Merging Partial Behaviour Models with Different Vocabularies

Shoham Ben-David[1], Marsha Chechik[2], and Sebastian Uchitel[3]

[1] Univ. of Waterloo, Canada
[2] Univ. of Toronto, Canada
[3] Univ. of Buenos Aires, Argentina

Abstract. Modal transition systems (MTSs) and their variants such as Disjunctive MTSs (DMTSs) have been extensively studied as a formalism for partial behaviour model specification. Their semantics is in terms of implementations, which are fully specified behaviour models in the form of Labelled Transition Systems. A natural operation for these models is that of *merge*, which should yield a partial model which characterizes all common implementations. Merging has been studied for models with the same vocabularies; however, to enable composition of specifications from different viewpoints, merging of models with *different* vocabularies must be supported as well. In this paper, we first prove that DMTSs are not closed under merge for models with different vocabularies. We then define an extension to DMTS called rDMTS, for which we describe a first exact algorithm for merging partial models, provided they satisfy an easily checkable compatibility condition.

1 Introduction

Behaviour models such as Labelled Transition Systems [10] and Statecharts [8] have been extensively studied as a means to formally describe and analyze behaviour of software systems. These models partition the space of behaviours in two, typically interpreted as *required* behaviour and *prohibited* behaviour. Although notions of refinement for these models have also been studied, limitations in terms of expressiveness have been shown to exist when behavior information is incomplete (e.g., [14]).

Partial behaviour models [3,11,12] allow distinguishing between required, possible and prohibited behaviour, hence supporting partial heterogeneous specifications that include existential (e.g., use-cases) and universal (e.g., safety properties) quantification of behaviour [13]. Refinement involves progressively eliminating possible behaviour until all behaviour is either required or prohibited, as in traditional behaviour models. Indeed, the semantics of partial behaviour models is defined in terms of *implementations*, i.e., two-valued models that provide all the required behaviour of the partial model, and any additional exhibited behaviour is defined as possible.

A key operation on partial models is *composition as conjunction* [16]. That is, given two partial models, it is often desirable to compute a new partial model that captures their common implementations. Such an operation, which we refer to as *model merging*, supports independent development of multiple partial viewpoints that cover different aspects of the intended behavior and subsequent composition into a single model that accurately captures all of these viewpoints.

Partial behaviour model merging has been studied extensively for the case where the models to be merged are defined *on the same vocabulary.* Fischbein and Uchitel [7] showed that Modal Transition Systems (MTSs) [11] are not closed under merge, although the set of common implementations can be represented by a finite set of MTSs. A tool for computing such a merge is described in [4]. Benes et al. [1] showed that a variant of MTSs, known as Disjunctive MTSs (DMTSs) [12], is closed under merge and provided a constructive algorithm for computing such a merge for a set of DMTSs.

Yet, often the partial models to be merged do not completely share their vocabularies. This is especially true in the software engineering context, where different viewpoints are expected to have different scopes and hence different vocabularies. Restricting the merge operation to work only for same-vocabulary models hinders the use of partial models in software engineering contexts.

Attempts at merging partial models defined on different vocabularies have mostly been unsuccessful so far. In [2], Chechik et al. examined the possibility of *embedding* for MTS models. Their idea was to embed each of the models to be merged into a common vocabulary, and then use the same-vocabulary merge algorithm on the results. They show that the embedding idea does not work for MTSs. In [6], Fischbein et al. suggested an *approximation* algorithm to merge MTSs defined on different vocabularies. They present several examples showing that their algorithm is incomplete, but do not try to characterize the subset of models for which the algorithm gives correct results.

In this paper, we approach again the problem of merging partial models defined on different vocabularies. We first prove that DMTSs (and thus MTSs as well) are *not* closed under such merge, which explains why previous attempts were not successful. We then introduce a variant of DMTSs, called *restricted* disjunctive modal transition systems (rDMTSs). Using rDMTSs, we provide the first *exact* algorithm for merging partial models (independently of whether they have the same vocabulary or not). While our algorithm is not complete, we are able to characterize the condition under which the algorithm produces the exact merge. In addition to the standard requirement that the partial models to be merged must be consistent (have a common implementation), we also require that they are *compatible*, i.e., all loops of length greater than one in one model must share some vocabulary with the other model.

Our approach is based on embedding [2]. We show how to embed a model M into a larger vocabulary $\mathcal{A}$, resulting in an rDMTS $M^{\mathcal{A}}$ that preserves the set of implementations of M. Thus, when merging models M and N defined on different vocabularies, we first embed each of them in the union of the vocabularies, and then apply an existing DMTS merge algorithm [1] adapted for rDMTSs. We prove that under the consistency and compatibility conditions, our algorithm produces the exact merge.

The rest of the paper is organized as follows. Sec. 2 gives the background on partial behaviour models. Sec. 3 presents a simple example of two consistent MTSs with a single-letter difference in their vocabularies, and proves that no DMTS can represent exactly the set of their common implementations. Sec. 4 and 5 present the main results of the paper – the rDMTS extension and the new merging algorithm, respectively. We summarize our results and discuss future research directions in Sec. 6. The proofs of most theorems are omitted due to space limitations.

2 Preliminaries

Transition Systems. We start with the concept of *Labelled Transition Systems* (LTSs) [10] which are commonly used for modeling concurrent systems.

Definition 1 (LTS [10]). *A* Labeled Transition System *(LTS) is a structure (S, L, δ, s_0), where S is a set of states, L is a set of labels, $\delta \subseteq (S \times L \times S)$ is the transition relation, and $s_0 \in S$ is the initial state.*

Disjunctive Modal Transition Systems (DMTS) [12] are used to specify sets of LTSs. A DMTS distinguishes between two types of transitions – the *possible* and the *disjunctive must.* Transitions that do not appear at all are considered as prohibited. Using a DMTS, one can explicitly model behaviors that are possible in the system and those that the system must exhibit.

Definition 2 (DMTS [12]). *A* Disjunctive Modal Transition System *(DMTS) M is a structure $(S_M, L, \delta^p, \Delta^r, m_0)$, where S is a set of states, L is a set of labels, $\delta^p \subseteq (S_M \times L \times S_M)$ is the* possible *(or* maybe*) transition relation, $\Delta^r \subseteq (S_M \times 2^{L \times S_M})$ is the* disjunctive must *transition relation, and $m_0 \in S_M$ is the initial state.*

Modal Transition Systems (MTSs) [11] are a special case of DMTSs where every disjunctive must has exactly one transition.

