Merging Partial Behavioural Models

Sebastian Uchitel
Department of Computing
Imperial College
180 Queen's Gate
London, SW7 2RH, UK
s.uchitel@doc.ic.ac.uk

Marsha Chechik
Department of Computer Science
University of Toronto
40 St. George Street
Toronto, Ontario, Canada M5S 2E4
chechik@cs.toronto.edu

ABSTRACT

Constructing comprehensive operational models of intended system behaviour is a complex and costly task. Consequently, practitioners have adopted techniques that support incremental elaboration of partial behaviour descriptions. A noteworthy example is the wide adoption of scenario-based notations such as message sequence charts. Scenario-based specifications are partial descriptions that can be incrementally elaborated to cover the system behaviour that is of interest. However, how should partial behavioural models described by different stakeholders with different viewpoints covering different aspects of behaviour be composed? How should partial models of component instances of the same type be put together?

In this paper, we propose *model merging* as a general solution to these questions. We formally define model merging based on observational refinement and show that merging consistent models is a process that should result in a minimal common refinement. Because minimal common refinements are not guaranteed to be unique, we argue that the modeller should participate in the process of elaborating such a model. We also discuss the role of the least common refinement and the greatest lower bound of all minimal common refinements in this elaboration process. In addition, we provide algorithms for i) checking consistency between two models; ii) constructing their least common refinement if one exists; iii) supporting the construction of a minimal common refinement if there is no least common refinement.

1. INTRODUCTION

State-based behaviour modeling and analysis has been shown to be successful in uncovering subtle design errors [3]. However, the adoption of such technologies by practitioners has been slow. Partly, this is due to the difficulty of constructing behavioural models – this task requires considerable expertise in modeling notations that developers often lack. In addition, and perhaps more importantly, the benefits of the analysis appear *after* comprehensive behavioural models have been built: classical state-based modeling approaches are generally not suited for providing early feedback, when system descriptions are still partial.

In contrast, interaction-based specifications, such as message sequence charts [11], are becoming increasingly popular. These scenario-based notations are partial behavioural descriptions that promote *incremental* elaboration of system behaviour.

Lately, there has been interest in developing an understanding and exploiting the relation between interaction-based and state-based modeling techniques [20]. In particular, several approaches to the synthesis of state-based models from scenarios-based specifications (e.g. [23, 14]) have been developed. These approaches aim to combine the benefits of the incremental elaboration in interaction-based specifications with the behavioural analysis in state-based models.

A current limitation of synthesis approaches is that the models being synthesized, e.g., labeled transition systems (LTSs) [13], are assumed to be complete descriptions of the system behaviour up to some level of abstraction, i.e., the state machine is assumed to completely describe the system behaviour with respect to a fixed alphabet of actions. This completeness assumption is limiting, considering that interaction-based specifications are inherently partial.

A more appropriate type of state-based model to synthesize is the one in which currently unknown aspects of behaviour can be explicitly modelled. Hence, these models can distinguish between positive, negative and unknown behaviours. Positive behaviour refers to the behaviour the system is expected to exhibit; negative behaviour refers to the behaviour the system is expected to never exhibit; unknown behaviour could become positive or negative, but the choice has not yet been made. State-based models that distinguish between these kinds of behaviour are referred to as *partial behavioural models*, e.g., Partial Labelled Transition Systems (PLTSs) [22], multi-valued state machines [6], Modal Transition Systems (MTSs) [15], Mixed Transition Systems [5] and multi-valued Kripke structures [2].

Although synthesis of partial behavioural models can provide substantial benefits [22], we have found that such models lack a specific concept that is particularly helpful in the context of behavioural model elaboration, namely, *model merging*. Scenarios are typically provided by different stakeholders with different viewpoints [9], describing different, yet overlapping aspects [4] of the same system. How should these partial models be put together? Alternatively, consider combining behavioural models of component instances of the same type. Typically, several instances of the same component may appear in a given scenario, e.g., several instances of a client component that concurrently access a server. Standard approaches to synthesis produce a separate behavioural model for each client instance (e.g. [23, 14]). However, it is reasonable to integrate all models of all client instances into a model for the client component type because all clients should share the same characteristics. How can these partial models be composed?

Composition of behavioural models is an old idea [18, 8]; however, its main focus has been on *parallel* composition which describes how two *different* components work together. In the context of model elaboration, what we are interested in is composing two partial descriptions of the *same* component to obtain a more elaborate version of both original partial descriptions. We call this operation a *merge*.

In this paper, we introduce the notion of merging in the context of an adaptation of MTSs [15]. We argue that the core concept underlying model merging is that of common observational refinement and define consistency as the existence of a common observational refinement. We show that merging consistent models is a process that should result in a minimal common observational refinement and discuss the role of the least common observational refinement and the greatest lower bound of all minimal common observational refinements in this process. We also provide algorithms that under a determinacy condition automatically check consistency, construct the least common observational refinement if there is one, and support the construction of a minimal common observational refinement otherwise.

The rest of this paper is organized as follows. In Section 2, we define MTSs and observational refinement. Section 3 describes merging MTSs. Section 4 presents the algorithms associated with the merging. We review related work in Section 5 and conclude the paper with a discussion, summary and directions for future research. Due to space restrictions, this paper does not include a more complex example to illustrate our approach; however, one can be found online [21].

2. BACKGROUND

In this section, we define, exemplify and discuss labelled transition systems, modal transition systems and refinement.

We start with the familiar concept of labeled transition systems (LTSs) [13] which are widely used for modelling and analyzing the behaviour of concurrent and distributed systems [3]. An LTS is a state transition system where transitions are labelled with actions. The set of actions of an LTS is called its *communicating alphabet* and constitutes the interactions that the modelled system can have with its environment. In addition, LTSs can have transitions labelled with τ, representing actions that are not observable by the environment. Examples of a graphical representation of LTSs are models $\mathcal{A}$ and $\mathcal{B}$, given in Figure 1. In this paper, the state labelled 0 is assumed to be the initial state of the transition system, unless stated otherwise.

Definition 1. (Labeled Transition Systems) *Let* States *be a universal set of states, Act be a universal set of observable action labels, and let $Act_\tau = Act \cup \{\tau\}$. A* labeled transition system *(LTS) is a tuple $P = (S, L, \Delta, s_0)$, where $S \subseteq$ States is a finite set of states, $L \subseteq Act_\tau$ is a set of labels, $\Delta \subseteq (S \times L \times S)$ is a transition relation between states, and $s_0 \in S$ is the initial state. We use $\alpha P = L \setminus \{\tau\}$ to denote the communicating alphabet of P.*

Existing semantics for LTSs assume that an LTS gives a complete behavioural description with respect to its alphabet. Consider the LTS $\mathcal{A}$ which models a read lock. Starting in state 0, this model allows sequences of alternating `acquireReadLock` and `releaseReadLock` actions, and, by the completeness assumption, does not allow two `acquireReadLocks` actions without having a `releaseReadLock` action in between. LTS $\mathcal{A}$ is modelling a lock that can be held by at most one reader at any time. Model $\mathcal{B}$, on the other hand, allows two readers to hold the lock simultaneously. $\mathcal{A}$ and $\mathcal{B}$ are not considered to be equivalent under any of the standard equivalence relations such as strong bisimulation, trace, observational, or failure equivalence [8, 18].

Modal transition systems (MTSs) allow explicit modelling of what is *not* known about the behaviour of a system. MTSs extend LTSs with an additional set of transitions that model the interactions with the environment that the system cannot be guaranteed to provide, but equally cannot be guaranteed to prohibit.

Definition 2. (Modal Transition Systems) *A* modal transition system *(MTS) M is a structure $(S, L, \Delta^r, \Delta^p, s_0)$, where $\Delta^r \subseteq \Delta^p$, (S, L, Δ^r, s_0) is an LTS representing* required *transitions of the system and (S, L, Δ^p, s_0) is an LTS representing* possible *(but not necessarily required) transitions of the system. We use $\alpha M = L \setminus \{\tau\}$ to denote the communicating alphabet of M.*

Figure 1 shows a graphical representation of some MTSs. For example, the MTS $\mathcal{C}$ models a partial policy for a read lock that can be acquired by at least one reader at any time, but does not rule out concurrent readers. Transition labels that have a question mark are those in $\Delta^p - \Delta^r$. We refer to these as "maybe" transitions, to distinguish them from required transitions (those in Δ^r). Note that LTSs are a special type of MTSs that do not have maybe transitions; thus, models $\mathcal{A}$ and $\mathcal{B}$ can be considered MTSs as well.

