Properties of Behavioural Model Merging

Greg Brunet[1], Marsha Chechik[1], and Sebastian Uchitel[2]

[1] Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S2E4
{gbrunet, chechik}@cs.toronto.edu

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

Abstract. Constructing comprehensive operational models of intended system behaviour is a complex and costly task. Consequently, practitioners adopt techniques that support partial behaviour decription such as scenario-based specifications, and focus on elaborating these descriptions iteratively. In previous work, we show how this process can be formally supported by Modal Transition Systems (MTSs), observational refinement, and model merging. In this paper, we study a number of properties of merging MTSs and give insights on the implications these results have on engineering and reasoning about behaviour models. We illustrate the utility of our results on a case study.

1 Introduction

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

The problem is that state-based models, e.g., labelled transition systems (LTSs) [16], 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, particularly if state-based modeling is to be adopted in iterative development processes [2], processes that adopt use-case and scenario-based specifications (e.g., [6]), or that are viewpoint-oriented [12].

In such development contexts, a more appropriate type of state-based model is one in which currently unknown aspects of behavior are explicitly modelled, distinguishing between positive, negative and unknown behaviours. Positive behaviours are those the system is expected to exhibit; negative behaviours are those the system is expected to never exhibit; unknown behaviours 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*. A number of such models exist and promising results on their use to support incremental modelling and viewpoint analysis has been reported (e.g. Partial Labelled Transition Systems (PLTSs) [28], Modal Transition Systems (MTSs) [20,27], Mixed Transition Systems [8] and multi-valued Kripke structures [5]).

Our work focuses on MTSs. These models have been studied in depth (e.g. [20,15]) and are equipped with a notion of refinement that captures the idea of elaboration of a partial description into a more comprehensive one, in which some knowledge about the unknown behaviour of the system has been gained and modeled as either positive or negative behavior.

A logical extension to the notion of refinement is that of *model merging* — a process that allows integration of what is known about the behaviour of a system as modelled by different MTSs. Model merging supports putting together partial behavioural descriptions of the same system but given from two different perspectives, possibly by different stakeholders with different viewpoints [12], describing different, yet overlapping, aspects of the same system.

In our previous work [27], we introduce the notion of merging and argue that the core concept underlying model merging is that of common observational refinement. Note that composition of behavioural models is not a novel idea (e.g. [23]); however, focus has been on *parallel* composition which describes how two *different* components work together. In the context of model elaboration, we are interested in composing two partial descriptions of the *same* component to obtain a more elaborate version of both original partial descriptions.

In this paper, we aim to provide the necessary support for using merge in practice by answering several fundamental questions: When can two systems be merged? When is merge unique (i.e., when can the merging process be automated)? What kinds of properties are preserved in a merge? How can complex models be merged?

The rest of this paper is organized as follows. After reviewing the background material in Section 2, we provide, in Section 3, conditions for existence and uniqness of merge. We study algebraic properties of merging in Section 4, both for the case when a unique merge exists and for the case when it yields one of several possible merges, and apply results of this paper to a case study in Section 5. We conclude with a discussion, summary, and directions for future research in Section 6. Proofs of the results in this paper are available in [3].

2 Background

In this section, we briefly review definitions of MTSs, define a 3-valued counterpart to weak μ-calculus, review definitions of merge, and fix the notation. For detailed explanations and discussion, refer to [27,3].

2.1 Transition Systems

We use the standard notion of labelled transition systems (LTS) and their extensions, modal transition systems (MTSs), which capture partial behavior.

Definition 1. *Let* States *be a universal set of states, Act be a universal set of observable action labels, and $Act_\tau = Act \cup \{\tau\}$. An* 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, and $s_0 \in S$ is the initial state. We use $\alpha P = L \setminus \{\tau\}$ to denote the communicating alphabet (vocabulary) of P.*

Definition 2. *An* 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 its* possible *(but not necessarily required) transitions. We use $\alpha M = L \setminus \{\tau\}$ to denote the communicating alphabet of M.*

Given an MTS $M = (S,\ L,\ \Delta^r,\ \Delta^p,\ s_0)$, M has a required transition on ℓ (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 has a maybe transition on ℓ ($M \xrightarrow{\ell}_m M'$) if $(s_0, \ell, s_0') \in \Delta^p - \Delta^r$. $M \xrightarrow{\ell}_p M'$ means $(s_0, \ell, s_0') \in \Delta^p$. For an MTS M, M_n denotes changing the initial state to n. For $\gamma \in \{r, p\}$, we write $M \overset{\epsilon}{\Longrightarrow}_\gamma M'$ to denote $M(\xrightarrow{\tau}_\gamma)^* M'$, and $M \overset{\epsilon}{\Longrightarrow}_m M'$ to denote $M(\overset{\epsilon}{\Longrightarrow}_p)(\xrightarrow{\tau}_m)(\overset{\epsilon}{\Longrightarrow}_p)M'$, i.e., there is at least one maybe transition on τ. For $\ell \neq \tau$ and $\gamma \in \{r, p\}$, we write $M \overset{\ell}{\Longrightarrow}_\gamma M'$ to denote $M(\overset{\epsilon}{\Longrightarrow}_\gamma)(\xrightarrow{\ell}_\gamma)(\overset{\epsilon}{\Longrightarrow}_\gamma)M'$, and $M \overset{\ell}{\Longrightarrow}_m M'$ to denote $M(\overset{\epsilon}{\Longrightarrow}_m)(\xrightarrow{\ell}_p)(\overset{\epsilon}{\Longrightarrow}_p)M'$ or $M(\overset{\epsilon}{\Longrightarrow}_p)(\xrightarrow{\ell}_m)(\overset{\epsilon}{\Longrightarrow}_p)M'$, i.e., the maybe transition precedes or is on ℓ along the path from M to M'. For $\ell \in Act_\tau$, let $\hat{\ell} = \ell$ if $\ell \neq \tau$ and $\hat{\ell} = \epsilon$ if $\ell = \tau$. For $\gamma \in \{r, m, p\}$ and $\ell \in Act_\tau$, we often write $s \xrightarrow{\ell}_\gamma s'$ to mean $M_s \xrightarrow{\ell}_\gamma M_{s'}$ and similarly for $\Longrightarrow_\gamma$. Transitions on the thick arrow $\Longrightarrow_\gamma$ are referred to as *observable* transitions.

Figure 3 depicts two MTS models,$\mathcal{A}$ and $\mathcal{B}$. The initial state of an MTS is labeled 0, unless stated otherwise, and maybe transitions are denoted with a question mark following the label, and transitions on sets are short for a single transition on every element of the set. Note that all transitions in model $\mathcal{B}$ are required transitions, and is thus an LTS as well.

We capture the notion of elaboration of a partial description into a more comprehensive one using *observational refinement*:

Definition 3. *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 $\ell \in Act_\tau$:*

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

For example, $\mathcal{A} \preceq_o \mathcal{B}$ (see Figure 3) because $\mathcal{B}$ preserves the required behaviour of $\mathcal{A}$, and $\mathcal{A}$ can simulate the possible behaviour of $\mathcal{B}$.