We use the notation $m \xrightarrow{\ell}_p m'$ to denote a possible transition $(m, \ell, m') \in \delta^p$ ($s \xrightarrow{\ell} s'$ if the model is an LTS). We use $\langle m, V \rangle$ to denote a disjunctive must transition in Δ^r, where V is a set of pairs $V = \{(l_1, m_1), ..., (l_n, m_n)\}$ with $l_1, ..., l_n \in L$ and $m_1, ..., m_n \in S_M$. A disjunct $(l_i, m_i) \in V$ is sometimes called a *leg*, and the entire disjunctive transition – a *DT.* Legs in a DT can also be self-loops. That is, for a DT $\langle s, V \rangle$, there can be legs $(\ell, m') \in V$ s.t. $m' = m$.

We follow [1] to require also that (1) if $\langle m, V \rangle \in \Delta^r$ then V is not empty, and (2) for all $\langle m, V \rangle \in \Delta^r$ and $(\ell, m') \in V$, we have that $(m, \ell, m') \in \delta^p$. That is, there exists a possible transition for every leg in a DT. Graphically, possible transitions are depicted by a question mark: $m \xrightarrow{\ell?} m'$.

A DMTS specifies a set of LTSs – its *implementations.* An LTS I is considered to be a *strong implementation* of a DMTS M if every transition in I is possible in M, and for every DT in M, at least one leg exists in I.

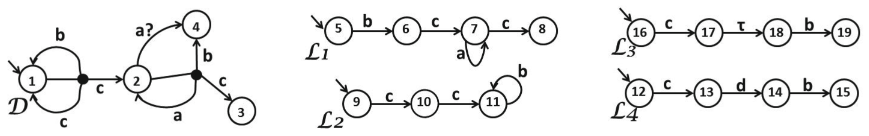

Fig. 1. A DMTS $\mathcal{D}$ and some of its possible implementations. $\mathcal{L}_1$ and $\mathcal{L}_2$ are strong implementations, $\mathcal{L}_3$ is an observational implementation, and $\mathcal{L}_4$ is an alphabet implementation.

Definition 3 (Strong Implementation of DMTSs [12]). *Let $M = (S_M, L, \delta^p_M, \Delta^r_M, m_0)$ be a DMTS and $I = (S_I, L, \delta_I, i_0)$ be an LTS. We say that I* strongly implements *M if (m_0, i_0) is contained in some strong implementation relation $\mathcal{R} \subseteq S_M \times S_I$, s.t. if $(m,i) \in \mathcal{R}$ then (1) $\forall (i \xrightarrow{\ell} i')$, $\exists (m \xrightarrow{\ell}_p m')$ s.t. $(m', i') \in \mathcal{R}$; and (2) $\forall \langle m, V\rangle \in \Delta^r_M$, $\exists (\ell, m') \in V$ and $\exists (i \xrightarrow{\ell} i')$ s.t. $(m', i') \in \mathcal{R}$.*

Example 1. Model $\mathcal{D}$ in Fig. 1 presents a DMTS with one DT on labels b and c and another on a, b and c. Recall that maybe transitions exist for every leg, although they are not explicitly shown. In addition, there is a maybe transition on label a from state 2 to state 4. The LTS $\mathcal{L}_1$ is a strong implementation of $\mathcal{D}$ through the implementation relation $\mathcal{R}_1 = \{(1,5),(1,6),(2,7),(3,8)\}$, and $\mathcal{L}_2$ is also an implementation through $\mathcal{R}_2 = \{(1,9),(1,10),(1,11)\}$.

The set of strong implementations of a DMTS M is denoted by $[[M]]$. Two DMTSs M and N are *consistent* if they have a common implementation, that is, if $[[M]] \cap [[N]] \neq \emptyset$. The *merge* (or "conjunction") of consistent models M and N is a model P s.t. $[[P]] = [[M]] \cap [[N]]$.

Observational and Alphabet Implementations of DMTSs. Hüttel and Larsen [9] were the first to examine MTSs in the presence of *unobservable* labels, denoted by τ, and introduced the notion of *observational* implementations of MTSs. That is, they defined conditions under which an LTS is an implementation of an MTS with unobservable (τ) transitions. Fischbein et al. [5] introduced a more restrictive definition for implementations in the presence of τ's, inspired by branching refinement. This definition was given for MTSs, and we adapt it here to apply to DMTSs. Informally, instead of requiring that a transition from a DT in a DMTS is immediately present in the implementation, as in Def. 3, the observational definition requires that such a transition exists in the implementation, but possibly after a finite (although unbounded) number of τ transitions.

Definition 4 (Observational Implementation of DMTSs). *Let $M = (S_M, L, \delta^p_M, \Delta^r_M, m_0)$ be a DMTS and $I = (S_I, L, \delta_I, i_0)$ be an LTS. We say that I is an* observational implementation *of M if (m_0, i_0) is contained in some* observational implementation relation *$\mathcal{R} \subseteq S_M \times S_I$ for which the following holds for all $(m,i) \in \mathcal{R}$:*

1. $\forall i \xrightarrow{\ell} i'$*, there exists a sequence of possible transitions* $m \xrightarrow{\tau}_p m_1 \xrightarrow{\tau}_p \ldots \xrightarrow{\tau}_p m_j \xrightarrow{\ell}_p m'$*,* $(m_k, i') \in \mathcal{R}$ *for* $1 \leq k \leq j$*, and* $(m', i') \in \mathcal{R}$*.*

2. $\forall (m, V) \in \Delta^r_m$*, there exists a sequence of must transitions* $i \xrightarrow{\tau} i_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} i_j \xrightarrow{\ell} i'$ *s.t.* $(m, i_k) \in \mathcal{R}$ *for* $1 \leq k \leq j$*, and* $\exists(\ell, m') \in V$ *s.t.* $(m', i') \in \mathcal{R}$*.*

In an observational implementation, DMTS and LTS are both defined over the same alphabet, with the addition of the label τ that gets a special treatment. To compare models defined over different alphabets, i.e., to define *observational alphabet implementations*, we follow [15,6], and *hide* labels that are not in the intersection of these alphabets. Hiding is done by replacing such labels by τ's, thus making them unobservable. The resulting models can then be compared using the observational implementation relation (Def. 4).

The definition of hiding in [15] was given for MTSs, and we adapt it to apply to DMTSs, considering every leg of a DT separately.