Given an MTS $M = (S, L, \Delta^r, \Delta^p, s_0)$, we say that M transitions on ℓ through a required transition (denoted $M \xrightarrow{\ell}_r M'$) if $M' = (S, L, \Delta^r, \Delta^p, s_0')$ and $(s_0, \ell, s_0') \in \Delta^r$. Similarly, M transitions on ℓ through a maybe transition ($M \xrightarrow{\ell}_m M'$) if $(s_0, \ell, s_0') \in \Delta^p - \Delta^r$. $M \xrightarrow{\ell}_p M'$ refers to possible transitions ($(s_0, \ell, s_0') \in \Delta^p$). We write $M \xrightarrow{\ell}_p$ to mean $\exists M' \cdot M \xrightarrow{\ell}_p M'$. We say that M proscribes ℓ ($M \stackrel{\ell}{\not\longrightarrow}$) if M cannot transit on ℓ through maybe or required transitions. Finally, for an MTS $M = (S, L, \Delta^r, \Delta^p, s_0)$ and a state $n \in S$, we denote changing the initial state of M from s_0 to n as M_n. For example, some transitions of the MTS $\mathcal{C}$, shown in Figure 1, are $\mathcal{C}_0 \xrightarrow{\texttt{acquireReadLock}}_r \mathcal{C}_1$ (between states 0 and 1), and $\mathcal{C}_1 \xrightarrow{\texttt{releaseReadLock}}_m \mathcal{C}_1$ (self-loop in state 1).

In this presentation, we associate each MTS with its communicating alphabet, extending the presentation of [15]. The communicating alphabet is the set of events that are relevant to the model, i.e., the scope of a partial description. Allowing models to have different scopes is fundamental for merging descriptions of different concerns and viewpoints. In addition, our choice is in line with process algebra semantics such as FSP [17].

Refinement of MTSs captures the notion of elaboration of a partial description into a more comprehensive one, in which some knowledge over the maybe behaviour has been

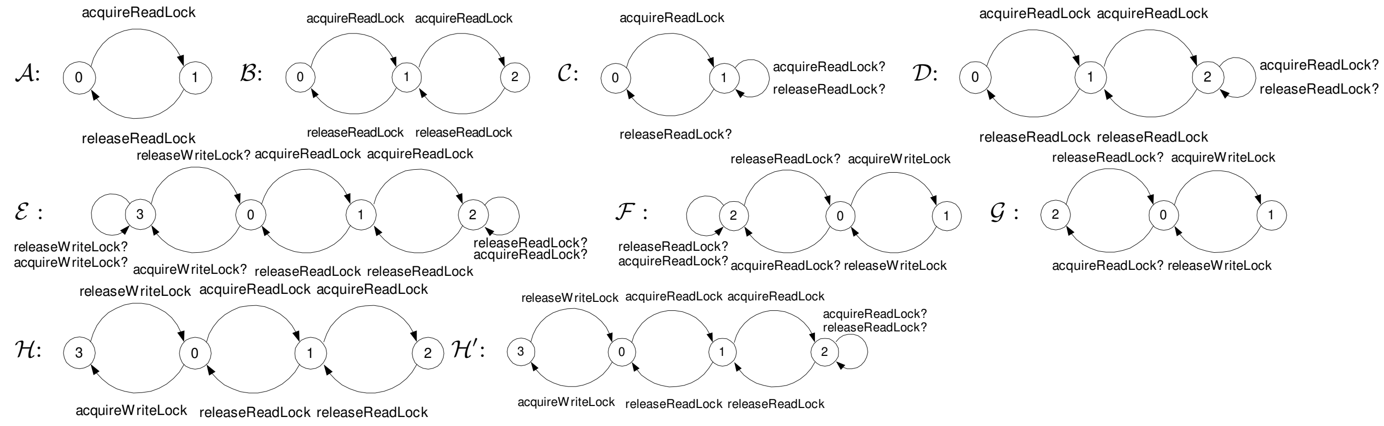

Figure 1: LTSs and MTSs: $\mathcal{A}$: at most one reader can acquire the lock; $\mathcal{B}$: at most two readers can hold the lock concurrently. $\mathcal{C}$: at least one reader can access the lock; $\mathcal{D}$: at least two readers can hold the lock concurrently; $\mathcal{E}$: readers cannot acquire the lock if it is held by writers; $\mathcal{F}$: at most one writer can access the lock but not while readers hold it; $\mathcal{G}$: at most one reader and writer and can access the lock but not concurrently; $\mathcal{H}$: maximum two concurrent readers and one writer, readers and writers exclude each other. $\mathcal{H}'$: at least two concurrent readers and maximum one writer, readers and writers exclude each other.

gained. It can be seen as being a "more defined than" relation between two partial models. Intuitively, refinement in MTSs is about converting maybe transitions into required transitions or removing them altogether: an MTS N refines M if N preserves all of the required and all of the proscribed behaviours of M. Alternatively, an MTS N refines M if N can simulate the required behaviour of M, and M can simulate the possible behaviour of N.

Definition 3. (Refinement) *Let $\wp$ be the universe of MTSs. N is a refinement of M, written $M \preceq N$, when $\alpha M = \alpha N$ and (M, N) is contained in some* refinement relation $R \subseteq \wp \times \wp$ *for which the following holds for all $\ell \in Act_\tau$:*

1. $(M \xrightarrow{\ell}_r M') \Rightarrow (\exists N' \cdot N \xrightarrow{\ell}_r N' \wedge (M', N') \in R)$
2. $(N \xrightarrow{\ell}_p N') \Rightarrow (\exists M' \cdot M \xrightarrow{\ell}_p M' \wedge (M', N') \in R)$

Note that the second condition guarantees that if N has a required transition, M has a maybe or a required transition, whereas if N has a maybe transition, then M has a maybe transition – otherwise, the first condition is violated.

Consider the MTSs shown in Figure 1. The MTS $\mathcal{C}$ is refined by the LTS $\mathcal{A}$ ($\mathcal{C} \preceq \mathcal{A}$), incorporating the new knowledge that the maybe self-loop at state $\mathcal{C}_1$ should be removed. The refinement relation between these models is $R = \{(\mathcal{C}_0, \mathcal{A}_0), (\mathcal{C}_1, \mathcal{A}_1)\}$. The LTS $\mathcal{B}$ refines $\mathcal{C}$ ($\mathcal{C} \preceq \mathcal{B}$), with the refinement relation $R = \{(\mathcal{C}_0, \mathcal{B}_0), (\mathcal{C}_1, \mathcal{B}_1), (\mathcal{C}_1, \mathcal{B}_2)\}$. Finally, the MTS $\mathcal{D}$ refines $\mathcal{C}$ via the relation $R = \{(\mathcal{C}_0, \mathcal{D}_0), (\mathcal{C}_1, \mathcal{D}_1), (\mathcal{C}_1, \mathcal{D}_2)\}$.

Although refinement captures the notion of model elaboration, it requires the alphabets of the processes being compared to be equal. In practice, model elaboration can lead to augmenting the alphabet of the system model to describe behavioural aspects that previously had not been taken into account. To capture this aspect of model elaboration, we introduce two concepts: hiding and observational refinement.

Hiding is an operation that makes a set of actions of a model unobservable to its environment by reducing the alphabet of the model and replacing transitions labelled with an action in the hiding set by τ, as shown in Figure 2.

$$\frac{M \xrightarrow{\ell}_\gamma M'}{(M \backslash X) \xrightarrow{\ell}_\gamma (M' \backslash X)} \begin{array}{l}\ell \notin X,\\ \gamma \in \{r, m\}\end{array} \qquad \frac{M \xrightarrow{\ell}_\gamma M'}{(M \backslash X) \xrightarrow{\tau}_\gamma (M' \backslash X)} \begin{array}{l}\ell \in X,\\ \gamma \in \{r, m\}\end{array}$$

Figure 2: Rules for the hiding operator.

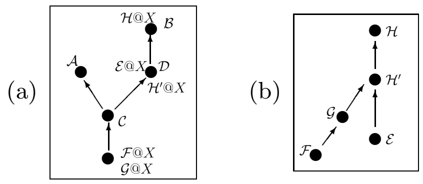

Figure 3: Observational refinements (without transitive relations) between models in Figure 1: (a): over the alphabet $X = \{$`acquireReadLock`, `releaseReadLock`$\}$; (b): over the alphabet $X \cup \{$`acquireWriteLock`, `releaseWriteLock`$\}$.