In this paper, we use refinement to mean observational refinement, unless otherwise stated. Two models are *observationally equivalent* ($\equiv_o$) if they refine each other. We denote by $M@X$ the result of restricting αM to X, i.e., replacing actions in $Act \backslash X$ with τ and reducing αM to X.

$$\text{TD}\ \frac{M \xrightarrow{\ell}_r M'}{M\|N \xrightarrow{\ell}_r M'\|N}\ \ell \notin \alpha N \qquad \text{MT}\ \frac{M \xrightarrow{\ell}_m M',\ N \xrightarrow{\ell}_r N'}{M\|N \xrightarrow{\ell}_m M'\|N'}\ \ell \neq \tau \qquad \text{MD}\ \frac{M \xrightarrow{\ell}_m M'}{M\|N \xrightarrow{\ell}_m M'\|N}\ \ell \notin \alpha N$$

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

Fig. 1. Rules for parallel composition

Definition 4. *Let M and N be MTSs where $M = (S_M,\ L_M,\ \Delta^r_M,\ \Delta^p_M,\ s_{0M})$ and $N = (S_N,\ L_N,\ \Delta^r_N,\ \Delta^p_N,\ s_{0N})$. Then* parallel composition *($\|$) is a symmetric operator such that $M\|N$ is an MTS $(S_M \times S_N,\ L_M \cup L_N,\ \Delta^r,\ \Delta^p,\ (s_{0M}, s_{0N}))$, where Δ^r and Δ^p are the smallest relations that satisfy the rules given in Figure 1.*

2.2 The Logic $\mathcal{L}^w_\mu$

While shown in [15] to characterize strong refinement, the 3-valued counterpart to μ-calculus ($\mathcal{L}_\mu$) [17] is not well-suited for describing the observable behaviour of an MTS and does not characterize *observational* refinement, because it makes no distinction between an observable action and τ. Instead, we define a 3-valued extension of *weak μ-calculus* ($\mathcal{L}^w_\mu$), which does make such a distinction. The 2-valued version of this logic has been shown to be a useful logic for expressing properties of LTSs in [26].

3-valued $\mathcal{L}^w_\mu$ enables a formula to evaluate to **t** (*true*), **f** (*false*), or $\bot$ (*maybe*). For a set of fixed point variables *Var*, $a \in Act_\tau$ and $Z \in Var$, an $\mathcal{L}^w_\mu$ formula ϕ has the grammar $\phi \triangleq \mathbf{t} \mid \mathbf{f} \mid \bot \mid Z \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a\rangle_o\phi \mid [a]_o\phi \mid \mu Z.\phi \mid \nu Z.\phi$, where $\langle a\rangle_o$ and $[a]_o$ are the *next* operators with intended meanings "exists a next state reachable via an observable transition on a" and "for all next states reachable via an observable transition on a", respectively. We write $\phi(Z)$ to denote a formula that might contain free occurrences of the variable Z. μ and ν represent the least and greatest fixed points, respectively.

Let ϕ be a formula in $\mathcal{L}^w_\mu$, $M = (S_M, L_M, \Delta^r_M, \Delta^p_M, s_0)$ be an MTS, and $e_1, e_2 : Var \rightarrow \mathcal{P}(S_M)$ be *environments* mapping fixed point variables to sets of states. $[\![\phi]\!]^\top_{e_1}$ ($[\![\phi]\!]^\bot_{e_2}$) denotes the set of states in M where ϕ is *true* (*false*). The set of states where ϕ is *maybe* is then $S_M \backslash ([\![\phi]\!]^\top_{e_1} \cup [\![\phi]\!]^\bot_{e_2})$ (i.e., ϕ is not *true* or *false*).

Definition 5. (3-valued Semantics of $\mathcal{L}^w_\mu$) *For an MTS M, a formula ϕ in $\mathcal{L}^w_\mu$, and environments e_1 and e_2, $[\![\phi]\!]^\top_{e_1} \subseteq S_M$ and $[\![\phi]\!]^\bot_{e_2} \subseteq S_M$ are defined as shown in Figure 2, where $a \in \mathrm{Act}_\epsilon$, and $e_i[Z \rightarrow S]$ is the same environment as e_i except it maps Z to S.*

$\phi_1 \vee \phi_2$, $[a]_o\phi$ and $\nu Z.\phi(Z)$ are defined through negation: $\phi_1 \vee \phi_2 = \neg\phi_1 \wedge \neg\phi_2$, $[a]_o\phi = \neg\langle a\rangle_o\neg\phi$, and $\nu Z.\phi(Z) = \neg\mu Z.\phi(\neg Z)$. The value of ϕ in M is its value in the initial state. We omit the environments from $[\![\phi]\!]^\top$ and $[\![\phi]\!]^\bot$ to mean that e_1 and e_2 map every Z in *Var* to $\emptyset$ and S_M, respectively.

For example, the property $\langle a\rangle_o\mathbf{t}$ (which expresses the ability to perform an *observable* transition on a) evaluates to *true* in both $\mathcal{A}$ and $\mathcal{B}$ in Figure 3, even

$$\begin{array}{rlrl}
[\![\mathbf{t}]\!]^{\top}_{e_1} & \triangleq S_M & [\![\varphi \wedge \psi]\!]^{\top}_{e_1} & \triangleq [\![\varphi]\!]^{\top}_{e_1} \cap [\![\psi]\!]^{\top}_{e_1} \\
[\![\mathbf{t}]\!]^{\perp}_{e_2} & \triangleq \emptyset & [\![\varphi \wedge \psi]\!]^{\perp}_{e_2} & \triangleq [\![\varphi]\!]^{\perp}_{e_2} \cup [\![\psi]\!]^{\perp}_{e_2} \\
[\![\perp]\!]^{\top}_{e_1} & \triangleq \emptyset & [\![\langle a \rangle_o \phi]\!]^{\top}_{e_1} & \triangleq \{s \in S_M \mid \exists s' \in S_M \cdot (s \overset{a}{\Longrightarrow}_r s' \wedge s' \in [\![\phi]\!]^{\top}_{e_1})\} \\
[\![Z]\!]^{\top}_{e_1} & \triangleq e_1(Z) & [\![\langle a \rangle_o \phi]\!]^{\perp}_{e_2} & \triangleq \{s \in S_M \mid \forall s' \in S_M \cdot (s \overset{a}{\Longrightarrow}_p s' \Rightarrow s' \in [\![\phi]\!]^{\perp}_{e_2})\} \\
[\![Z]\!]^{\perp}_{e_2} & \triangleq e_2(Z) & [\![\mu Z.\phi(Z)]\!]^{\top}_{e_1} & \triangleq \cap\{S \subseteq S_M \mid [\![\phi]\!]^{\top}_{e_1[Z \to S]} \subseteq S\} \\
[\![\neg\phi]\!]^{\top}_{e_1} & \triangleq [\![\phi]\!]^{\perp}_{e_2} & [\![\mu Z.\phi(Z)]\!]^{\perp}_{e_2} & \triangleq \cap\{S \subseteq S_M \mid [\![\phi]\!]^{\perp}_{e_2[Z \to S]} \subseteq S\}
\end{array}$$

Fig. 2. 3-valued semantics of $\mathcal{L}^w_\mu$

though the transition on a in $\mathcal{B}$ is preceded by a τ. Additionally, the property $[a]_o\langle b \rangle_o\mathbf{t}$ evaluates to *maybe* in $\mathcal{A}$ because $\mathcal{A}_0 \xrightarrow{a}_r \mathcal{A}_1$ is the only transition on a from the initial state and $\mathcal{A}_1 \xrightarrow{b}_m \mathcal{A}_1$ is the only transition on b from $\mathcal{A}_1$.