Definition 5 (Hiding). *Let* $M = (S_M, \alpha M, \delta^p, \Delta^r, m_0)$ *be a DMTS and* X *be a set of labels.* M *with the labels of* X *hidden, denoted* $M \backslash X$*, is a DMTS* $(S_M, \alpha M \backslash X, \delta^{p'}, \Delta^{r'}, m_0)$*, where* $\Delta^{r'}$ *is derived from* Δ^r *by replacing every leg* $(\ell, m') \in V$ *in a DT* $\langle m, V \rangle \in \Delta^r$*, with a leg* (τ, m') *if* $\ell \in X$*. The set* $\delta^{p'}$ *is derived from* δ^p *in the same way, replacing possible transitions* $m \xrightarrow{\ell}_p m'$ *by* $m \xrightarrow{\tau}_p m'$ *if* $\ell \in X$*. For a set of labels* Y*, we use* $M@Y$ *to denote* $M \backslash (\alpha M \backslash Y)$*.*

Definition 6 (Alphabet Implementation of DMTSs). *An LTS* $I = (S_I, \alpha I, \delta_I, i_0)$ *is an* alphabet implementation *of a DMTS* $M = (S_M, \alpha M, \delta^p_M, \Delta^r_M, m_0)$ *if* $\alpha M \subseteq \alpha I$ *and* $I@\alpha M$ *is an observational implementation of* M*.*

Example 2. Consider again DMTS $\mathcal{D}$ in Fig. 1. LTS $\mathcal{L}_3$ is an observational implementation of $\mathcal{D}$, via the relation $\{(1,16),(2,17),(2,18),(4,19)\}$. LTS $\mathcal{L}_4$ is defined on the alphabet $\{a, b, c, d\}$; hiding d results in the model $\mathcal{L}_3$. Thus, $\mathcal{L}_4$ is an alphabet implementation of $\mathcal{D}$.

For a model M with an alphabet αM, let $\mathcal{A}$ be an alphabet such that $\alpha M \subseteq \mathcal{A}$. We denote by $[[M]]_{\mathcal{A}}$ the set of implementations over $\mathcal{A}$ that are alphabet implementations of M.

3 DMTSs Are Not Closed under Alphabet Merge

Our first result explains why the previous attempts to find an alphabet merge algorithm for MTSs have failed. We prove that DMTSs (and thus MTSs as well) are *not closed* under alphabet merge by analyzing the following simple example.

Consider the models in Fig. 2. Model $\mathcal{I}$ has a single must transition labelled c, and we assume its vocabulary is $\{c\}$. We want to merge it with model $\mathcal{J}$ defined over the vocabulary $\{b, c\}$. Thus, we seek a DMTS that specifies exactly all the implementations over the vocabulary $\{b, c\}$ that are common to $\mathcal{I}$ and $\mathcal{J}$. These implementations would be considered "strong" for $\mathcal{J}$, since they share

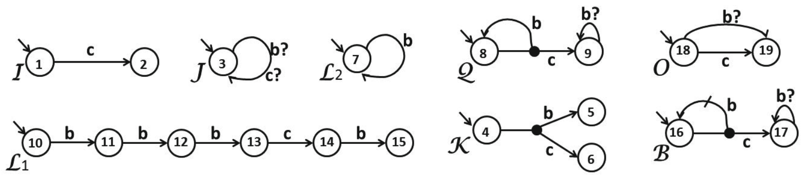

Fig. 2. Models $\mathcal{I}$ and $\mathcal{J}$ do not have a merge in terms of a DMTS. Model $\mathcal{L}_1$ is an example of a common implementation of $\mathcal{I}$ and $\mathcal{J}$. Model $\mathcal{Q}$ is *almost* the merge of $\mathcal{I}$ and $\mathcal{J}$, but the LTS $\mathcal{L}_2$ in an implementation of $\mathcal{Q}$ while not of $\mathcal{I}$. The rDMTS $\mathcal{B}$ is the merge of $\mathcal{I}$ and $\mathcal{J}$. State 4 of model $\mathcal{K}$ is an example of a single-b-allowed state, while state 18 of model $\mathcal{O}$ is not.

the same vocabulary, and "alphabet" for $\mathcal{I}$. The LTS $\mathcal{L}_1$ in Fig. 2 is an example of a possible common implementation: it has a sequence of b transitions followed by one c transition, and then another b transition. It is a strong implementation of $\mathcal{J}$ since $\mathcal{J}$ allows any combination of b's and c's. When all of the b transitions are hidden, it is an observational implementation of $\mathcal{I}$.

Note that the set of common implementations includes all of the implementations that have a finite sequence of b's followed by a c (and possibly more b's after that). Since the length of the b sequence is unbounded (though finite), the number of such implementations is *infinite*. We show that a finite-state DMTS cannot represent such a set.

For our proof, we need the notion of a state from which a single b transition is allowed in an implementation: Let $N = (S_N, \mathcal{A}, \delta^p_N, \Delta^r_N, n_0)$ be a DMTS. A state $s \in S_N$ is a *single-b-allowed* state if there exists an LTS $I = (S_I, \mathcal{A}, \delta_I, i_0)$ strongly implementing N with an implementation relation $\mathcal{R}$, and there exists a state $i \in S_I$, s.t. (1) $(s, i) \in \mathcal{R}$; (2) there exists $i \xrightarrow{b} i' \in \delta_I$; and (3) no other transitions from i exist.

That is, for a state s in N to be a single-b-allowed state, we examine the possible implementations of N. If a legal implementation exists, with a state i corresponding to s (according to the implementation relation), from which a single b transition is departing, then we say that s is a single-b-allowed state. State 4 of model $\mathcal{K}$ in Fig. 2 is a single-b-allowed state, while state 18 of model $\mathcal{O}$ is not, since in every implementation the corresponding state must have a transition on c departing from it.

We now return to prove that no DMTS merge exists for $\mathcal{I}$ and $\mathcal{J}$. Assume by way of contradiction that there exists a DMTS M such that its implementations are exactly all of the implementations common to $\mathcal{I}$ and $\mathcal{J}$. Consider the initial state m_0 of M. The transitions from m_0 must allow an implementation with a single b transition leaving m_0 (such as $\mathcal{L}_1$). This can be achieved, e.g., by a DT as in model $\mathcal{K}$, also allowing an implementation with a single c transition leaving m_0 (such as $\mathcal{I}$ itself, which is a common implementation of $\mathcal{I}$ and $\mathcal{J}$). Note that m_0 *is* a single-b-allowed state, as defined above.

Let us now examine paths in M that contain possible b transitions (we ignore must b transitions, if exist). Let π be the longest path in M, starting from m_0, such that (1) π contains only possible transitions on label b; (2) π does not visit

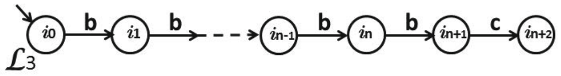

Fig. 3. $\mathcal{L}_3$ is a legal implementation of $\mathcal{I}$ and $\mathcal{J}$ of Fig. 2, showing that a DMTS merge does not exist

a state more than once; and (3) all the states on π are single-b-allowed states. Note that there must be at least one state on π (even if no transition), since the initial state m_0 is a single-b-allowed state. Since M is finite-state, π must be finite. Let n, $n \geq 1$, be the number of states on π and let m_{n-1} denote its final state. Thus, m_{n-1} is a single-b-allowed state, but all the states reachable from m_{n-1} via a transition on b are either not single-b-allowed, or already appear on π. We consider both cases below.