Definition 4. (Hiding) *Let $M = (S, L, \Delta^r, \Delta^p, s_0)$ be an MTS and $X \subseteq Act$ be a set of observable actions. M with the actions of X hidden, denoted $M \backslash X$, is an MTS $(S, L \backslash X, \Delta^{r'}, \Delta^{p'}, s_0)$, where $\Delta^{r'}$ and $\Delta^{p'}$ are the smallest relations that satisfy the rules in Figure 2. We use $M @ \alpha X$ to denote $M \backslash (Act \backslash X)$.*

Let $w = w_1, \ldots, w_k$ be a word over Act_τ. Then $M \xrightarrow{w}_r N$ means that there exist $M_0, \ldots, M_k$ such that $M_0 = M$, $M_k = N$, and $M_i \xrightarrow{w_{i+1}}_r M_{i+1}$ for $0 \le i < k$. We use $M \overset{\ell}{\Longrightarrow}_r M'$ to denote $M \xrightarrow{\tau^* \ell \tau^*}_r M'$. On the other hand, $M \xrightarrow{w}_m N$ means that there exist $M_0, \ldots, M_k$ such that $M_0 = M$, $M_k = N$, $M_i \xrightarrow{w_{i+1}}_p M_{i+1}$, for $0 \le i < k$, and $\exists j \cdot 0 \le j \le k \cdot M_j \xrightarrow{w_{j+1}}_m M_{j+1}$, i.e., there is at least one maybe transition on some letter of w. We use $M \overset{\ell}{\Longrightarrow}_m M'$ to denote $\exists M'' \cdot M \xrightarrow{\tau^* \ell}_m M''$ and $M'' \xrightarrow{\tau *}_r M'$, i.e., the maybe transition precedes ℓ on the path from M to M'. Finally, for $\gamma \in \{r, m, p\}$, we extend $\Longrightarrow_\gamma$ to words in the same way as we do for $\longrightarrow_\gamma$.

To compare a model with another one with an augmented alphabet, we must hide the additional actions in the second model and then use *observational refinement* – effectively, refinement that ignores differences in τ transitions.

Definition 5. (Observational Refinement) *N is an observational refinement of M, written $M \preceq_O N$, if $\alpha M = \alpha N$ and (M,N) is contained in some refinement relation $R \subseteq \wp \times \wp$ for which the following holds for all $\ell \in Act$:*

1. $(M \stackrel{\ell}{\Longrightarrow}_r M') \Rightarrow (\exists N' \cdot N \stackrel{\ell}{\Longrightarrow}_r N' \ \wedge \ (M',N') \in R)$
2. $(N \stackrel{\ell}{\Longrightarrow}_p N') \Rightarrow (\exists M' \cdot M \stackrel{\ell}{\Longrightarrow}_p M' \ \wedge \ (M',N') \in R)$

These conditions exclude the case in which N has a required transition on ℓ, whereas M has a maybe transition.

Consider again the MTSs shown in Figure 1. In model $\mathcal{E}$, if a process acquires the write lock, then the read lock cannot be acquired until the write lock is released. Note that this model does not indicate that processes are actually allowed to acquire the write lock, as these transitions are maybe. Hiding the actions `acquireWriteLock` and `releaseWriteLock`, results in an MTS just like $\mathcal{E}$ but with labels `acquireWriteLock`? and `releaseWriteLock`? changed to τ?. Furthermore, the resulting model is observationally refined by the MTS $\mathcal{D}$:

$$\mathcal{E}' = \mathcal{E} \setminus \{\texttt{acquireWriteLock}, \texttt{releaseWriteLock}\} \preceq_O \mathcal{D}$$

where the refinement relation is:

$$R = \{(\mathcal{E}'_0, \mathcal{D}_0), (\mathcal{E}'_1, \mathcal{D}_1), (\mathcal{E}'_2, \mathcal{D}_2), (\mathcal{E}'_3, \mathcal{D}_0)\}$$

In fact, $\mathcal{E}'$ is also a refinement of $\mathcal{D}$ via the inverse of R. We say that $\mathcal{E}'$ and $\mathcal{D}$ are *observationally equivalent*, written $\mathcal{E}' \equiv_O \mathcal{D}$.

Figure 3 depicts observational refinements that hold between models in Figure 1. Each graph relates models with the same alphabet: $X = \{\texttt{acquireReadLock}, \texttt{releaseReadLock}\}$ in Figure 3(a) and

$$Y = X \cup \{\texttt{acquireWriteLock}, \texttt{releaseWriteLock}\}$$

in Figure 3(b). Nodes with multiple labels indicate models that are observationally equivalent. Note that models $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ have the alphabet X. Consequently, they cannot be related through observational refinement to models with an augmented alphabet Y, i.e. $\mathcal{E}$, $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, and $\mathcal{H}'$. For this reason, $\mathcal{A}$ through $\mathcal{D}$ do not appear in Figure 3(b). However, these models can be related through observational refinement to $\mathcal{E}$, $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, and $\mathcal{H}'$ if the latter have their alphabets restricted to X, and are depicted in Figure 3(a).

3. MERGING MODELS

In this section, we introduce the notion of merging modal transition systems. Figure 4 provides an abstract summary of the concepts discussed in this section. In this figure, arrows depict observational refinements (i.e., an edge from P to Q indicates that Q is a refinement of P). All models are assumed to have the same alphabet, and transitive relations are not depicted.

The intuition we wish to capture by merging is that of augmenting the knowledge we have of the behaviour of a system by taking what we know from the two partial descriptions of the system. Clearly, the notion of refinement underlies this intuition as it captures the "more defined than" relation between two partial models. Hence, merging two models of the same system is about finding a common refinement for these models, i.e., finding a model that is more defined than both.

It is possible that the models being merged have different alphabets. Merging these models results in a model whose

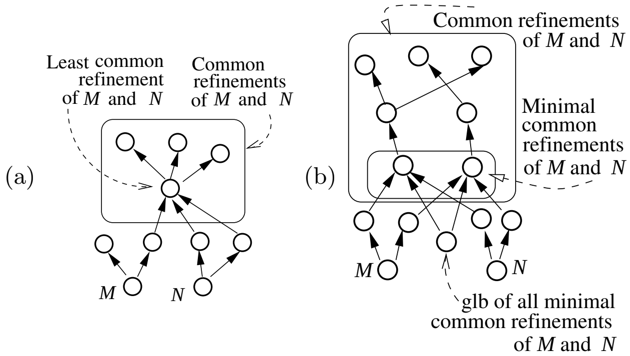

Figure 4: Building common refinement for models M and N: (a) M and N have the least common refinement; (b) M and N have no least common refinement.

alphabet is a superset of the original ones. Hence, the merge of two partial behavioural models should be an observational refinement of each with appropriately restricted alphabets.

Definition 6. (Common Observational Refinement) *A modal transition system P is a common refinement of modal transition systems M and N if $\alpha P \supseteq (\alpha M \cup \alpha N)$, $M \preceq_O P@\alpha M$ and $N \preceq_O P@\alpha N$.*

From this point on, when we refer to common refinement, we always mean observational refinement.

Refer to Figure 1 for the following example. The MTS $\mathcal{F}$ specifies the writers policy for acquiring a read-write lock: i) readers exclude writers, ii) writers exclude readers, iii) at most one writer can have the lock at any given time, iv) the number of concurrent readers allowed is not known. We can merge $\mathcal{F}$ with the MTS $\mathcal{D}$ that states that there can be at least two concurrent readers. Model $\mathcal{H}$ is a common refinement of these models. Note that $\mathcal{D} \preceq_O \mathcal{H} \setminus \{\texttt{acquireWriteLock}, \texttt{releaseWriteLock}\}$ holds via the relation

$$R = \{(\mathcal{D}_0, \mathcal{H}_0), (\mathcal{D}_1, \mathcal{H}_1), (\mathcal{D}_2, \mathcal{H}_2), (\mathcal{D}_0, \mathcal{H}_3)\}$$

and $\mathcal{F} \preceq_O \mathcal{H}$ holds via the relation

$$R = \{(\mathcal{F}_0, \mathcal{H}_0), (\mathcal{F}_1, \mathcal{H}_3), (\mathcal{F}_2, \mathcal{H}_2), (\mathcal{F}_2, \mathcal{H}_1)\}$$

$\mathcal{H}$ is a refinement of $\mathcal{D}$ and $\mathcal{F}$ and thus can simulate the required behavior of $\mathcal{D}$ and $\mathcal{F}$. For example, like $\mathcal{D}$, $\mathcal{H}$ allows up to two readers to access the lock concurrently, e.g., the trace

`acquireReadLock`, `acquireReadLock`,
`releaseReadLock`, `releaseReadLock`, ...