The logic $\mathcal{L}^w_\mu$ characterizes observational refinement. In the 3-valued world, this means that an MTS M is refined by an MTS N if and only if all *true* and *false* $\mathcal{L}^w_\mu$ properties in M are preserved in N.

Theorem 1. *If M and N are MTSs with $\alpha M = \alpha N$, then:*
$M \preceq_o N \Leftrightarrow \forall \phi \in \mathcal{L}^w_\mu \cdot (s_{0M} \in [\![\phi]\!]^{\top} \Rightarrow s_{0N} \in [\![\phi]\!]^{\top}) \wedge (s_{0M} \in [\![\phi]\!]^{\perp} \Rightarrow s_{0N} \in [\![\phi]\!]^{\perp})$

Finally, if M is an LTS, the semantics in Definition 5 reduces to the standard 2-valued semantics in [26].

2.3 Merging Models

The intuition behind merge is to find a more precise system by combining what is known from two partial descriptions of that system. This is a process aimed at finding a common *observational* refinement, and may require human intervention [27]. We review this process below.

Definition 6. *An MTS P is a* common refinement (CR) *of MTSs M and N if $\alpha P \supseteq (\alpha M \cup \alpha N)$, $M \preceq_o P@\alpha M$ and $N \preceq_o P@\alpha N$.*

We denote the set of CRs of models M and N by $\mathcal{CR}(M, N)$. Two MTSs, M and N, are *consistent* iff $\mathcal{CR}(M, N) \neq \emptyset$. For example, models $\mathcal{G}$ and $\mathcal{H}$ over

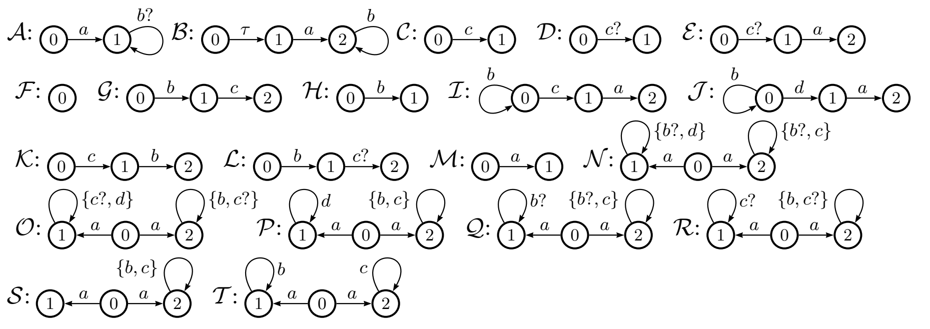

Fig. 3. Example MTSs

the vocabulary $\{\mathtt{b}, \mathtt{c}\}$ in Figure 3 are inconsistent because $\mathcal{H}$ proscribes the observable trace $\mathtt{bc}$, whereas $\mathcal{G}$ requires it.

In [27], it is argued that the merged model should not introduce unnecessary behaviours, and is therefore based on finding a *minimal common refinement*:

Definition 7. *An MTS P is a* minimal common refinement (MCR) *of MTSs M and N if $P \in \mathcal{CR}(M, N)$, $\alpha P = \alpha M \cup \alpha N$, and there is no MTS $Q \not\equiv_o P$ such that $Q \in \mathcal{CR}(M, N)$ and $Q@\alpha P \preceq_o P$.*

Let $\mathcal{MCR}(M, N)$ be the set of all MCRs of M and N. The *merge* of two consistent MTSs M and N, written $M + N$, is *one* of the models in $\mathcal{MCR}(M, N)$. Therefore, by Theorem 1, merge preserves all *true* and *false* $\mathcal{L}^w_\mu$ properties. Additionally, if there are several MCRs (see Section 3), merging involves a choice of the most appropriate one, which requires human intervention [27].

3 Existence and Uniqueness of Merge

In this section, we give practical conditions for existence and uniqueness of merge. If the latter condition is satisfied, the merge process can be fully automated.

Existence. Since merge is based on observational refinement, by Theorem 1, consistent systems over the *same* vocabulary should agree on all concrete behaviours, i.e., there should be no $\mathcal{L}^w_\mu$ property that is *true* in one system and *false* in the other (a *distinguishing* property).

Theorem 2. *If M and N are MTSs with $\alpha M = \alpha N$, then:*

$$\mathcal{CR}(M, N) \neq \emptyset \Leftrightarrow (\nexists \phi \in \mathcal{L}^w_\mu \cdot s_{0M} \in [\![\phi]\!]^\top \wedge s_{0N} \in [\![\phi]\!]^\perp)$$

Distinguishing properties can be used as a form of feedback when two systems are inconsistent. For example, the property $\langle b \rangle_o \langle c \rangle_o \mathbf{t}$ is *true* in $\mathcal{G}$ and *false* in $\mathcal{H}$, and $\mathcal{G}$ and $\mathcal{H}$ are inconsistent.

Models with *different* vocabularies must be first restricted to the shared vocabulary, but a property that distinguishes between the restricted versions of two inconsistent systems does not always exist (i.e., inconsistencies may be caused by non-shared actions). For example, $\mathcal{I}$ and $\mathcal{J}$ (see Figure 3) with $\mathtt{c} \notin \alpha\mathcal{J}$ and $\mathtt{d} \notin \alpha\mathcal{I}$ are inconsistent because $\mathcal{I}$ requires that $\mathtt{b}$'s are only proscribed after a $\mathtt{c}$, whereas $\mathcal{J}$ requires that $\mathtt{b}$'s are only proscribed after a $\mathtt{d}$. However, $\mathcal{I}@(\alpha\mathcal{I} \cap \alpha\mathcal{J}) = \mathcal{J}@(\alpha\mathcal{I} \cap \alpha\mathcal{J})$, and therefore no property distinguishes them by Theorem 2. Sufficient conditions for such properties to exist, and algorithms to check consistency and to construct distinguishing properties are given in [3]. Intuitively, the conditions require that following a non-shared action in one system (e.g., $\mathtt{c}$ in $\mathcal{I}$) does not lead to a state that is inconsistent with the other system that has not changed state (e.g., $\mathcal{I}_1$ is inconsistent with $\mathcal{J}_0$). This makes sense because the non-shared action is unobservable to the other system.

Uniqueness. When $|\mathcal{MCR}(M, N)| = 1$ (up to observational equivalence), the unique merge is called the *least common refinement* (LCR), denoted $\mathcal{LCR}_{M,N}$.