(1) If a b transition from m_{n-1} leads to a state that is already on π, then $[[M]]$ includes an implementation L with a loop on b transitions. But L is not an implementation of $\mathcal{I}$ (since c never appears in L)! Thus, M cannot be the merge of $\mathcal{I}$ and $\mathcal{J}$.

(2) If a b transition from m_{n-1} leads to a state m_n that is not a single-b-allowed state, then either (a) no b transitions can be taken from m_n, or (b) a b transition can be taken from m_n, but only together with another transition (on c or on b or on both). In either case, the implementation $\mathcal{L}_3$ in Fig. 3 does not exist in $[[M]]$, since it includes a path with $n+1$ different single-b-allowed states, while we assumed that the longest such path in M has only n single-b-allowed states. Note though that $\mathcal{L}_3$ is a legal implementation of $\mathcal{I}$ and $\mathcal{J}$. Thus, M cannot be the merge of $\mathcal{I}$ and $\mathcal{J}$.

Based on the above discussion, we conclude the following:

Theorem 1. *DMTSs are not closed under alphabet merge.*

4 Restricted Disjunctive MTS and Embedding

In Sec. 3 we have given a simple example for which an alphabet merge does not exist in the form of a DMTS. This can be fixed by making a small extension to DMTSs.

Consider again models $\mathcal{I}$ and $\mathcal{J}$ of Fig. 2. Model $\mathcal{Q}$ of Fig. 2 is *almost* their merge: it defines all of the legal common implementations of $\mathcal{I}$ and $\mathcal{J}$, but allows one additional implementation, shown as model $\mathcal{L}_2$, with a self-loop on b in the initial state. $\mathcal{L}_2$ is an implementation of $\mathcal{J}$ but not of $\mathcal{I}$ (since no c is ever reached), and thus it is not a common implementation. What if we restrict b in $\mathcal{Q}$ such that the implementation $\mathcal{L}_2$ is ruled out?

In this section, we introduce a new formalism, called *restricted disjunctive MTS (rDMTS)*, that does exactly this. It allows self-looped transition in a DT to be marked as 'restricted'. Model $\mathcal{B}$ in Fig. 2 marks the self-loop on b as restricted. We define a strong implementation relation for rDMTSs that rules out the unwanted implementations, by restricting marked transitions to appear in an implementation only a finite number of times.

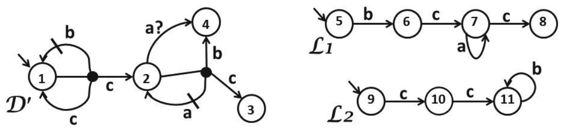

Fig. 4. Model $\mathcal{D}'$ is an rDMTS, $\mathcal{L}_1$ is a strong implementation of it while $\mathcal{L}_2$ is not

We then define the notion of *embedding* that is a key element in our algorithm. Using rDMTS, a model M with an alphabet αM can be embedded (or re-defined) into a larger alphabet $\mathcal{A}$, in a way that preserves alphabet implementations. The embedding procedure is simple: for every DT in M, we add self-looped legs on every letter from $\mathcal{A} \setminus \alpha M$, and mark them as 'restricted'. Model $\mathcal{B}$ in Fig. 2 is the embedding of model $\mathcal{I}$ in the alphabet $\{b, c\}$. We prove that this process preserves implementations, that is, if we let $M^{\mathcal{A}}$ be the re-defined rDMTS, we have that $[[M^{\mathcal{A}}]] = [[M]]_{\mathcal{A}}$. In our example, all of the alphabet implementations of $\mathcal{I}$ over $\{b, c\}$ are also strong implementations of $\mathcal{B}$ and vice versa. Thus, $[[\mathcal{B}]] = [[\mathcal{I}]]_{\{b,c\}}$.

Using rDMTS and the notion of the embedding, we can present our alphabet merge algorithm. Let M and N be models defined over the alphabets αM and αN, respectively, and let $\mathcal{A}$ be the union of the alphabets: $\mathcal{A} = \alpha M \cup \alpha N$. Embedding each model into $\mathcal{A}$ results in rDMTSs $M^{\mathcal{A}}$ and $N^{\mathcal{A}}$ over the *same* alphabet. Two same-alphabet DMTSs can be merged using the algorithm in [1]; we extend it to apply to rDMTSs and prove that if models satisfy a simple *compatibility* condition, our algorithm produces the exact merge.

Our method consists of three main components described below. In Sec. 4.1, we formally define the new formalism, rDMTS, together with a strong implementation relation for it. In Sec. 4.2, we give the *embedding* procedure that preserves alphabet implementations. Finally in Sec. 5, we present an adaptation of the existing strong merge procedure of [1] to work for rDMTSs.

4.1 Restricted Disjunctive MTS

An rDMTS differs from a DMTS in two ways: syntactically – some of the legs of every DT in an rDMTS can be marked as "restricted", and semantically – implementations of a given rDMTS must fulfill additional requirements.

Definition 7 (Restricted DMTS (rDMTS)). *$M = (S, L, \delta^p, \Delta^r, m_0, T)$ is a* restricted DMTS *if $(S, L, \delta^p, \Delta^r, m_0)$ is a DMTS and $T : \Delta^r \longrightarrow 2^{L \times S}$ is a* restricting marking function*, such that for $\langle m, V\rangle \in \Delta^r$, $T(\langle m, V\rangle) \subsetneq V$, and if $(\ell, m') \in T(\langle m, V\rangle)$ then $m' = m$. That is, every restricted leg is a self-loop.*

For an rDMTS M, we denote by $M_{\downarrow}$ its DMTS part (without the restriction marking). For a DT $\langle m, V\rangle$, we use V_T to denote the set $T(\langle m, V\rangle)$, and call the legs in V_T the *restricted* legs. The non-restricted legs, those in $V \setminus V_T$, are called the *eventual* legs and are denoted by V_E. Note that since $T(\langle m, V\rangle) \subsetneq V$, V_E cannot be empty.

Example 3. Model $\mathcal{D}'$ in Fig. 4 is an example of an rDMTS. Restricted legs are marked with a small line. Note that (a) not all self-loops are necessarily restricted, (b) there can be eventual legs labelled the same way as restricted ones, and (c) different DTs may have differently labelled restricted legs.

We define implementations of rDMTSs by preventing restricted legs from always being picked in a disjunctive transition. That is, we want to ensure that *eventual* legs, that belong to V_E, are eventually picked in every implementation.