Like $\mathcal{F}$, $\mathcal{H}$ allows one writer to access the lock (e.g., the trace `acquireWriteLock`, `releaseWriteLock`, ...). On the other hand, $\mathcal{F}$ and $\mathcal{D}$ can simulate the possible behaviour of $\mathcal{H}$, i.e. $\mathcal{H}$ cannot introduce behaviour that is proscribed in $\mathcal{F}$ or $\mathcal{D}$. If $\mathcal{H}$ had (possible) traces allowing concurrent access to the lock by readers and writers, e.g., `acquireWriteLock`, `acquireReadLock`, ..., $\mathcal{H}$ would not be a refinement of $\mathcal{F}$ nor $\mathcal{D}$, as these are not possible in $\mathcal{F}$ or $\mathcal{D}$.

Note that common refinement allows $\mathcal{H}$ to proscribe traces that were possible in $\mathcal{F}$ or $\mathcal{D}$. In other words, $\mathcal{H}$ may not be able to simulate the possible behaviour of $\mathcal{D}$ and $\mathcal{F}$. For example, the trace

`acquireReadLock`, `acquireReadLock`, `acquireReadLock`, ...

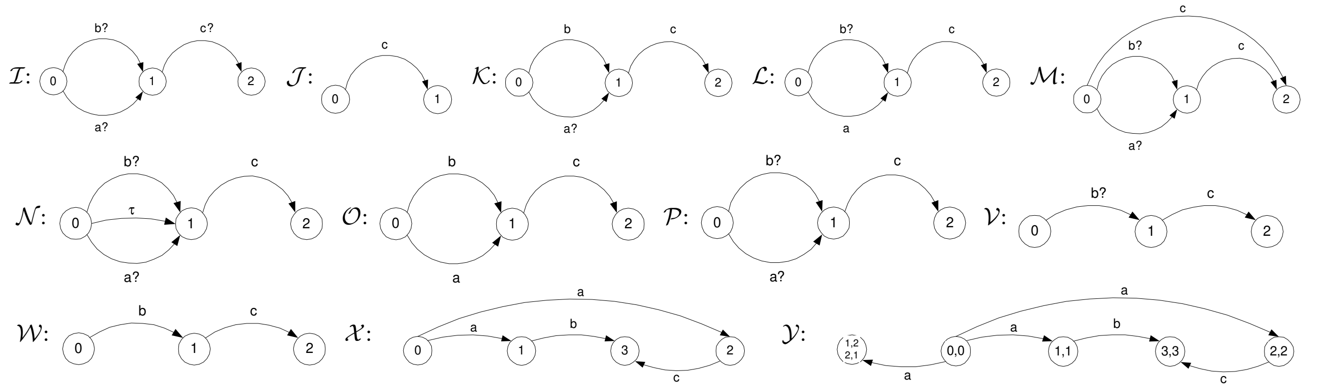

Figure 5: Example MTS $\mathcal{I}$ and $\mathcal{J}$; their minimal common refinements $\mathcal{K}$ and $\mathcal{L}$; models $\mathcal{M}$ and $\mathcal{N}$ which are not common refinements of $\mathcal{I}$ and $\mathcal{J}$; $\mathcal{O}$: the least upper bound of $\mathcal{I}$ and $\mathcal{J}$; $\mathcal{P}$: the greatest lower bound of minimal refinements of $\mathcal{I}$ and $\mathcal{J}$. $\mathcal{W}$: a minimal common refinement of $\mathcal{V}$ and $\mathcal{J}$. $\mathcal{X}$: a non-deterministic MTS and $\mathcal{Y}$: composition of $\mathcal{X}$ with itself that is not a refinement of $\mathcal{X}$.

(a) (b)

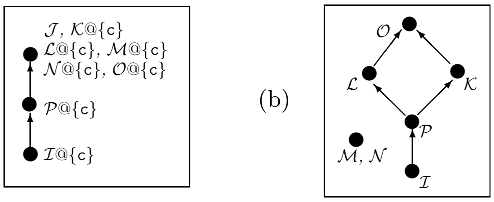

Figure 6: Relationships between some MTSs in Figure 5: (a) with respect to the alphabet {c}; (b) with respect to the alphabet {a, b, c}.

that allows three or more concurrent readers, is a possible trace of $\mathcal{F}$ and $\mathcal{D}$, but is proscribed in $\mathcal{H}$.

Consequently, the merged model $\mathcal{H}$ introduces knowledge that neither of the original models has, e.g., proscribing three or more concurrent readers. Instead, we prefer a less refined model $\mathcal{H}'$ that does not make such restrictions. $\mathcal{H}'$ is called the *least common refinement* of $\mathcal{D}$ and $\mathcal{F}$.

Definition 7. (Least Common Refinement) *A modal transition system P is the least common refinement of modal transition systems M and N if P is a common refinement of M and N, $\alpha P = \alpha M \cup \alpha N$, and for any common refinement Q of M and N, $P \preceq Q@\alpha P$.*

The merge of modal transition systems cannot be defined as their least common refinement for two reasons. Firstly, it is possible that there is no common refinement at all. Secondly, it is possible that a common refinement exists, but there is no least one. We discuss these possibilities below.

Consider again the models in Figure 1. Models $\mathcal{A}$ and $\mathcal{B}$ do not have a common refinement: suppose some model P is a common refinement of $\mathcal{A}$ and $\mathcal{B}$, with $\alpha P = \alpha\mathcal{A} = \alpha\mathcal{B}$, and w = `acquireReadlock`, `acquireReadlock`. We know that $\mathcal{B} \stackrel{w}{\Longrightarrow}_r \mathcal{B}_2$, and because $\mathcal{B} \preceq_O P$, then $\exists P'$ such that $P \stackrel{w}{\Longrightarrow}_r P'$. Since $\mathcal{A} \preceq_O P$, trace w should be possible in $\mathcal{A}$, which is a contradiction. In fact, model $\mathcal{A}$ is inconsistent with all models that give concurrent read access to more than one process, namely $\mathcal{B}$, $\mathcal{D}$, $\mathcal{E}$, $\mathcal{H}$ and $\mathcal{H}'$, as shown in Figure 3(a).

Definition 8. (Consistency) *Two MTSs M and N are* consistent *if there exists an MTS P such that P is a common refinement of M and N.*

Consistency does not guarantee the existence of the least common refinement. Consider the models shown in Figure 5. Models $\mathcal{I}$ and $\mathcal{J}$ do have common refinements, e.g., $\mathcal{K}$, $\mathcal{L}$ and $\mathcal{O}$, but no least common refinement. Intuitively, to find the least common refinement between $\mathcal{I}$ and $\mathcal{J}$, we must refine $\mathcal{I}$ into $\mathcal{I}'$ so that $(\mathcal{I}' \setminus \{\mathtt{a}, \mathtt{b}\}) \stackrel{\mathtt{c}}{\Longrightarrow}_r$. Hence, we must transform the maybe transition on c in $\mathcal{I}$ to a required transition and also transform one of the maybe transitions on a or b. If we transform all three transitions, we obtain the model $\mathcal{O}$. If we choose not to transform the transition either on a or on b, then we obtain the models $\mathcal{K}$ and $\mathcal{L}$. Note that these common refinements are not comparable (neither is a refinement of the other) because of the different choices made on which maybe transition to make required.

It is not possible to find common refinements of $\mathcal{I}$ and $\mathcal{J}$ which are less refined than $\mathcal{K}$ and $\mathcal{L}$. For example, $\mathcal{P}$ is less refined than both but is not a refinement of $\mathcal{J}$. Hence, we refer to $\mathcal{K}$ and $\mathcal{L}$ as the *minimal common refinements* of $\mathcal{I}$ and $\mathcal{J}$. Note that models $\mathcal{M}$ and $\mathcal{N}$ are incorrect attempts of building minimal common refinements of $\mathcal{I}$ and $\mathcal{J}$. These are not refinements of $\mathcal{I}$ because they both can transit on c from the initial state through (a sequence of) required transitions, while $\mathcal{I}$ cannot do so from its initial state.

Definition 9. (Minimal Common Refinement) *An MTSs P is a* minimal common refinement *of MTSs M and N if P is a common refinement of M and N, $\alpha P = \alpha M \cup \alpha N$; and there is no MTSs $Q \not\equiv P$ such that Q is a common refinement of M and N and $Q@\alpha P \preceq P$.*

Remark 1. *If P is the least common refinement of M and N, it is also a minimal common refinement. In addition, if P is the only minimal common refinement of M and N, then it is also their least common refinement. Finally, if M and N are consistent, then they have a minimal common refinement.*

If two models are consistent but have no least common refinement, then their merge could result in *any* of their minimal common refinements. However, any choice of minimal common refinement rules out the others! Hence, it is helpful to find a model that characterizes the point in which incompatible decisions must be made in order to merge two models and produce a minimal common refinement. This model is the greatest lower bound (glb) of all minimal common refinements. The glb is the most refined model from which we can arrive through refinement to any of the minimal common refinements. The glb always exists and is always unique with respect to observational equivalence. Note that the glb itself may not be a common refinement of the models being merged.