One way that multiple incomparable MCRs may exist is if there are several ways of merging behaviours that correspond to non-deterministic choices. For example, consider models $\mathcal{Q}$ and $\mathcal{R}$ in Figure 3 with vocabulary $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Both $\mathcal{Q}$ and $\mathcal{R}$ have two non-equivalent successors on $\mathtt{a}$ from the initial state, i.e., $\mathcal{Q} \xrightarrow{a}_r \mathcal{Q}_1$ and $\mathcal{Q} \xrightarrow{a}_r \mathcal{Q}_2$ such that $\mathcal{Q}_1 \not\equiv_o \mathcal{Q}_2$, and similarly for $\mathcal{R}$. However, both of $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are consistent with $\mathcal{R}_1$ and $\mathcal{R}_2$. In particular, $\mathcal{S}$ is in $\mathcal{MCR}(\mathcal{Q}, \mathcal{R})$ and corresponds to merging $\mathcal{Q}_1$ with $\mathcal{R}_1$, and $\mathcal{Q}_2$ with $\mathcal{R}_2$, whereas $\mathcal{T}$ corresponds to merging $\mathcal{Q}_1$ with $\mathcal{R}_2$, and $\mathcal{Q}_2$ with $\mathcal{R}_1$. Since $\mathcal{S} \not\equiv_o \mathcal{T}$, $\mathcal{LCR}_{\mathcal{Q},\mathcal{R}}$ does not exist. Sufficient conditions that restrict the existence of such choices, e.g., requiring that choices similar to those available with $\mathcal{Q}_1$, $\mathcal{Q}_2$,$\mathcal{R}_1$, and $\mathcal{R}_2$ lead to equivalent behaviours, have been given in [3] and are omitted here due to space limitations. The conditions are consequences of the modelling notation: non-deterministic choice could be abstracting different aspects of the system, and hence, choices could be composed in different ways, leading to multiple MCRs.

4 Algebraic Properties of Merge

In practice, merging is likely to be used in combination with refinement and parallel composition (for one such example, refer to Section 5). Therefore, it is essential to study *algebraic* properties of merge to guarantee that the overall process yields sensible results. For example, does the order in which various partial models are merged matter? Is it the same to merge two models and elaborate the result through refinement than to elaborate the models independently and then merge them? In this section, we aim to answer such questions. Specifically, we show that while the existence of multiple non-equivalent MCRs does not guarantee many of the properties that hold for the LCR case, the right choice of MCR among the possible merges can be made in order to guarantee particular algebraic properties, further emphasizing the need for human intervention in merge.

4.1 Properties of LCRs

Throughout this subsection, whenever we write $M + N$, we assume that M and N are consistent MTSs and $+$ results in $\mathcal{LCR}_{M,N}$.

Proposition 1. *For MTSs M, N, and P, the $+$ operator satisfies:*
1. *(Idempotency)* $M + M \equiv_o M$.
2. *(Commutativity)* $M + N \equiv_o N + M$.
3. *(Associativity)* $(M + N) + P \equiv_o M + (N + P)$.

A useful property of $+$ is monotonicity with respect to observational refinement: $(M \preceq_o P) \wedge (N \preceq_o Q) \Rightarrow M + N \preceq_o P + Q$. This allows for elaborating different viewpoints independently while ensuring that the properties of the original viewpoints put together still hold.

Proposition 2. (Monotonicity) *The operator $+$ is monotonic with respect to observational refinement.*

We now look at distributing parallel composition over merging. Assume that two stakeholders have developed partial models M and N of the intended behaviour of a component M. Each stakeholder will have verified that some required properties hold in a given context (other components and assumptions on the environment $P_1, \ldots, P_n$). It would be desirable if merging viewpoints M and N preserved the properties of both stakeholders under the same assumptions on the environment, i.e., in $(M + N) \parallel P_1 \parallel \cdots \parallel P_n$. This would be supported if $(M\|P_1\|\cdots\|P_n) + (N\|P_1\|\cdots\|P_n) \preceq_o (M + N)\|P_1\|\cdots\|P_n$; but unless some conditions are imposed on the model vocabularies, this property does not hold.

Example 1. Consider models $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{F}$ in Figure 3 and assume that $\alpha\mathcal{F} = \emptyset$. $\mathcal{D} + \mathcal{F}$ is always equivalent to $\mathcal{D}$, and by rule MT in Figure 1, so is $(\mathcal{D} + \mathcal{F})\|\mathcal{C}$. On the other hand, $\mathcal{D}\|\mathcal{C}$ is equal to $\mathcal{D}$ and $\mathcal{F}\|\mathcal{C}$ is equal to $\mathcal{C}$, by rules MT and TD, respectively. It follows that $(\mathcal{F}\|\mathcal{C}) + (\mathcal{D}\|\mathcal{C})$ is equivalent to $\mathcal{C}$, and hence: $(\mathcal{F}\|\mathcal{C}) + (\mathcal{D}\|\mathcal{C}) \equiv_o \mathcal{C} \not\preceq_o \mathcal{D} \equiv_o (\mathcal{F} + \mathcal{D})\|\mathcal{C}$.

The desired property fails due to the parallel composition of $\mathcal{F}$ and $\mathcal{C}$. Since c does not belong to $\alpha\mathcal{F}$, parallel composition does not restrict the occurrence of c when composing $\mathcal{F}$ with $\mathcal{C}$. However, this is methodologically wrong if we assume that $\mathcal{F}$ and $\mathcal{D}$ model the same component (which is reasonable because $\mathcal{F}$ and $\mathcal{D}$ are being merged). From $\mathcal{D}$, we know that the system modelled by $\mathcal{F}$ can communicate over c. Hence, c should be included in $\alpha\mathcal{F}$; otherwise, the communicating interface between the components modelled by $\mathcal{F}$ and $\mathcal{C}$ is under-specified. Therefore, when composing two partial models in parallel, the entire interface through which the corresponding system components communicate should be in the alphabet of their partial descriptions. We therefore require that $\alpha P \subseteq \alpha M \cap \alpha N$, where $P = P_1\|\cdots\|P_n$, for distributivity to hold.

Proposition 3. (Distributivity) *If M, N, and P are MTSs such that $\alpha P \subseteq \alpha M \cap \alpha N$, then:* $(M\|P) + (N\|P) \preceq_o (M + N)\|P$.

The other direction of Proposition 3 does not hold: $(M + N)\|P \not\preceq_o (M\|P) + (N\|P)$. This makes sense, as the composition of M with P may restrict the behaviours of M, for instance, making certain states of M unreachable. It is possible that $M\|P + N\|P$ does not refine $(M + N)\|P$ because inconsistencies are caused by those states of M that are unreachable in $M\|P$.

Example 2. Assume that models $\mathcal{D}$, $\mathcal{E}$, and $\mathcal{F}$ in Figure 3 are over the vocabulary $\{\mathsf{a}, \mathsf{c}\}$. Models $\mathcal{D}$ and $\mathcal{E}$ are consistent and their LCR is $\mathcal{F}$. So, $(\mathcal{D} + \mathcal{E})\|\mathcal{D} \equiv_o \mathcal{F}$ by the rules in Figure 1. On the other hand, $\mathcal{D}\|\mathcal{D} = \mathcal{D}$ and $\mathcal{E}\|\mathcal{D} = \mathcal{D}$, and therefore by Idempotency, $\mathcal{D}\|\mathcal{D} + \mathcal{E}\|\mathcal{D} \equiv_o \mathcal{D}$. Since $\mathcal{F} \not\preceq_o \mathcal{D}$, the result follows.