Definition 8 (Strong Eventual Implementation). *Let* $M = (S_M, L, \delta^p_M, \delta^r_M, m_0, T)$ *be an rDMTS and* $I = (S_I, L, \delta_I, i_0)$ *be an LTS.* I *is a* strong eventual implementation *of* M *if* (m_0, i_0) *is contained in some* strong eventual implementation relation $\mathcal{R} \subseteq S_M \times I_M$*, where* $\mathcal{R}$ *is a strong implementation relation on* $M_\downarrow$ *and* I *(Def. 3), and for all* $(m, i) \in \mathcal{R}$ *and* $\langle m, V\rangle \in \Delta^r_m$*, there exists a sequence of transitions (called an* eventuality path*)* $i \xrightarrow{\ell_1} i_1 \xrightarrow{\ell_2} ... \xrightarrow{\ell_j} i_j \xrightarrow{\ell} i'$ *in* I*, s.t. (1)* $\forall 1 \le k \le j$, $(\ell_k, m) \in V_T$ *and* $(m, i_k) \in \mathcal{R}$*; (2) there exists* m' *with* $(\ell, m') \in V_E$*; and (3)* $(m', i') \in \mathcal{R}$*.*

Example 4. LTS $\mathcal{L}_1$ in Fig. 4 is an implementation of rDMTS $\mathcal{D}'$ via the relation $\mathcal{R}_1 = \{(1, 5), (1, 6), (2, 7), (3, 8)\}$. In $\mathcal{L}_1$, a self-loop on state 7 on the restricted a-transition is allowed since an eventuality path from state 7 exists. $\mathcal{L}_2$ is an implementation of $(\mathcal{D}')_\downarrow$ via the relation $\mathcal{R}_2 = \{(1, 9), (1, 10), (1, 11)\}$, but it is not a strong eventual implementation of $\mathcal{D}'$ since there is no eventuality path from state 11.

4.2 rDMTS Embedding

In order to embed a model M into a larger alphabet $\mathcal{A}$, we add self-loop *maybe transitions* on every label from $\mathcal{A} \setminus \alpha M$ to every state of M. In addition, we add self-loop *legs* on every label from $\mathcal{A} \setminus \alpha M$ to every DT in M. We use the restriction function to mark all new legs as restricted. An input to the embedding procedure, formalized in Def. 9, is an rDMTS rather than a DMTS, indicating that an rDMTS can also be embedded into a larger alphabet.

Note that a DMTS can be easily converted into an rDMTS by defining the restriction function T as $T(\langle m, V\rangle) = \emptyset$ for every $\langle m, V\rangle \in \Delta^r$.

Definition 9 (Embedding in a Larger Alphabet). *Let* $M = (S, \alpha M, \delta^p, \Delta^r, m_0, T)$ *be an rDMTS, and* $\mathcal{A}$ *be an alphabet s.t.* $\alpha M \subseteq \mathcal{A}$*. For each state* $m \in S$*, we define a set of "legs" to be added:* $R(m) = \{(\ell, m) \mid \ell \in \mathcal{A} \setminus \alpha M\}$*. An* embedding of M into $\mathcal{A}$ *is an rDMTS* $M^{\mathcal{A}} = (S, \mathcal{A}, \delta^{p'}, \Delta^{r'}, m_0, T')$ *s.t. (1)* $\delta^{p'} = \delta^p \cup \{(m, \ell, m) \mid \ell \in \mathcal{A} \setminus \alpha M\}$*; (2)* $\Delta^{r'} = \{\langle m, V \cup R(m)\rangle | \ \langle m, V\rangle \in \Delta^r\}$*; and (3)* $T'(\langle m, V \cup R(m)\rangle) = T(\langle m, V\rangle) \cup R(m)$*.*

Note that (a) the embedding operation *adds* restricted legs but does not touch existing legs, whether restricted or not, and (b) $(M^{\mathcal{A}})_\downarrow \ne M$ since the embedding procedure adds disjunctive legs as well as maybe transitions, but those are not removed when looking at the DMTS part $(M^{\mathcal{A}})_\downarrow$. The $_\downarrow$ operator removes only the restricting markings, leaving the transitions themselves unchanged.

Example 5. Model $\mathcal{I}$ in Fig. 2 is embedded in the alphabet $\{b, c\}$ to get model $\mathcal{B}$ of the same figure. In Fig. 6, $\mathcal{B}$ is further embedded in the alphabet $\{a, b, c\}$ to get model $\mathcal{B}'$.

The above definitions establish that the rDMTS embedding is compositional:

Proposition 1. *Let M be a model and $\mathcal{A}_1, \mathcal{A}_2$ be alphabets s.t. $\alpha M \subseteq \mathcal{A}_1 \subseteq \mathcal{A}_2$. Then, $(M^{\mathcal{A}_1})^{\mathcal{A}_2} = M^{\mathcal{A}_2}$.*

The proof follows directly from Def. 9.

The following theorem guarantees that the embedding procedure constructs an rDMTS such that all alphabet implementations of the original model are strong eventual implementations of the rDMTS. Thus, alphabet implementations of the original DMTS are preserved.

Theorem 2. *Let M be a DMTS and I be an LTS s.t. $\alpha M \subseteq \alpha I$. I is an alphabet implementation of M iff I is a strong eventual implementation of $M^{\alpha I}$.*

Having defined an embedding operation for rDMTSs, models with different vocabularies can be lifted to models with the same vocabularies and merged using a strong merge operator.

5 Merge Using rDMTSs

The merge algorithm for rDMTSs is based on the algorithm of Benes et al. [1] for merging DMTSs defined on the same alphabet. We first review the algorithm given in [1] and then discuss the modifications we need to make so that it applies to rDMTSs.

5.1 Strong Merge of DMTSs

In order for two DMTSs to be merged, the models must be *consistent*, that is, they must have at least one common implementation. The algorithm of [1] is based on a *consistency relation* between the states of the DMTSs to be merged. States m and n are in a consistency relation if for each DT $\langle m, V\rangle$, at least one leg in V has a corresponding possible transition from n and vice versa:

Definition 10 (DMTSs Consistency Relation [1]). *A* strong consistency relation *between DMTSs $M = (S_M, L, \delta^p_M, \Delta^r_M, m_0)$ and $N = (S_N, L, \delta^p_N, \Delta^r_N, n_0)$ is a relation $\mathcal{C} \subseteq S_M \times S_N$ s.t. $(m_0, n_0) \in \mathcal{C}$ and $\forall (m, n) \in \mathcal{C}$, the following holds:*

1. *$\forall \langle m, V\rangle \in \Delta^r_M$, $\exists (l, m') \in V$ and $n \xrightarrow{\ell}_p n'$ in N s.t. $(m', n') \in \mathcal{C}$.*
2. *$\forall \langle n, U\rangle \in \Delta^r_N$, $\exists (q, n') \in U$ and $m \xrightarrow{q}_p m'$ in M s.t. $(m', n') \in \mathcal{C}$.*

Based on a consistency relation $\mathcal{C}$ between M and N, we can now compose them into a single DMTS. The composition is done by constructing, for each DT $\langle m, V\rangle$ in M (or N), a corresponding DT $\langle p, W\rangle$ in the composed model P, where a leg (ℓ, p') exists in W whenever (ℓ, m') exists in V, a leg $n \xrightarrow{\ell}_p n'$ is possible in N, and $(m', n') \in \mathcal{C}$.