For example, the glb of the minimal common refinements of the models $\mathcal{I}$ and $\mathcal{J}$ (see Figure 5) is the model $\mathcal{P}$. Note that this model is not a refinement of $\mathcal{J}$, but could be refined to become one. Further, any refinement of this model rules out the possibility of arriving at one of the minimal refinements ($\mathcal{K}$ or $\mathcal{L}$).

The relationship between modes $\mathcal{I}$ through $\mathcal{N}$ is shown in Figure 6. As in Figure 3, each graph relates models with the same alphabets. Since $\mathcal{J}$ does not have a or b in its alphabet, it cannot be compared to the other models unless these have their alphabets restricted to $\alpha\mathcal{J}$. Hence, $\mathcal{J}$ does not appear in Figure 6(b), where arrows depict observational refinement between models over the alphabet $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. However, $\mathcal{J}$ does appear in Figure 6(a), where the compared models have the alphabet $\{\mathtt{c}\}$.

Definition 10. (Greatest lower bound) *Let M and N be consistent modal transition systems. We say that an MTS Q is a* lower bound *of all minimal common refinements of M and N if $\alpha Q = \alpha M \cup \alpha N$ and for any minimal common refinement P of M and N, it holds that $Q \preceq P$. We say that a lower bound of all minimal common refinements of M and N is the* greatest lower bound *(glb) if for any other lower bound Q', it holds that $Q' \preceq Q$.*

Remark 2. *If P is the least common refinement of M and N, then, by Remark 1, P is also the glb of all minimal common refinements of M and N.*

In conclusion, what should the result of merging two consistent modal transition systems M and N be? If M and N have the least common refinement, then this is the desired result of the merge. However, if M and N are consistent but do not have the least common refinement, then the merge process should result in one of the minimal common refinements of M and N. Model merging should support the modeller in choosing which minimal common refinement is the most appropriate. This can be done by producing the glb of all minimal common refinements of M and N and supporting its elaboration to produce a minimal common refinement. This would allow the modeller to choose, possibly after validating with stakeholders, which is the appropriate way of combining two different descriptions of the behaviour of the same system.

4. MERGE ALGORITHMS

In this section, we present several algorithms for computing the merge of two MTSs. The basis of our algorithms is the $+_u$ operator. We discuss the use of this operator for checking consistency and its limitations for building the least and the minimal common refinements. We also present the $+_l$ operator and show how it can be used to build the least common refinement or a lower bound to all minimal common refinements. In the latter case, we show how to support the elaboration to obtain a minimal common refinement.

4.1 Building a Common Refinement

We first introduce the $+_u$ operator and then show that assuming consistency and the determinacy condition, $M +_u N$ is a common refinement of M and N and also an upper bound to all minimal common refinements of M and N.

$$\text{TD}\ \frac{M \xrightarrow{\ell}_r M'}{M +_u N \xrightarrow{\ell}_r M' +_u N}\ \ell \notin \alpha N \qquad \text{DT}\ \frac{N \xrightarrow{\ell}_r N'}{M +_u N \xrightarrow{\ell}_r M +_u N'}\ \ell \notin \alpha M$$

$$\text{TM}\ \frac{M \xrightarrow{\ell}_r M',\ N \xrightarrow{\ell}_m N'}{M +_u N \xrightarrow{\ell}_r M' +_u N'}\ \ell \neq \tau \qquad \text{MT}\ \frac{M \xrightarrow{\ell}_m M',\ N \xrightarrow{\ell}_r N'}{M +_u N \xrightarrow{\ell}_r M' +_u N'}\ \ell \neq \tau$$

$$\text{DM}\ \frac{N \xrightarrow{\ell}_m N'}{M +_u N \xrightarrow{\ell}_r M +_u N'}\ \ell \notin \alpha M \qquad \text{MD}\ \frac{M \xrightarrow{\ell}_m M'}{M +_u N \xrightarrow{\ell}_r M' +_u N}\ \ell \notin \alpha N$$

$$\text{TT}\ \frac{M \xrightarrow{\ell}_r M',\ N \xrightarrow{\ell}_r N'}{M +_u N \xrightarrow{\ell}_r M' +_u N'}\ \ell \neq \tau \qquad \text{MM}\ \frac{M \xrightarrow{\ell}_m M',\ N \xrightarrow{\ell}_m N'}{M +_u N \xrightarrow{\ell}_m M' +_u N'}\ \ell \neq \tau$$

Figure 7: Rules for the $+_u$ operator.

Definition 11. (The $+_u$ Operator) *Let M and N be MTSs where $M = (S_M, L_M, \Delta^r_M, \Delta^p_M, s_{0_M})$ and $N = (S_N, L_N, \Delta^r_N, \Delta^p_N, s_{0_N})$. $M +_u N$ is an MTS $(S_M \times S_N, L_M \cup L_N, \Delta^r, \Delta^p, (s_{0_M}, s_{0_N}))$, where Δ^r and Δ^p are the smallest relations that satisfy the rules given in Figure 7.*

We explain the rules of Figure 7 below. Intuitively, the models being merged are run in parallel, synchronizing on shared actions and producing transitions in the merged model that amount to merging knowledge from both models. This means that maybe transitions in one model can be overridden by transitions that are known to be required or proscribed in the other. For instance, if M can transit on ℓ through a maybe transition, and N can do so via a required transition, then $M +_u N$ can transit on ℓ through a required transition as well, as indicated by rules TM and MT in Figure 7. If M can transit on ℓ through a maybe transition and N cannot transit on ℓ, then $M +_u N$ cannot transit on ℓ.

For cases in which there is an agreement between both models, the rules are as expected. If both M and N can transit on ℓ over required transitions, then $M +_u N$ can do so as well, as indicated by the rule TT in Figure 7. Maybe transitions in both models are treated similarly (see rule MM), and if neither models can transit on ℓ, then the composition cannot either.

The rules discussed so far create a composition that is a refinement of the original models: all required transitions are clearly preserved in the composition; furthermore, maybe transitions are introduced into the composition only if one of the original models has a maybe transition. We now address the problem of handling states in which the models disagree on whether the action is allowed or proscribed. Such states are *disagreement states*, formally defined below.

Definition 12. (Disagreement States) *Let M and N be MTSs where $M = (S_M, L_M, \Delta^r_M, \Delta^p_M, s_{0_M})$ and $N = (S_N, L_N, \Delta^r_N, \Delta^p_N, s_{0_N})$. We say that $(m, n) \in (S_M \times S_N)$ is a* disagreement state *if there exists a label $\ell \in (\alpha M \cap \alpha N)$ for which (1) $M_m \xrightarrow{\ell}_r$ and $N_n \not\xrightarrow{\ell}$ or (2) $M_m \not\xrightarrow{\ell}$ and $N_n \xrightarrow{\ell}_r$.*

Consider the models $\mathcal{I}$ and $\mathcal{J}$ of Figure 5. We have that $\mathcal{J} \xrightarrow{\mathtt{c}}_r \mathcal{J}_1$ but that $\mathcal{I} \not\xrightarrow{c}$. If we allow $\mathcal{I} +_u \mathcal{J}$ to transition on c, i.e., $(\mathcal{I} +_u \mathcal{J}) \xrightarrow{\mathtt{c}}_m$, then $(\mathcal{I} +_u \mathcal{J})@\alpha\mathcal{I}$ is not a refinement of $\mathcal{I}$. Instead, $(\mathcal{I} +_u \mathcal{J})$ should not have a transition on c. However, we must guarantee that $(\mathcal{I} +_u \mathcal{J})@\alpha\mathcal{J}$ is a refinement of $\mathcal{J}$. Hence, because $\mathcal{J} \xrightarrow{\mathtt{c}}_r \mathcal{J}_1$, $(\mathcal{I} +_u \mathcal{J})@\alpha\mathcal{J}$ should be able to transit through required τ transitions to a state in which it can then make a required transition on c ($(\mathcal{I} +_u \mathcal{J})@\alpha\mathcal{J} \stackrel{\mathtt{c}}{\Longrightarrow}_r$).