In Example 2, $\mathcal{D}$ and $\mathcal{E}$ have a disagreement on a after following the maybe transitions on c, which results in the merge $\mathcal{F}$. The source of this disagreement is removed upon composing both $\mathcal{D}$ and $\mathcal{E}$ with $\mathcal{D}$, because $\mathcal{D}$ restricts the behaviour of $\mathcal{E}$ on a. The merge of $\mathcal{D}\|\mathcal{D}$ and $\mathcal{E}\|\mathcal{D}$ therefore allows more behaviours, and does not refine $(\mathcal{D} + \mathcal{E})\|\mathcal{D}$.

4.2 Properties of MCRs

In this subsection, we present algebraic properties of $+$ without assuming the existence of the LCR. The algebraic properties are therefore stated in terms of sets and the different choices that can be made when picking an MCR. Idempotence is the only property in Section 4.1 that still holds, since an LCR always exists between a system and itself. The rest of the properties discussed in Section 4.1 require some form of weakening.

Commutativity does not hold in general: if M and N are any two MTSs that have at least two different MCRs, then certainly not every $M+N$ is equivalent to every $N+M$. On the other hand, $\mathcal{MCR}(M,N)$ is always equal to $\mathcal{MCR}(N,M)$, and therefore the same MCR can be chosen.

Proposition 4. (Commutativity) *$\mathcal{MCR}(M,N) = \mathcal{MCR}(N,M)$.*

Associativity fails for the same reason that commutativity fails. The strongest form of associativity in terms of sets is:

$$\forall A \in \mathcal{MCR}(M,N) \cdot \forall B \in \mathcal{MCR}(N,P) \cdot \mathcal{MCR}(A,P) = \mathcal{MCR}(M,B)$$

The following example shows that this form does not hold in general.

Example 3. Consider models $\mathcal{C}$, $\mathcal{H}$, and $\mathcal{M}$ in Figure 3 and assume that $\alpha\mathcal{C} = \{\mathtt{c}\}$, $\alpha\mathcal{H} = \{\mathtt{b}\}$, and $\alpha\mathcal{M} = \{\mathtt{a}\}$. Model $\mathcal{K}$ is in $\mathcal{MCR}(\mathcal{C},\mathcal{H})$, and there is no $D \in \mathcal{MCR}(\mathcal{M},\mathcal{C})$ such that $\mathcal{MCR}(\mathcal{K},\mathcal{M}) = \mathcal{MCR}(\mathcal{H},D)$.

In Example 3, $\mathcal{K}$ requires that action $\mathtt{c}$ precedes action $\mathtt{b}$ in every trace, and therefore so does every MCR of $\mathcal{K}$ and $\mathcal{M}$, since neither $\mathtt{b}$ nor $\mathtt{c}$ is in $\alpha\mathcal{M}$. However, because $\mathtt{b}$ is not in $\alpha\mathcal{M}$ or $\alpha\mathcal{C}$, for every D in $\mathcal{MCR}(\mathcal{M},\mathcal{C})$, there is an MCR of $\mathcal{H}$ and D such that action $\mathtt{c}$ follows action $\mathtt{b}$. Hence, $\mathcal{MCR}(\mathcal{K},\mathcal{M}) \neq \mathcal{MCR}(\mathcal{H},D)$ for *any* D in $\mathcal{MCR}(\mathcal{M},\mathcal{C})$. In fact, Example 3 shows that there exists M, N, and P such that: $\exists A \in \mathcal{MCR}(M,N) \cdot \forall B \in \mathcal{MCR}(N,P) \cdot \mathcal{MCR}(A,P) \neq \mathcal{MCR}(M,B)$. Therefore, set equality of $\mathcal{MCR}(A,P)$ and $\mathcal{MCR}(M,B)$ for associativity is not possible. Additionally, it can be shown that fixing both A in $\mathcal{MCR}(M,N)$ and B in $\mathcal{MCR}(N,P)$ is unreasonable, as it may force incompatible decisions to be made when merging A with P and M with B [3]. Instead, the following proposition outlines two forms of associativity, without set equality, that fix some A in $\mathcal{MCR}(M,N)$ or some B in $\mathcal{MCR}(N,P)$, but not both.

Proposition 5. (Associativity) *If M, N, and P are MTSs, then:*

1. *$\forall A \in \mathcal{MCR}(M,N) \cdot \exists B \in \mathcal{MCR}(N,P) \cdot (\mathcal{MCR}(A,P) \cap \mathcal{MCR}(M,B) \neq \emptyset)$,*
2. *$\forall B \in \mathcal{MCR}(N,P) \cdot \exists A \in \mathcal{MCR}(M,N) \cdot (\mathcal{MCR}(A,P) \cap \mathcal{MCR}(M,B) \neq \emptyset)$,*

where $\mathcal{CR}(A,P) \neq \emptyset$ and $\mathcal{CR}(M,B) \neq \emptyset$.

Condition (1) in Proposition 5 says that for any $M+N$, there exists some $N+P$ such that the same MCR for $(M+N)+P$ and $M+(N+P)$ can be selected. Condition (2) is analogous to condition (1) with the roles of $M+N$ and $N+P$ reversed. Note that Proposition 5 reduces to Proposition 1 if all sets of MCRs are singletons (up to equivalence).

Monotonicity is also disrupted by multiple MCRs. It is not expected that any choice of $M + N$ is refined by any choice of $P + N$ when M is refined by P, because incompatible decisions may be made in the two merges. Rather, there are two desirable forms of monotonicity: (1) whenever $M + N$ is chosen, some $P + N$ can be chosen such that $P + N$ refines $M + N$; and (2) whenever $P + N$ is chosen, then some $M + N$ can be chosen such that $P + N$ refines $M + N$. Form (1) does not hold, as the following example shows.

Example 4. Models $\mathcal{D}$ and $\mathcal{H}$ in Figure 3 with $\alpha\mathcal{D} = \{\mathtt{c}\}$ and $\alpha\mathcal{H} = \{\mathtt{b}\}$ are consistent, and their merge $\mathcal{D} + \mathcal{H}$ may result in model $\mathcal{K}$. Also, $\mathcal{D} \preceq_o \mathcal{F}$ (assuming that $\alpha\mathcal{F} = \{\mathtt{c}\}$) and models $\mathcal{F}$ and $\mathcal{H}$ are consistent. However, $\mathcal{LCR}_{\mathcal{F},\mathcal{H}}$ is equivalent to $\mathcal{H}$ over $\{\mathtt{b}, \mathtt{c}\}$, and since $\mathcal{H} \not\preceq_o \mathcal{K}$, no MCR of $\mathcal{F}$ and $\mathcal{H}$ that refines $\mathcal{K}$ can be chosen.

Form (1) fails because there are two choices of refinement being made. On the one hand, by picking a minimal common refinement for M and N over others, we are deciding over incompatible refinement choices. On the other hand, we are choosing how to refine M into P. These two choices need not be consistent, leading to failure of monotonicity. This tells us that choosing an MCR adds information to the merged model, which may be inconsistent with evolutions of the different viewpoints that are represented by the models being merged. Form (2) always holds, as stated below.