Definition 11 (Compose [1]). *Let M and N be DMTSs with the same vocabulary L, and let $\mathcal{C}$ be a consistency relation between them. The $+$ operator between M and N is defined as $[M + N]_{\mathcal{C}} = (\mathcal{C}, L, \delta^p_{M+N}, \Delta^r_{M+N}, (m_0, n_0))$, where δ^p_{M+N} and Δ^r_{M+N} are defined to be the smallest relations that satisfy the following rules:*

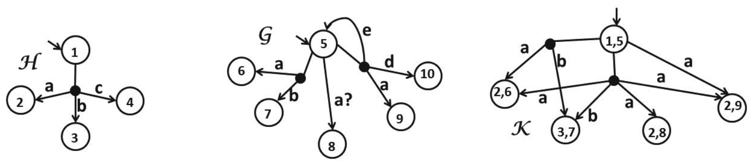

Fig. 5. DMTS $\mathcal{H}$ and $\mathcal{G}$, and their strong merge $\mathcal{K}$

(RM) $\frac{\langle m,V\rangle}{\langle (m,n),W\rangle}$, *where* $W = \{(l,(m',n')) \mid (l,m') \in V \wedge n \xrightarrow{\ell}_p n' \wedge (m',n') \in \mathcal{C}\}$

(MR) $\frac{\langle n,U\rangle}{\langle (m,n),W\rangle}$, *where* $W = \{(l,(m',n')) \mid (l,n') \in U \wedge m \xrightarrow{\ell}_p m' \wedge (m',n') \in \mathcal{C}\}$

(MM) $\frac{m \xrightarrow{\ell}_p m',\ n \xrightarrow{\ell}_p n'}{(m,n) \xrightarrow{\ell}_p (m',n')}$

When $\mathcal{C}$ is the largest consistency relation between M and N, the composition w.r.t. $\mathcal{C}$ becomes the *merge* of M and N.

Theorem 3 (Correctness of Strong DMTS Merge [1]). *Let M and N be DMTSs with the same vocabulary. If $\mathcal{C}$ is the largest consistency relation between the states of M and N then $[M + N]_{\mathcal{C}}$ is the merge of M and N.*

Example 6. Consider the DMTSs $\mathcal{H}$ and $\mathcal{G}$ in Fig. 5, defined over the alphabet $\{a,b,c,d,e\}$. Model $\mathcal{H}$ has one DT with three legs, and $\mathcal{G}$ has two DTs. Rules *MR* and *RM* produce one DT in the merged model $\mathcal{K}$ for each DT in the original models, resulting in three altogether. $\mathcal{H}$'s DT contributes a four-legged DT in $\mathcal{K}$; three of the legs are labelled by a and the fourth – by b. The merged DT is constructed by taking all a-labelled maybe transitions in $\mathcal{G}$ that reach a state consistent with the state 2 of $\mathcal{H}$. In $\mathcal{G}$, there are three such transitions, leading to states 6, 8, and 9 (recall that there is a maybe transition for every leg of a DT), all of which are consistent with state 2 of $\mathcal{H}$. The three-legged DT in $\mathcal{G}$ (on labels a, d and e) results in a DT with a single transition labelled a in $\mathcal{K}$ that reaches the state (2,9) since model $\mathcal{H}$ has no maybe transitions on d or e.

Note that every DT $\langle p, W\rangle$ in the composition P of DMTSs M and N has a *source* DT $\langle m, V\rangle$ in either N or M, and every leg in W has a source leg in V. As discussed above, a DT in P is introduced either by a rule *RM* or *MR*, based on a DT that exists in either M or N. The notions of a source DT and source leg are needed in the sequel, and we formalize them below.

Definition 12 (Source DT, Source Leg). *Let M and N be consistent DMTSs, $\mathcal{C}$ be a consistency relation on their states and P be their composition with relation to $\mathcal{C}$. Let $\langle p, W\rangle \in \Delta^r_P$ be a DT in P. We say that $\langle m, V\rangle \in \Delta^r_M$ is the* source DT *of $\langle p, W\rangle$ if (1) $p = (m,n)$ and (2) $(\ell, p') \in W$ iff there exist $(\ell, m') \in V$ and $n \xrightarrow{\ell}_p n'$ in N s.t. $(m',n') \in \mathcal{C}$. We say that a leg $(\ell, m') \in V$ is the* source leg *of $(\ell, p') \in W$ if $p' = (m', n')$.*

5.2 Composition of rDMTSs

We now define the composition of two rDMTSs, M and N. This composition is not necessarily their *merge*, i.e., the set of implementations represented by it can sometimes include implementations that are not common to M and N. In Sec. 5.3, we characterize the cases for which the composition algorithm yields exactly the merge of M and N.

The composition is obtained by modifying the algorithm given in Def. 11, making it applicable to rDMTSs. We first modify the consistency relation on which the composition is based: we define *eventual* consistency relation, to comply with the definition of strong eventual implementations of rDMTS (Def. 8). We modify the composition itself by adding restriction markings on legs of the composed model.

Definition 13 (Eventual Consistency Relation). *Let $M = (S_M, L, \delta^m_M, \Delta^r_M, m_0, T_M)$ and $N = (S_N, L, \delta^m_N, \Delta^r_N, n_0, T_N)$ be rDMTSs. $\mathcal{C}$ is an* eventual consistency relation *between M and N if it is a strong consistency relation between $M_\downarrow$ and $N_\downarrow$, and for all $(m,n) \in \mathcal{C}$, the following holds:*

1. *$\forall \langle m, V\rangle \in \Delta^r_M$, there exists a sequence of possible transitions in N (called* a possible eventuality path*): $n \xrightarrow{\ell_1}_p n_1 \xrightarrow{\ell_2}_p ... \xrightarrow{\ell_j}_p n_j \xrightarrow{\ell}_p n'$ s.t. (i) $\forall 1 \leq i \leq j$, $(\ell_i, m) \in V_T$ and $(m, n_i) \in \mathcal{C}$; (ii) there exists m' with $(\ell, m') \in V_E$, and (iii) $(m', n') \in \mathcal{C}$.*
2. *$\forall \langle n, U\rangle \in \Delta^r_N$, there exists a sequence of possible transitions in M: $m \xrightarrow{q_1}_p m_1 \xrightarrow{q_2}_p ... \xrightarrow{q_i}_p m_i \xrightarrow{q}_p m'$ with the same conditions as above.*

This relation requires the existence of a consistency relation between the DMTS parts $M_\downarrow$ and $N_\downarrow$ (Def. 10). In addition, we need to make sure that for every DT in M (or N), at least one non-restricted leg is eventually allowed on a path in N, and all the transitions in between are restricted. This guarantees the existence of a common implementation. The following theorem states that the opposite is also correct: if a common implementation exists then so does an eventual consistency relation.