Special care needs to be taken when merging models with different alphabets. If a label ℓ does not belong to the alphabet of one of the models, this means that this model is not concerned with ℓ. Hence, if one of the models can transit on ℓ through a required transition, then the composition can do so too, but the state of the other model is unchanged, as ℓ is out of the scope. This is captured by TD and DT rules in Figure 7 ("D" stands for "do not care"). Similarly, if one model does not care about ℓ and the other cannot transit on it, then the composition should not either.

A similar reasoning could be applied to explain rules MD and DM in Figure 7; however, note that these rules state that if M can transit on ℓ through a *maybe* transition and ℓ is not in N's alphabet, then $M +_u N$ can transit on ℓ through a *required* transition rather than a *maybe*. For example, applying the $+_u$ operator to the models $\mathcal{I}$ and $\mathcal{J}$ in Figure 5, results in the model $\mathcal{O}$, which is their common refinement.

To prove that $+_u$ builds common refinements, we need to ensure that the models are consistent (see Section 4.2). We also must limit non-determinism in the models being composed. Consider the modal transition system $\mathcal{X}$ shown in Figure 5. Composing this model with itself should yield $\mathcal{X}$, whereas computing $\mathcal{Y} = \mathcal{X} +_u \mathcal{X}$ yields a model which is not a refinement of $\mathcal{X}$. The problem here is with the non-deterministic behaviour of $\mathcal{X}$. From state 0, $\mathcal{X}$ can transition on `a` to either state 1 or state 2, and transitions enabled and proscribed in these states are different. Our $+_u$ operator cannot cope with such non-deterministic choices, so for the remainder of this paper, we assume that the models being composed satisfy the *determinacy condition*, i.e. they do not produce a composite state where there is a non-deterministic choice on some label ℓ that leads to states that are not observationally equivalent.

DEFINITION 13. (Determinacy Condition) *Let* $M = (S_M, L_M, \Delta^r_M, \Delta^p_M, s_{0_M})$ *be an MTS. We say that* M_s *is* non-deterministic *on* ℓ *if there are states* q *and* r *in* M *such that* $M_s \xrightarrow{\ell}_p M_q$, $M_s \xrightarrow{\ell}_p M_r$, *and* $M_q \not\equiv M_r$.

Let $N = (S_N, L_N, \Delta^r_N, \Delta^p_N, s_{0_N})$ *be an MTS. We say that the* determinacy condition *holds for a composition* $C = (S_M \times S_N, L_C, \Delta^r_C, \Delta^p_C, (s_{0_M}, s_{0_N}))$ *if for all reachable states* (m, n) *of* C *and all labels* $\ell \in L_M \cap L_N$, *it is not the case that* M_m *and* N_n *are non-deterministic on* ℓ.

It is important to note that the determinacy condition is weaker than simply requiring MTSs to be deterministic. For example, in Figure 1, state 3 of $\mathcal{E}$ has a non-deterministic choice over label `releaseWriteLock`. However, in $\mathcal{E} +_u \mathcal{F}$, this state is reached when $\mathcal{F}$ is in state 1, which is deterministic on `releaseWriteLock`. The same happens with state 2 of $\mathcal{F}$ which is non-deterministic on `releaseReadLock`, but in $\mathcal{E} +_u \mathcal{F}$, this state is reached when $\mathcal{E}$ is in states 1 or 2, both of which are deterministic on `releaseReadLock`. Thus, $\mathcal{E} +_u \mathcal{F}$ satisfies the determinacy condition, so the merge is possible. On the other hand, $\mathcal{X} +_u \mathcal{X}$ does not satisfy it in state $(0, 0)$, and so the merge cannot be performed.

THEOREM 1. ($+_u$ builds common observational refinements) *If* M *and* N *are consistent modal transition systems and* $M +_u N$ *satisfies the determinacy condition, then* $M +_u N$ *is a common observational refinement of* M *and* N.

PROOF. The proof proceeds by showing that the rules TT, MM, MT, TM, DM, DT, MD, and DM in Figure 7 make the composition preserve required transitions of M and N, while the fact that no other rules are present ensures that M and N preserve the maybe transitions of the composition. There are two main points that the proof addresses: non-deterministic choice and disagreement states. Non-deterministic choice is handled the same way as proving that the parallel composition of deterministic processes can simulate the composed processes [18]. The proof that disagreement states still result in a common refinement even though no transitions are produced in this case by the rules in Figure 7, follows from consistency of M and N. Consistency guarantees that if a reachable state (m, n) has a disagreement on ℓ and $M_m \stackrel{\ell}{\not\longrightarrow}$, then M must be able to transit through maybe transitions on actions that are unobservable to N and reach a state from which it can transit on ℓ, i.e., $\exists w \in (Act_\tau \setminus \alpha N)^*$ and $\exists m'$ such that $M_m \stackrel{w}{\Longrightarrow}_p M_{m'}$ and $M_{m'} \xrightarrow{\ell}_r$. The MD rule converts maybe transitions that are unobservable to N into required transitions in the composite model, guaranteeing that $(M_m +_u N_n) \xrightarrow{\ell}_r$. □

Finally, we address the precision of the $+_u$ operator: this operator does not build minimal common refinements. For example, $\mathcal{I} +_u \mathcal{J} = \mathcal{O}$, whereas minimal common refinements of $\mathcal{I}$ and $\mathcal{J}$ are $\mathcal{K}$ and $\mathcal{L}$ (see Figure 5). In fact, $+_u$ produces an overapproximation, or an upper bound with respect to the refinement ordering, of all minimal common refinements of the composed models, which is reflected in its name: "u" in $+_u$ stands for upper bound. We address this problem in Section 4.3.

THEOREM 2. ($+_u$ is an upper bound of all minimal common refinements) *If* M *and* N *are consistent and* $M +_u N$ *satisfies the determinacy condition, then for every* Q *that is a minimal common refinement of* M *and* N, *it holds that* $Q@\alpha(M +_u N) \preceq_0 M +_u N$.

PROOF. Any minimal common refinement Q of models M and N differs from $M +_u N$ only from the result of applying MD and MD rules: some required transitions of $M +_u N$ can be maybe transitions in Q. □

4.2 Checking Consistency

We now present an algorithm for checking consistency of two MTSs under the assumption of the determinacy condition.

To check consistency of two MTSs M and N, we use the $+_u$ operator to build a composite model of M and N and meanwhile check what happens when a disagreement state is reached. The algorithm is shown in Figure 8.

THEOREM 3. (Consistency check algorithm is sound) *If* $M +_u N$ *satisfies the determinacy condition, Algorithm 1 called with parameters* M *and* N *returns null if and only if* M *and* N *are consistent.*

PROOF. The proof of the $\Leftarrow$ direction follows directly from Theorem 1. The proof of the $\Rightarrow$ direction is based on the similar reasoning as the proof of Theorem 1. □

Refer to MTSs in Figure 1. Algorithm 1, applied to inconsistent models $\mathcal{A}$ and $\mathcal{B}$, would return the pair (1, 2) signaling that this disagreement state (in which $\mathcal{A}$ proscribes the occurrence of `acquireReadLock` in state 1 while $\mathcal{B}$ has a required transition on the same label in state 2) is a source of inconsistency. If the algorithm is applied to models $\mathcal{G}$ and $\mathcal{B}$, then the pair (2, 1) would be returned, signaling an inconsistency based on the fact that $\mathcal{G}$ constrains the number

Algorithm 1. *(Consistency Check)*

Input: MTSs M and N.
Output: if M and N are consistent, return null; otherwise, return a disagreement state.

Build $M +_u N$, marking each disagreement state
For each marked state (m, n)
  If $N_n \stackrel{\ell}{\not\longrightarrow}$
    If $N_n \stackrel{w.\ell}{\not\longrightarrow}$ for some $w \in (Act_\tau \backslash \alpha M)^*$
      Return (m, n)
  If $M_m \stackrel{\ell}{\not\longrightarrow}$
    If $M_m \stackrel{w.\ell}{\not\longrightarrow}$ for some $w \in (Act_\tau \backslash \alpha N)^*$
      Return (m, n)
Return null

Figure 8: Algorithm for checking consistency between two MTSs.

$$\text{DM}\ \frac{N \xrightarrow{\ell}_m N'}{M +_l N \xrightarrow{\ell}_m M +_l N'}\ \ell \notin \alpha M \qquad \text{MD}\ \frac{M \xrightarrow{\ell}_m M'}{M +_l N \xrightarrow{\ell}_m M' +_l N}\ \ell \notin \alpha N$$

Figure 9: Two rules for the $+_l$ operator.

of readers to 1, while $\mathcal{B}$ allows two readers. Similarly, the algorithm would detect the inconsistencies between pairs ($\mathcal{G}$ and $\mathcal{D}$) and ($\mathcal{G}$ and $\mathcal{E}$).