Proposition 6. (Monotonicity) *If M, N, P, and Q are MTSs, then:*

$$M \preceq_o P \land N \preceq_o Q \Rightarrow \forall B \in \mathcal{MCR}(P, Q) \cdot \exists A \in \mathcal{MCR}(M, N) \cdot A \preceq_o B$$

Thus, once $P + Q$ is chosen, there always exists some $M + N$ that it refines, and so the properties of M and N are preserved in $P + Q$. Note that if $\mathcal{MCR}(M, N)$ is a singleton set, Proposition 6 reduces to Proposition 2, as expected. In practical terms, this means that if the various viewpoints are still to be elaborated, the results of reasoning about one of their possible merges (picked arbitrarily) are not guaranteed to carry through once the viewpoints have been further refined.

We now address distributivity in the context of multiple MCRs. Similar to monotonicity, there are two desirable forms of this property: (1) given any A in $\mathcal{MCR}(M\|P, N\|P)$, there is some B in $\mathcal{MCR}(M, N)$ such that A is refined by $B\|P$; and (2) given any B in $\mathcal{MCR}(M, N)$, there is some A in $\mathcal{MCR}(M\|P, N\|P)$ such that A is refined by $B\|P$. Form (1) does not hold, as the following example shows.

Example 5. Consider models $\mathcal{F}$, $\mathcal{N}$, $\mathcal{O}$, $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$ in Figure 3 and assume that $\alpha\mathcal{F} = \{\mathtt{d}\}$, and the rest have vocabulary $\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$. By the rules in Figure 1, $\mathcal{N}\|\mathcal{F} = \mathcal{Q}$ and $\mathcal{O}\|\mathcal{F} = \mathcal{R}$, and furthermore, $\mathcal{T}$ is in $\mathcal{MCR}(\mathcal{Q}, \mathcal{R})$. On the other hand, $\mathcal{LCR}_{\mathcal{N},\mathcal{O}}$ is $\mathcal{P}$, and $\mathcal{P}\|\mathcal{F} = \mathcal{S}$, which is not a refinement of $\mathcal{T}$.

In the previous example, $\mathcal{LCR}_{\mathcal{N},\mathcal{O}}$ exists because the required transitions on $\mathtt{d}$ in these models restrict the choices that can be made with respect to combining the non-determinism on action $\mathtt{a}$: ($\mathcal{N}_1$ and $\mathcal{O}_1$) and ($\mathcal{N}_2$ and $\mathcal{O}_2$) are consistent, but neither ($\mathcal{N}_1$ and $\mathcal{O}_2$) nor ($\mathcal{N}_2$ and $\mathcal{O}_1$) are consistent. Upon composing with

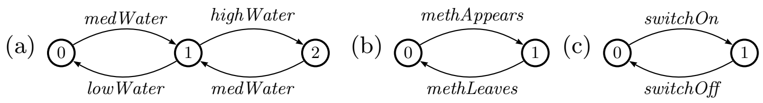

Fig. 4. The MTSs for (a) *WaterLevelSensor*, (b) *MethaneSensor*, and (c) *Pump*

$\mathcal{F}$, the source of the inconsistencies between ($\mathcal{N}_1$ and $\mathcal{O}_2$) and ($\mathcal{N}_2$ and $\mathcal{O}_1$) is removed, and consequently, $\mathcal{LCR}_{\mathcal{N}\|\mathcal{F},\mathcal{O}\|\mathcal{F}}$ does not exist. In particular, similar to Example 2 in Section 4.1, parallel composition may remove inconsistencies between M and N, allowing for more common refinements of $M\|P$ and $N\|P$.

On the other hand, Form (2) holds, and is of particular utility when elaborating models from different viewpoints (see Section 5).

Proposition 7. (Distributivity) *If M, N, and P are such that $\alpha P \subseteq \alpha M \cap \alpha N$, then: $\forall B \in \mathcal{MCR}(M,N) \cdot \exists A \in \mathcal{MCR}(M\|P, N\|P) \cdot A \preceq_o B\|P$.*

In this subsection, we showed that when + does not necessarily produce an LCR, most properties studied in Section 4.1 fail to hold. Intuitively, the existence of inequivalent MCRs implies that merging involves a choice that requires some form of human intervention: a choice which is loaded with domain knowledge. This impacts the results on algebraic properties when moving from LCRs to MCRs. However, we have shown that the right choices of MCRs can be made in order to guarantee particular algebraic properties.

5 A Case Study: The Mine Pump

In this section, we show how our results support elaboration of partial models for a mine pump case study [18].

Overview. A pump controller is used to prevent the water in a mine sump from passing some threshold, and hence flooding the mine. To avoid the risk of explosion, the pump may only be active when there is no methane gas present in the mine. The pump controller monitors the water and methane levels by communicating with two sensors. In addition, the pump is equipped with a danger light that is intended to reflect the presence of methane in the sump.

We model the mine pump with four components: *WaterLevelSensor*, *MethaneSensor*, *PumpControl*, and *Pump*. The complete system, *MinePump*, is the parallel composition of these components, namely (*PumpControl* $\|$ *Pump* $\|$ *MethaneSensor* $\|$ *WaterLevelSensor*).

WaterLevelSensor models the water sensor and includes assumptions on how the water level is expected to change between low, medium, and high. *MethaneSensor* keeps track of whether methane is present in the mine, and *Pump* models the physical pump that can be switched on and off. For simplicity, we assume to have complete knowledge for these descriptions and hence model them with LTSs depicted in Figure 4, where initially the water is low, the pump is off, and no methane is present.

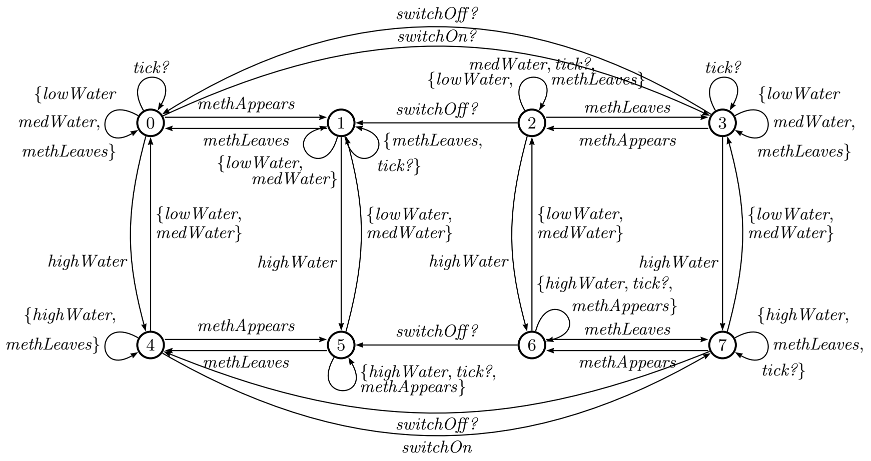

Fig. 5. The MTS for *OnPolicy*

PumpControl describes the controller that monitors water and methane levels, controls the pump in order to guarantee the safety properties of the pump system, and also maintains the status of the danger light according to the methane level. This informal description leaves open the exact water level at which to turn the pump on and off. For example, the pump could be turned on when there is high water or possibly when the water is not low, (e.g., at a medium level). The pump could be turned off when there is low water or possibly when the water is not high.