Theorem 4. *Let M and N be rDMSTs. An eventual consistency relation exists between M and N if and only if there exists an LTS I that is a strong eventual implementation of both M and N.*

The composition of rDMTSs can now be defined. We base it on the largest *eventual* consistency relation $\mathcal{C}$ on the rDMTSs at hand (Def. 13), and use the *source* DT (Def. 12) to mark restricted legs: a self-looped leg in a DT of the composed model is marked as restricted if and only if its source leg is restricted.

Definition 14 (Composition of rDMTSs). *Let $M = (S_M, L, \delta^m_M, \Delta^r_M, m_0, T_M)$ and $N = (S_N, L, \delta^m_N, \Delta^r_N, n_0, T_N)$ be rDMTSs and let $\mathcal{C}$ be the largest eventual consistency relation between them. We define $P = (S_P, L, \delta^m_P, \Delta^r_P, p_0, T_P)$, to be the composition of M and N, where $P_\downarrow = [M_\downarrow + N_\downarrow]_\mathcal{C}$ (Def. 11). For each $\langle p, W\rangle \in \Delta^r_P$, let $\langle m, V\rangle \in \Delta^r_M$ be its* source *DT. We define $T_P(\langle p, W\rangle) = \{(l, p) \in W \mid \exists (l, m) \in T_M(\langle m, V\rangle)\}$.*

The rDMTS composition inherits the restriction markings of each DT from its source DT. Note that the rDMTS resulting from a compose operation can be composed again, if desired. We demonstrate this in the example below.

Example 7. Model $\mathcal{B}$ of Fig. 6, defined over the alphabet $\{b, c\}$, is the merge of models $\mathcal{I}$ and $\mathcal{J}$ of Fig 2. We want to compose it further with model $\mathcal{M}$ of Fig. 6, defined

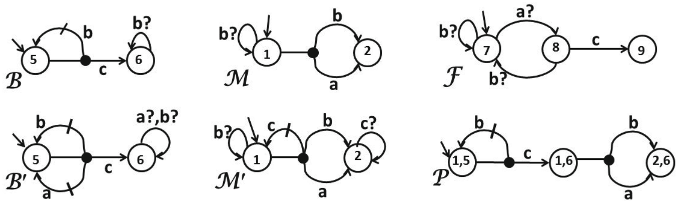

Fig. 6. rDMTSs $\mathcal{M}'$ and $\mathcal{F}$ are consistent with $\mathcal{B}'$ but $\mathcal{F}$ is not compatible with $\mathcal{B}'$. Model $\mathcal{P}$ is the merge of $\mathcal{B}'$ and $\mathcal{M}'$.

over $\{a, b\}$. We thus embed each of the models in the alphabet {a,b,c}: we add a self-loop leg labelled a to the single DT of $\mathcal{B}$, as well as a self-loop possible transition labelled a to state 6. The result is shown as model $\mathcal{B}'$ of Fig. 6. In the same way, we add self-loop transitions labelled c to $\mathcal{M}$, to get model $\mathcal{M}'$. In order to compose them, we find their largest consistency relation $\mathcal{C} = \{(1, 5), (1, 6), (2, 6)\}$. The composition according to Def. 14 is shown in model $\mathcal{P}$.

5.3 Characterizing rDMTS Merge

The composition algorithm given in Def. 14 does not always construct the merge of the input rDMTS. In this section, we define the notion of *compatibility* of two models, and prove that the composition of two compatible rDMTSs is guaranteed to be their merge.

In terms of the original models (before embedding), compatibility means that all loops in one model share some vocabulary with the other model. In the rDMTS terms, it means that there are no loops (of size larger than one) in one model, on labels that are restricted in the other model.

More specifically, let M and N be the rDMTSs to be merged, and let $\mathcal{C}$ be their largest consistency relation. For $(m, n) \in \mathcal{C}$, we require that m does not participate in a loop consisting of labels that are all restricted in some DT from n, and vice versa. Presence of the consistency relation $\mathcal{C}$ allows us to limit this requirement only to pairs of states in $\mathcal{C}$, and thus it is easier to check on rDMTSs rather than on the original models.

We begin by defining a loop on a set of labels.

Definition 15 (A-loops). *Let $M = (S_M, L, \delta^m_M, \Delta^r_M, m_0, T_M)$ be an rDMTS, $m \in S_M$ be a state, and $\mathcal{A}$ be a set of labels. An $\mathcal{A}$-*loop *from m is a sequence of maybe transitions, $m \xrightarrow{\ell_1}_p m_1 \xrightarrow{\ell_2}_p ... \xrightarrow{\ell_j}_p m$, s.t. $\ell_1, ..., \ell_j \in \mathcal{A}$ and $m_1 \neq m$.*

Example 8. Model $\mathcal{F}$ in Fig. 6 has an $\{a, b\}$-loop from state 7.

Using the concept of an $\mathcal{A}$-loop, we now define compatibility between states.

Definition 16 (State Compatibility). *Let $M = (S_M, L, \delta^m_M, \Delta^r_M, m_0, T_M)$ and $N = (S_N, L, \delta^m_N, \Delta^r_N, n_0, T_N)$ be rDMTSs, and $m \in S_M$, $n \in S_N$ be states. Let $\langle m, V\rangle \in \Delta^r_M$, and $\mathcal{A}_T$ be the set of restricted labels in V. If there are no $\mathcal{A}_T$-loops from n then n* is $\langle m, V\rangle$-compatible*. If for all V s.t. $\langle m, V\rangle \in \Delta^r_M$, n is $\langle m, V\rangle$-compatible, then n* is compatible with *m.*

Example 9. Consider models $\mathcal{M}'$ and $\mathcal{B}'$ in Fig. 6. From state 5 of $\mathcal{B}'$, there is only one DT: $(5, \{(a, 5), (b, 5), (c, 6)\})$, with $\mathcal{A}_T = \{a, b\}$. State 1 of $\mathcal{M}'$ has no $\{a, b\}$-loops. Therefore, state 1 of $\mathcal{M}'$ and state 5 of $\mathcal{B}'$ are compatible.