4.3 Building a Lower Bound of the Merge

As we noted earlier, the operator $+_u$ computes a common refinement of consistent models M and N, but this refinement may not necessarily be minimal. Instead, we wish to provide an algorithm for constructing the least common refinement of M and N, if there is one, or the set of minimal common refinements otherwise. Here we present the $+_l$ operator and show that under the assumption of determinacy, it is a lower bound to all minimal common refinements. In Section 4.4, we discuss how to elaborate this model to become a minimal common refinement.

Recall that $+_u$ introduces imprecision through DM and MD rules, making the corresponding transition in $M +_u N$ *required*. Operator $+_l$, defined below, relaxes these restrictions.

Definition 14. (The $+_l$ Operator) *The $+_l$ operator is defined as $+_u$, but replacing MD and DM rules of Figure 7 with those in Figure 9.*

Computing $\mathcal{I} +_l \mathcal{J}$ for the models shown in Figure 5 yields $\mathcal{P}$, but as we discussed in Section 3, this model is not a refinement of $\mathcal{J}$. Neither of the maybe transitions on `a` or on `b` in $\mathcal{J}$ have been converted to required transitions. $+_l$ computes an underapproximation, or the lower bound (thus the meaning of "*l*" in its name) of all minimal refinements of the models being merged, as stated below.

Theorem 4. ($+_l$ is a lower bound of all minimal common refinements) *If M and N are consistent and $M +_l N$ satisfies the determinacy condition, then for any minimal common refinement Q of M and N, $M +_l N \preceq_0 Q@\alpha(M +_l N)$.*

Thus, $M +_l N$ approximates the greatest lower bound of M and N from below, and it would be reasonable to expect that our merge algorithms compute it and then help the modeller refine it into the minimal common refinement of his/her choice. Unfortunately, $+_l$ does not necessarily compute the actual glb. Consider merging the models $\mathcal{J}$ and $\mathcal{V}$ (see Figure 5). Their least common refinement is $\mathcal{W}$. However, $\mathcal{V} +_l \mathcal{J}$ results in $\mathcal{V}$ which is not a refinement of $\mathcal{J}$. The point is that DM and MD rules for the merge operator should convert some maybe transitions into required transitions. But which transitions should be converted? If all are, as in the computation of $+_u$, then minimality is lost. The *right* rules for computing glb for all minimal common refinements are somewhere between the MD and DM rules of $+_u$ and $+_l$. The choice of which transitions should be converted is discussed in Section 4.4.

Clearly, if the DM and MD rules are never applied, then $M +_u N = M +_l N$. Further, both operators produce the least common refinement of M and N, if one exists and the determinacy condition holds. In particular, when models being merged have the same alphabet and no maybe τ transitions, then DM and MD rules are not applied.

Theorem 5. (Sufficient condition for $+_l$ to be the least common refinement) *$M +_l N$ is the least common refinement of M and N if M and N are consistent, the determinacy condition holds for $M +_l N$, and MD and DM rules have not been used.*

In practice, we have found that the $+_l$ operator produces the least common refinement in many model merging contexts. In particular, it suffices for the readers and writers policies of Figure 1 and for the example in [21].

When none of the sufficient conditions of Theorem 5 hold, then the $+_l$ operator produces a model that can be refined into a minimal common refinement of both models by choosing an appropriate subset of maybe transitions generated by MD and DM rules and converting them into required transitions, as described below.

4.4 Elaboration

The goal of merging two consistent models is to arrive at their minimal common refinement. We now show how to refine a lower bound of all minimal common refinements, obtained via $+_l$, into a minimal common refinement.

The reason why $+_l$ operator builds a lower bound to all minimal common refinements and may not even be a common refinement itself is that DM and MD rules result in maybe transitions, whereas sometimes required transitions should be produced instead. Suppose $M +_N Q$ is being constructed and disagreement state (m, n) is reached. Without loss of generality, assume that $M_m \xrightarrow{\ell}_r N_{n'}$ whereas $N_n \stackrel{\ell}{\not\longrightarrow}$. Since M and N are assumed to be consistent, there exists $w \in (\alpha N \backslash \alpha M)^*$ and states n' and n'' of N such that $N_n \stackrel{w}{\Longrightarrow}_m N_{n'}$ and $N_{n'} \xrightarrow{\ell}_m N_{n''}$. We know from the DM rule that $(M_m +_l N_n) \stackrel{w}{\Longrightarrow}_m (M_m +_l N_{n'})$ and from rule TM – that $(M_m +_l N_{n'}) \xrightarrow{\ell}_r (M_{m'} +_l N_{n''})$. Hence, $(M_m +_l N_n)\backslash\alpha M \stackrel{\ell}{\Longrightarrow}_m (M_{m'} +_l N_{n''})\backslash\alpha M$. However, to obtain a minimal common refinement, because $M_m \xrightarrow{\ell}_r M_{m'}$, we need to obtain $(M_m +_l N_n)\backslash\alpha M \stackrel{\ell}{\Longrightarrow}_r (M_{m'} +_l N_{n''})\backslash\alpha M$. In cases where the least common refinement exists, only one w can be produced, and this transformation can be fully automated. In cases where more than one w satisfies the above condition, several minimal common refinements are possible. To obtain one, we need to pick a w from the above set and transform $(M_m +_l N_n) \stackrel{w}{\Longrightarrow}_m (M_m +_l N_{n'})$

Algorithm 2. *(Elaboration Algorithm)*

Input: Consistent MTSs M and N that satisfy the determinacy condition.

Output: A minimal common refinement P of M and N and a boolean flag set to true iff P is their least common refinement.

1: f := true
2: Build $P = M +_l N$, marking each disagreement state
3: For each marked state (m, n)
4: If $N_n \not\xrightarrow{\ell}$
5: Build set T with all w where $w \in (\alpha N \backslash \alpha M)^*$ such that $\exists N_{n'}$ where $N_n \stackrel{w}{\Longrightarrow}_m N_{n'}$ and $N_{n'} \xrightarrow{\ell}_m N_{n''}$
6: If $|T| > 1$
7: f := false /* No least common refinement*/
8: w' := User choice of the element of T
9: Else
10: $w' := w \in T$ /* this w is unique */
11: Replace maybe transitions in P with required transitions such that $(M_m +_l N_n) \stackrel{w}{\Longrightarrow}_r (M_m +_l N_{n'})$
12: Else /* $M_m \not\xrightarrow{\ell}$ */
13: Treatment is similar to the above case
14: Return f and P

Figure 10: An Elaboration Algorithm.

into $(M_m +_l N_n) \stackrel{w}{\Longrightarrow}_r (M_m +_l N_{n'})$. Further, different w traces can be used to give feedback to the user to support him/her in choosing the minimal common refinement that is most appropriate for the problem being modelled. The elaboration algorithm is shown in Figure 10.

Note that we have to be careful in step 5 of the algorithm: potentially, there can be an infinite number of w traces from which a modeller could choose if there are maybe loops in N. An implementation of this elaboration process needs to provide a subset of such traces, in particular, by putting a bound on the number of times a maybe loop is taken. In fact, it is likely that once a disagreement state with several refinement options has been identified, the modeller would want to evaluate the options by inspecting or animating the models. Hence, step 5 could be simplified to checking whether the disagreement state has more than one refinement option, rather than computing all the options, and then requesting the user to convert one maybe transition to a required one.

An execution of the algorithm for models $\mathcal{I}$ and $\mathcal{J}$ of Figure 5 identifies the pair (0, 0) as a disagreement state on action `c`, and displays `a`? and `b`? as the options for refining the composition to achieve a minimal common refinement. Depending on the choice, either model $\mathcal{K}$ or $\mathcal{L}$ would be reached.

We now look at complexity of our algorithms for merging models M and N with S_M and S_N states and T_M and T_N transitions (T_i is $O(S_i \times L_i)$). The potential size of the state space of a minimal common refinement of M and N is $S = O(|S_M| \times |S_N|)$. Checking whether M and N are consistent is very similar to checking weak bisimulation, and takes $O(L + S \times T)$ [1], where T is the number of transitions and $L = |L_M \cup L_N|$ is the total number of actions in the merged model. Computing $+_u$ and $+_l$ does not increase this complexity. Finally, we analyze complexity of the elaboration algorithm. Step 5 of the algorithm can produce an exponential number of w, even if the number of times each maybe loop is N traversed a finite number of times. In fact, since these words are to be displayed to the user, it does not make sense to compute more than a few different w's. In this case, step 5 can be done by breadth-first search in the τ-graph of N, taking $O(T_N)$.