Partial Behaviour Models. We assume that there are two stakeholders for the pump controller: one with the knowledge of when the pump should be on (referred to as the on policy) and another with knowledge of when the pump should be off (referred to as the off policy).

The MTS models used to describe the policies for the pump controller use an event labelled *tick*, which models the passage of time units as kept by a global clock. All components whose behaviour is timed synchronize to this event. This corresponds to a standard approach to modelling discrete time in event-based formalisms [22]. Modelling time is required for systems such as the mine pump, where urgency of certain events, such as switching the pump off to avoid an explosion, must be captured.

Consider the *OnPolicy* model (Figure 5) provided by one of the stakeholders. This model attempts to describe how the pump controller will ensure that the pump is on in order to satisfy the safety requirements for the mine. To do so, the controller keeps track of the water level and methane presence information provided by the various sensors. States 4 to 7 and 0 to 3 are those in which the water level is high and not high, respectively, while states {1, 2, 5, 6} and {0, 3, 4, 7} are those in which methane is present and not present, respectively.

The on policy requires the pump controller to switch the pump on when there is high water and no methane present (see transition labelled *switchOn* from state 4 to state 7), and leaves the possibility open for turning the pump on when there is medium water or low water (see *switchOn?* transition from state 0 to state 2). Event *tick* is not allowed to occur in state 4, which captures the fact that the pump controller is required to react fast enough to switch the pump on before the time unit expires (or that there is no longer an urgency because the water is no longer high – leading to state 0 – or there is methane – leading to state 5). In addition, all other *tick* events are maybe transitions modelling the fact that time may pass at any rate on all other states, and hence indicating that there are no other timing requirements for the pump controller.

The *OffPolicy* turns the pump off when there is low water or methane appears. In addition, *OffPolicy* models a danger light with actions *dangerLightOn* and *dangerLightOff*, which is turned on when methane is present in the mine. The actions that refer to the danger light are not in the scope of *OnPolicy*; in other words, they are not in the alphabet of *OnPolicy*. Due to the size of *OffPolicy* (16 states, 112 transitions), we do not depict it here; it is available in [3].

Properties. We consider four properties that stakeholders expect their models to satisfy, but due to lack of space omit their formalization in $\mathcal{L}^w_\mu$. The first two properties (Φ_1 and Φ_2) are expected to be satisfied by both policies. Φ_1 states that the pump should only be turned on if it is off and similarly, Φ_2 states that the pump should only be turned off if it is on. These two properties necessitate including both *switchOn* and *switchOff* in the scope of *OnPolicy* and *OffPolicy*. In addition, the stakeholder for the on policy expects that when there is high water and no methane, the pump should be on (Φ_3), while the stakeholder for the off policy expects that if there is low water or methane present, the pump should be off (Φ_4).

It is possible to show that both $MinePump_1$ and $MinePump_2$ satisfy properties Φ_1 and Φ_2 and additionally, $MinePump_1$ satisfies Φ_3 and $MinePump_2$ satisfies Φ_4. As *OnPolicy* leaves the off policy open by modelling possibilities for turning the pump off with maybe behaviour, and similarly, *OffPolicy* leaves the on policy open, properties Φ_3 and Φ_4 evaluate to *maybe* in $MinePump_2$ and $MinePump_1$, respectively.

Merge. Given the results presented above, we claim that the system model resulting from the merged policies and models for the environment satisfy all requirements, i.e. (*OnPolicy* + *OffPolicy*) $\|$ *WaterLevelSensor* $\|$ *MethaneSensor* $\|$ *Pump* satisfies Φ_1, Φ_2, Φ_3, and Φ_4. The argument is as follows.

Using the consistency algorithm in [3], *OnPolicy* and *OffPolicy* can be shown to be consistent, and hence the full pump controller can be defined as *PumpControl* = *OnPolicy* + *OffPolicy*. Additionally, the alphabet restrictions in Proposition 7 are satisfied by $MinePump_1$ and $MinePump_2$, and therefore there exists a merge of $MinePump_1$ and $MinePump_2$ that is refined by *MinePump*. Hence, by definition of merge and Theorem 1, properties Φ_1, Φ_2, Φ_3, and Φ_4 hold in *MinePump*. In particular, the *maybe* properties Φ_3 and Φ_4 of $MinePump_1$ and $MinePump_2$ become *true* properties of *MinePump*, which corresponds to the on and off policies being refined into concrete behaviours in the merge.

The above reasoning did not rely on the fact that the LCR of *OnPolicy* and *OffPolicy* exists. Existence of the LCR does guarantee that the merge of the policies can be built automatically using the algorithms reported in [3].

In addition, by Proposition 7, the properties of the compositions $MinePump_1$ and $MinePump_2$ are preserved in addition to those of *OnPolicy* and *OffPolicy*. This is important as some relevant properties may only hold under certain assumptions on the environment. In this case study, *OnPolicy* could easily be modified to be under-specified enough so as not to satisfy Φ_1 without the assumption (modelled in *WaterLevelSensor*) that water levels cannot jump directly from low to high without going through medium.

Now suppose that further information regarding the on and off policies becomes available, e.g., that the pump controller should try to keep the pump off as much as possible while satisfying existing requirements. This means that the pump should only be switched on when the water is high and should be switched off as soon as the water is no longer high. By monotonicity (Proposition 6), we know that rather than being forced to elaborate the merged model *MinePump* or *PumpControl*, we can effectively refine the two original policies into $OnPolicy'$ and $OffPolicy'$ and then merge them with the guarantee that in the final model $((OnPolicy' + OffPolicy') \parallel Pump \parallel MethaneSensor \parallel WaterLevelSensor)$ all required properties still hold.

Conclusion. In this section, we have exemplified several of the properties of merge discussed in this paper. Specifically, we showed that we can start from partial models from different stakeholders, each satisfying certain system requirements (possibly under some assumptions on the environment), and elaborate through refinement, merge and parallel composition a system model that preserves the properties of the initial viewpoints.

6 Conclusions

In this section, we summarize the paper, compare our work with related approaches, and discuss directions for future research.

Summary. Merging is a process based on finding a common observational refinement of consistent systems [27], and therefore preserves weak μ-calculus properties of the original systems. In this paper, we studied fundamental questions related to using merge in practice. In particular, we showed that existence of merge is characterized by weak μ-calculus properties and can be decided algorithmically, and described conditions for uniqueness, which are essential for automating merge. Together with several algebraic properties (both in the case when the LCR exists and when merge results in one of several MCRs), our results provide the necessary support for merging complex systems, such as those involving the parallel composition of several components, as demonstrated in the case study.

Related Work. Explicit partiality corresponds naturally to the lack of information at modelling time [9] or to the loss of information due to abstraction [4,8,14,25].

State-machine formalisms have been extended to allow partiality in states (e.g., Partial Kripke Structures (PKs) [4]), transitions (e.g., MTSs [21], Mixed Transition Systems [8]), or both (e.g., Generalized KMTSs [25]). In all these formalisms, properties that are preserved in a more defined model have been identified, e.g. Hennessy-Milner logic for MTSs [21], 3-valued CTL for PKs, and 3-valued μ-calculus for KMTSs [14].

