho_2} M_2$ are respectively M' and M'' shown in Figure 4.

The property preservation results in Section 2 carry over to merges represented as parameterized state-machines. All we need to do is to use parameterized forms of L_μ (see e.g. [1]), instead of $\mathbf{4}$-valued L_μ.

References

1. R. Alur and T. Henzinger. Real-time logics: Complexity and expressiveness. In *LICS*, pages 390–401, 1990.
2. N. Belnap. “A Useful Four-Valued Logic”. In Dunn and Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 30–56. Reidel, 1977.
3. M. Huth, R. Jagadeesan, and D. A. Schmidt. “Modal Transition Systems: A Foundation for Three-Valued Program Analysis”. In *ESOP’01*, volume 2028 of *LNCS*, 2001.
4. D. Kozen. “Results on the Propositional μ-calculus”. *Theoretical Computer Science*, 27:334–354, 1983.
5. K. Larsen and B. Thomsen. “A Modal Process Logic”. In *LICS’88*, 1988.
6. S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave. Matching and merging of statecharts specifications. In *29th International Conference on Software Engineering (ICSE’07)*, 2007. To Appear.