5. DISCUSSION AND RELATED WORK

Although our work discusses merging of MTSs [15], this notion is applicable to other partial behavioural models such as Partial Labelled Transition Systems (PLTS) [22], multi-valued state machines [6], and Mixed Transition Systems [5]. The underlying principle of all of these should be common observational refinement, although the exact definition of merge operators will differ according to the specific characteristics of each formalism.

To the best of our knowledge, there is no prior work specifically on merging models that describe the observable behaviour of a system. On the other hand, merging operational specifications in which system states are explicitly described is frequently done (e.g. [24, 19, 7, 10]). In [24], states are modelled as valuations of state propositions, and states with compatible valuations can be merged. In [7], states can be merged only if they have the same label. [19] proposes a more general approach, but the emphasis of this work is on preserving model *structure* (i.e., the states and the accessibility relation between them) rather than preserving *behavioural properties*. Our approach differs from the work on computing least common generalizations from examples, as merge preserves simulation (which is stronger than trace inclusion) on required transitions and (in the opposite direction) on possible transitions. Hussain and Huth [10] also study the problem of finding a common refinement between multiple MTSs, focusing on the complexity of the relevant model-checking decision procedures. Instead, we address the more general problem of supporting engineering activities in model elaboration; we see merging as the process of selecting the *most appropriate* common refinement. In addition, we consider merging models with different alphabets. The goal of the work by Larsen et al. [16] is to decompose a complete specification into several partial ones to enable compositional proofs. In doing so, they define a sufficient condition for constructing common refinements in MTSs with the same alphabet. Their condition is more restrictive than our determinacy condition.

Our work focuses on operational descriptions. However, an alternative approach is to specify observable behaviour declaratively [12]. Declarative specifications based on classical logics are partial, yet they do not need to describe the unknown properties explicitly: such properties are those for which neither truth nor falsity can be inferred from the rest of the specification. Merging declarative behavioural specifications comes naturally as the conjunction of the corresponding theories; however, understanding which behaviours are possible is fairly difficult. Further, as in the case of operational descriptions, not all pairs of models have the least common refinement. Thus, some support for constructing an approximation of minimal common refinements and elaborating it into a desired minimal refinement is needed. To the best of our knowledge, there are no approaches that provide such support.

Larsen and Thomsen [15] define a parallel composition operator over MTSs. Its intent is different from the ones presented in this paper. Parallel composition assumes that models being composed describe different systems, whereas

$$\text{MT}\ \frac{M \xrightarrow{\ell}_m M',\ N \xrightarrow{\ell}_r N'}{M \| N \xrightarrow{\ell}_m M' \| N'}\ \ell \neq \tau \qquad \text{TM}\ \frac{M \xrightarrow{\ell}_r M',\ N \xrightarrow{\ell}_m N'}{M \| N \xrightarrow{\ell}_m M' \| N'}\ \ell \neq \tau$$

Figure 11: Two rules for the parallel composition operator.

merging treats those as different models of the same system. Note the difference between the TM and MT rules for parallel composition, shown in Figure 11, with those for $+_u$ and $+_l$: the combined model has a maybe transition in the former case and a required transition in the latter.

It is also important to note that our approach does not address ontological issues regarding the labels used in models being merged. Here we assume that labels have been used consistently according to one common ontology.

6. SUMMARY AND FUTURE WORK

The motivation for the work presented in this paper comes from the need to support the elaboration of partial behavioural models. In particular, our work has been motivated by existing limitations of scenario-based model synthesis techniques, hence the focus on observable behaviour rather than on the model structure. However, our work could also be applicable in the context of composing models that cover different viewpoints [9] or aspects [4].

We have argued that observational refinement is the formal underlying principle of model merging of partial behavioural models and that merging is a process that should produce a minimal common observational refinement of two consistent models. Modulo the determinacy condition, we have presented an algorithm for checking model consistency and algorithms for supporting the merge process. For the case in which there is only one minimal common refinement (i.e. the least common refinement exists), we have presented an algorithm that can build it automatically. For the other case, we have presented an algorithm that computes the lower bound of all minimal common refinements, which can then be elaborated by the modeller into the desired minimal common refinement.

In the near future, we expect to work on the efficiency of the algorithms presented in this paper and produce implementations for them. Experimentation using model merging remains to be done, and we aim to apply the algorithms in the context which has motivated our work, namely, scenario-based model synthesis and elaboration. We also intend to work on ways in which the determinacy condition for merging behavioural models can be weakened.

ACKNOWLEDGMENTS

We thank Arie Gurfinkel and Shiva Nejati for their comments on an earlier draft of this paper and also Michael Huth for his comments on this line of work. We acknowledge EPSRC grant READS GR/S03270/01 and NSERC for partially funding this work.

7. REFERENCES

[1] A. Bouali and R. de Simone. "Symbolic Bisimulation Minimization". In *Proceedings of CAV'92*, pages 96–108, 1992.

[2] M. Chechik, B. Devereux, S. Easterbrook, and A. Gurfinkel. "Multi-Valued Symbolic Model-Checking". *ACM TOSEM*, 12(4):1–38, 2003.

[3] E. Clarke and J. M. Wing. "Formal Methods: State of the Art and Future Directions". *ACM Computing Surveys*, 28(4):626–643, 1996.

[4] S. Clarke and R. J. Walker. "Composition Patterns: An approach to Designing Reusable Aspects". In *Proceedings of ICSE'01*, pages 5–14, May 2001.

[5] D. Dams. *Abstract Interpretation and Partition Refinement for Model Checking*. PhD thesis, Eindhoven University of Technology, 1996.

[6] R. Diaz-Redondo, J. Pazos-Arias, and A. Fernandez-Vilas. "Reusing Verification Information of Incomplete Specifications". In *Proceedings of the Workshop on Component-Based SE*, 2002.

[7] S. Easterbrook and M. Chechik. "A Framework for Multi-Valued Reasoning over Inconsistent Viewpoints". In *Proceedings of ICSE'01*, pages 411–420, May 2001.

[8] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, Englewood Cliffs, New Jersey, 1985.

[9] A. Hunter and B. Nuseibeh. "Managing Inconsistent Specifications: Reasoning, Analysis and Action". *ACM TOSEM*, 7(4):335–367, 1998.

[10] A. Hussain and M. Huth. "On Model Checking Multiple Hybrid Views". Submitted for publication, July 2004.

[11] ITU. "Message Sequence Charts". Technical Report Recommendation Z.120, International Telecommunications Union. Telecommunication Standardisation Sector, 2000.

[12] D. Jackson. "Alloy: a Lightweight Object Modelling Notation". *ACM TOSEM*, 11(2):256–290, 2002.

[13] R. Keller. "Formal Verification of Parallel Programs". *Communications of the ACM*, 19(7):371–384, 1976.

[14] I. Krueger, R. Grosu, P. Scholz, and M. Broy. "From MSCs to Statecharts". In F. J. Rammig, editor, *Distributed and Parallel Embedded Systems*. Kluwer Academic Publishers, 1999.

[15] K. Larsen and B. Thomsen. "A Modal Process Logic". In *Proceedings of LICS'88*, pages 203–210, 1988.

[16] K. G. Larsen, B. Steffen, and C. Weise. "A Constraint Oriented Proof Methodology based on Modal Transition Systems". In *Proceedings of TACAS'95*, pages 13–28, May 1995.

[17] J. Magee and J. Kramer. *Concurrency: State Models and Java Programs*. John Wiley & Sons Ltd., New York, 1999.

[18] R. Milner. *Communication and Concurrency*. Prentice-Hall, London, 1989.

[19] M. Sabetzadeh and S. Easterbrook. "Analysis of Inconsistency in Graph-Based Viewpoints: A Category-Theoretic Approach". In *Proceedings of ASE'03*, pages 12–21, October 2003.

[20] ICSE Workshop on Scenarios and State Machines: Model, Algorithms and Tools (SCESM), 2002-2004.

[21] S. Uchitel and M. Chechik. "Merging MTSs for a B2B E-Commerce Site", `http://www.doc.ic.ac.uk/~su2/merge/examples`.

[22] S. Uchitel, J. Kramer, and J. Magee. "Behaviour Model Elaboration using Partial Labelled Transition Systems". In *Proceedings of ESEC/FSE'03*, pages 19–27, 2003.

[23] S. Uchitel, J. Kramer, and J. Magee. "Synthesis of Behavioural Models from Scenarios". *IEEE TSE*, 29(2):99–115, 2003.

[24] J. Whittle and J. Schumann. "Generating Statechart Designs from Scenarios". In *Proceedings ICSE'00*, pages 314–323, 2000.