Definition 17 (Model Compatibility). *Let M and N be consistent rDMTSs, with a consistency relation $\mathcal{C}$. M and N are* compatible models with respect to $\mathcal{C}$ *if for all $(m, n) \in \mathcal{C}$, m is compatible with n, and n is compatible with m.*

Example 10. Models $\mathcal{M}'$ and $\mathcal{F}$ in Fig. 6 are consistent with model $\mathcal{B}'$. Yet, $\mathcal{F}$ is not compatible with $\mathcal{B}'$. To see this, note that $(5, 7)$, representing the initial states, must exist in every consistency relation between $\mathcal{F}$ and $\mathcal{B}'$. Now consider the loop on labels a and b from state 7 of $\mathcal{F}$. This is an $\{a, b\}$-loop (see Example 8). But $\{a, b\}$ is exactly the set $\mathcal{A}_T$ of the single DT of $\mathcal{B}'$ (see Example 9). Thus, by Def. 17, $\mathcal{F}$ and $\mathcal{B}'$ are not compatible. $\mathcal{M}'$ has no $\{a, b\}$-loops at all and thus is compatible with $\mathcal{B}'$.

The theorem below is one of the main results of our paper, stating correctness of our rDMTS composition operation given in Def. 14.

Theorem 5. *Let M and N be rDMTSs over the same vocabulary and $\mathcal{C}$ be the largest eventual consistency relation between them. Assume that M and N are compatible w.r.t. $\mathcal{C}$ and let P be their composition, as defined by Def. 14. Then P is the strong merge of M and N.*

5.4 Alphabet Merge of Partial Behavioral Models

We now combine the results of Sec. 4 and 5, to form an algorithm for the merge of two models, whether they are LTSs, MTSs, DMTSs or rDMTSs, and whether they are defined on the same alphabet or not.

Algorithm 1 (Alphabet Merge) *Let M and N be models with alphabets αM and αN, respectively, and let $\mathcal{A} = \alpha M \cup \alpha N$. The* alphabet merge *of M and N, denoted by $M +_\alpha N$, is an rDMTS constructed by the following algorithm:*

1. *Construct the embedded models $M^{\mathcal{A}}$ and $N^{\mathcal{A}}$ (Def. 9).*
2. *Compute the largest consistency relation $\mathcal{C}$ on $M^{\mathcal{A}}$ and $N^{\mathcal{A}}$ (Def. 13).*
3. *If $\mathcal{C} = \emptyset$, or if $M^{\mathcal{A}}$ and $N^{\mathcal{A}}$ are not compatible w.r.t. $\mathcal{C}$, return NULL.*
4. *Return the composition of $M^{\mathcal{A}}$ and $N^{\mathcal{A}}$ as defined in Def. 14.*

Theorem 6. *Let P be the result of Algorithm 1 when called on M and N. If P is not NULL, then the set of strong eventual implementations of P is exactly the set of alphabet implementations common to M and N.*

The proof follows immediately from Theorems 2 and 5.

6 Discussion and Future Work

The difficulty in merging models defined over different vocabularies stems from the fact that a common implementation might have to be considered a *strong* implementation of one model and at the same time an *observational* implementation of the other. Restricted DMTSs, introduced in this paper, bridge the gap between the immediate nature of a strong implementation and the eventual nature of an observational implementation.

A key result of this paper is that rDMTSs are closed under merge for compatible models, which is an important step forward in providing a framework for merging operational yet partial models of system behaviour. We believe the compatibility requirement is sensible from an engineering point of view (models are required to represent viewpoints in which there is a certain degree of overlap). However, experimentation on whether this limitation is inconvenient in practice is necessary. Furthermore, we believe that the compatibility requirement can be relaxed at the cost of making the "restriction marking" function more complex; investigating this further is left for future work.

Acknowledgements. Shoham Ben-David is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship. This work was partially supported by ERC StG PBM-FIMBSE and by the Ontario Ministry of Research and Innovation.

References

1. Beneš, N., Černá, I., Křetínský, J.: Modal Transition Systems: Composition and LTL Model Checking. In: Bultan, T., Hsiung, P.-A. (eds.) ATVA 2011. LNCS, vol. 6996, pp. 228–242. Springer, Heidelberg (2011)
2. Chechik, M., Brunet, G., Fischbein, D., Uchitel, S.: Partial behavioural models for requirements and early design. In: Proc. of MMOSS 2006 (2006)
3. Dams, D.: Abstract Interpretation and Partition Refinement for Model Checking. PhD thesis, Eindhoven University of Technology, The Netherlands (July 1996)
4. D'Ippolito, N., Fischbein, D., Chechik, M., Uchitel, S.: MTSA: The Modal Transition System Analyser. In: Proc. of ASE 2008, pp. 475–476 (2008)
5. Fischbein, D., Braberman, V.A., Uchitel, S.: A Sound Observational Semantics for Modal Transition Systems. In: Leucker, M., Morgan, C. (eds.) ICTAC 2009. LNCS, vol. 5684, pp. 215–230. Springer, Heidelberg (2009)
6. Fischbein, D., D'Ippolito, N., Brunet, G., Chechik, M., Uchitel, S.: Weak Alphabet Merging of Partial Behavior Models. ACM TOSEM 21(2), 9 (2012)
7. Fischbein, D., Uchitel, S.: On Correct and Complete Strong Merging of Partial Behaviour Models. In: Proc. of SIGSOFT FSE 2008, pp. 297–307 (2008)
8. Harel, D.: StateCharts: A Visual Formalism for Complex Systems. Sc. of Comp. Prog. 8, 231–274 (1987)
9. Hüttel, H., Larsen, K.G.: The Use of Static Constructs in A Modal Process Logic. In: Meyer, A.R., Taitslin, M.A. (eds.) Logic at Botik 1989. LNCS, vol. 363, pp. 163–180. Springer, Heidelberg (1989)
10. Keller, R.M.: Formal Verification of Parallel Programs. CACM 19(7) (1976)
11. Larsen, K.G., Thomsen, B.: A Modal Process Logic. In: Proc. of LICS (1988)
12. Larsen, K.G., Xinxin, L.: Equation Solving Using Modal Transition Systems. In: Proc. of LICS 1990, pp. 108–117 (1990)
13. Sibay, G., Uchitel, S., Braberman, V.A.: Existential Live Sequence Charts Revisited. In: Proc. of ICSE 2008, pp. 41–50 (2008)
14. Uchitel, S., Brunet, G., Chechik, M.: Synthesis of Partial Behavior Models from Properties and Scenarios. IEEE TSE 35(3), 384–406 (2009)
15. Uchitel, S., Chechik, M.: Merging Partial Behavioural Models. In: Proc. of SIGSOFT FSE 2004, pp. 43–52 (2004)
16. Zave, P., Jackson, M.: Conjunction as composition. ACM TOSEM 2, 379–411 (1993)