The approaches closest to ours are those of Larsen et al. [19,20] and Huth et al. [13,14]. [20] introduced an operator with a behaviour similar to our merge (called *conjunction*), but defined only for MTSs over the same vocabulary with no τ transitions, and for which there is an *independence relation* (at which point the LCR exists). Although not studied in depth, the operators in [20,19] are based on strong refinement. We have shown that the existence of multiple MCRs introduces a number of subtle issues for a similar operator based on observational refinement. Extensions to MTSs have been proposed to guarantee uniqueness of merge and generalization (e.g. [19,14]). The price paid is more complicated modelling frameworks which engineers may not adopt as easily, and hence we focus on MTSs. In addition, non-uniqueness of merge can be seen as an opportunity for elicitation, validation, and negotiation of partial descriptions. Finally, Hussain and Huth [13] study the problem of finding a common (strong) refinement between multiple MTS, but focus on the complexity of the relevant model-checking problems rather than engineering issues (e.g., existence, uniqueness and algebraic properties). Our models are more general in that we allow τ transitions and different alphabets, but less general in that the work in [13] handles hybrid constraints between the models.

Our work focuses on merging models that describe only the *observable* behaviour of a system, and thus simulation-like relations are central to merging. Other approaches to merging descriptions exist, but the models being merged include state information [9,24,5,29,11,1] and consequently other notions of preservation may apply, such as isomorphism [9,24].

An alternative to partial operational descriptions, on which we focus, is the use of declarative specifications. For instance, classical logics are partial and support merging as the conjunction of theories. Similarly, Live Sequence Charts [7] support merging through logical conjunction, as each chart can be interpreted as a temporal logic formula. We believe that our approach is more suitable for exploration and validation of unknown behaviours, since explicit reasoning about such behaviours is an integral part of our merging process.

Future Work. The long-term goal of our work is to provide automated support for creating, merging and elaborating partial behavioural models, as well as enabling users to choose the desired merge from the set of possible minimal common refinements. In the near future, we plan to conduct additional case studies, and produce implementations of the merge algorithms found in [3]. In addition, since weak μ-calculus is expressive but can be subtle to use, we plan to extend the logic Fluent LTL (FLTL) [10], which is a simple language for expressing complex temporal properties of LTSs, to reasoning about partial models and use it as the specification language in our framework.

Acknowledgements. The work was partially funded by EPSRC grant PBM EP/C541138 and NSERC. We thank Shiva Nejati for finding interesting counterexamples to our claims.

References

1. T. Ball, V. Levin, and F. Xie. "Automatic Creation of Environment Models via Training". In *TACAS'04*, volume 2988 of *LNCS*, pages 93–107, 2004.
2. B. Boem and R. Turner. *Balancing Agility and Discipline: A Guide for the Perplexed.* Person Education, 2004.
3. G. Brunet. "A Characterization of Merging Partial Behavioural Models". Master's thesis, University of Toronto, Department of Computer Science, January 2006.
4. G. Bruns and P. Godefroid. "Model Checking Partial State Spaces with 3-Valued Temporal Logics". In *CAV'99*, volume 1633 of *LNCS*, pages 274–287, 1999.
5. M. Chechik, B. Devereux, S. Easterbrook, and A. Gurfinkel. "Multi-Valued Symbolic Model-Checking". *ACM TOSEM*, 12(4):1–38, October 2003.
6. CREWS. "Cooperative Requirements Engineering With Scenarios", 1999.
7. W. Damm and D. Harel. "LSCs: Breathing Life into Message Sequence Charts.". *FMSD*, 19(1):45–80, 2001.
8. D. Dams, R. Gerth, and O. Grumberg. "Abstract Interpretation of Reactive Systems". *ACM TOPLAS*, 2(19):253–291, 1997.
9. S. Easterbrook and M. Chechik. "A Framework for Multi-Valued Reasoning over Inconsistent Viewpoints". In *ICSE'01*, pages 411–420, 2001.
10. D. Giannakopoulou and J. Magee. "Fluent Model Checking for Event-Based Systems". In *ESEC/FSE'03*, pages 257–266, 2003.
11. S. Horwitz, J. Prins, and T. Reps. "Integrating Noninterfering Versions of Programs.". *ACM TOPLAS*, 11(3):345–387, 1989.
12. A. Hunter and B. Nuseibeh. "Managing Inconsistent Specifications: Reasoning, Analysis and Action". *ACM TOSEM*, 7(4):335–367, 1998.
13. A. Hussain and M. Huth. "On Model Checking Multiple Hybrid Views". In *1st Int. Symp. on Leveraging Applications of FMs*, pages 235–242, 2004.
14. M. Huth, R. Jagadeesan, and D. Schmidt. "A Domain Equation for Refinement of Partial Systems". Submitted, 2002.
15. M. Huth, R. Jagadeesan, and D. A. Schmidt. "Modal Transition Systems: A Foundation for Three-Valued Program Analysis". In *ESOP'01*, volume 2028 of *LNCS*, pages 155–169, 2001.
16. R. Keller. "Formal Verification of Parallel Programs". *Communications of the ACM*, 19(7):371–384, 1976.
17. D Kozen. "Results on the Propositional μ-calculus". *TCS*, 27:334–354, 1983.
18. J. Kramer, J. Magee, and M. Sloman. "CONIC: an Integrated Approach to Distributed Computer Control Systems". *IEE Proceedings*, 130(1):1–10, 1983.
19. K. Larsen and L. Xinxin. "Equation Solving Using Modal Transition Systems". In *LICS'90*, pages 108–117, 1990.
20. K. G. Larsen, B. Steffen, and C. Weise. "A Constraint Oriented Proof Methodology based on Modal Transition Systems". In *TACAS'95*, LNCS, pages 13–28, 1995.
21. K.G. Larsen and B. Thomsen. "A Modal Process Logic". In *LICS'88*, pages 203–210, 1988.
22. J. Magee and J. Kramer. *"Concurrency - State Models and Java Programs".* John Wiley, 1999.

23. R. Milner. *Communication and Concurrency*. Prentice-Hall, New York, 1989.
24. M. Sabetzadeh and S.M. Easterbrook. “Analysis of Inconsistency in Graph-Based Viewpoints: A Category-Theoretic Approach”. In *ASE’03*, pages 12–21, 2003.
25. S. Shoham and O. Grumberg. “Monotonic Abstraction-Refinement for CTL”. In *TACAS’04*, volume 2988 of *LNCS*, pages 546–560, 2004.
26. C. Stirling. “Modal and Temporal Logics for Processes”. In *VIII Banff Conf. on Logics for Concurrency : Structure Versus Automata*, pages 149–237, 1996.
27. S. Uchitel and M. Chechik. “Merging Partial Behavioural Models”. In *FSE’04*, pages 43–52, 2004.
28. S. Uchitel, J. Kramer, and J. Magee. “Behaviour Model Elaboration using Partial Labelled Transition Systems”. In *ESEC/FSE’03*, pages 19–27, 2003.
29. J. Whittle and J. Schumann. “Generating Statechart Designs from Scenarios”. In *ICSE’00*, pages 314–323, 2000.