Tasks Decomposition of System Models for Human-Machine Interaction Analysis

Guillaume Maudoux
Université catholique de Louvain, Belgium
guillaume.maudoux@uclouvain.be

Sébastien Combéfis
École Centrale des Arts et Métiers, Belgium
s.combefis@ecam.be

Charles Pecheur
Université catholique de Louvain, Belgium
charles.pecheur@uclouvain.be

ABSTRACT

This paper is concerned with the problem of learning how to interact safely with complex automated systems. With large systems, human-machine interaction errors like automation surprises are more likely to happen. Previous works have introduced the notion of full-control mental models for operators. These are formal system abstractions embedding the required information to control a system completely and without surprises. Full-control mental models can be used as training material but are ineffective as their control over a system is only guaranteed when fully learned.

This work investigates the problem of decomposing full-control mental models into smaller independent tasks. These tasks each allow to control a subset of the system and can be learned incrementally to control more and more features of the system. This paper proposes an operator that describes how two mental models are merged when learned sequentially. With that operator, we show how to generate a set of small tasks with the required properties.

ACM Classification Keywords

B.8.2. Performance and Reliability: Performance Analysis and Design Aids; D.2.4. Software/Program Verification: Formal methods; H.1.2. Models and Principles: User/Machine Systems

Author Keywords

Task decomposition; LTS; Task analysis.

INTRODUCTION

The field of human-computer interaction analysis formalises how human operators interact with automated systems and studies how to assert and improve the quality of these interactions. An important problem is to ensure that humans can interact with a system without surprises and provide a description of such interactions.

Surprises are defined as mismatches between expectations of the operator and the actual behaviour of a system. An operator maintains a model of the system called a mental model[7].

To cite this article use the following information:

Guillaume Maudoux, Sébastien Combéfis, and Charles Pecheur. 2015. Tasks Decomposition of System Models for Human-Machine Interaction Analysis. In *Proceedings of the Workshop on Formal Methods in Human Computer Interaction* (FoMHCI'15), 7-12.
http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:hbz:82-rwth-2015-030425

He builds it by experimenting on and learning about the system. Operators are assumed to behave according to their mental model. Therefore, mental models should always allow to control the system in use. In that context, learning new features of the system should be done in such a way that the new mental model of the users also allow to control the system. In particular, operators must not interact with the system until the end of a learning phase and cannot fill their functions during that time.

Operators that have learnt all the possible behaviours of a system have built a full-control mental model. Such models have been defined in [4] and techniques to build minimal ones have been described in [3] and [2]. These models allow to control safely all the features of a system. However, learning full-control mental-models is impractical as it implies to learn all the features of a system in one big step. This means that newly hired operators are useless before they master the full complexity of the system. Large systems might even be too complex for one operator to manage. In that case, the system must be split in tasks dedicated to different operators.

To be practical, learning processes should provide a set of tasks in the form of small compatible mental models that can be combined incrementally into bigger models. Each task and each intermediate mental model should ensure safe interactions with the system without necessarily describing all of its features.

Our work investigates how to decompose a full-control mental model into smaller tasks that individually control the system. Learned sequentially, these tasks should augment the mental model of the operator until he possesses a full-control mental model of the system.

Tasks have long been used in the context of human-machine interactions. They can be defined during the system design process and used for system validation like in [5]. They can also be synthesised through a goal that a user needs to achieve and relate to controler synthesis as explained in the Ramadge-Wonham framework [6]. In this work, we propose tasks that have the property of being small and combinable, with no guarantee that they correspond to meaningful objectives for users.

In this paper, we introduce an operator to combine mental models and we argue that it is coherent with the intuition of learning a task. We describe the related decomposition operation and show that mental models can be decomposed into a finite set of basic mental models. We also show that it is pos-

sible to build a set of tasks that properly control the system and such that all their combinations also control the system and eventually have full-control over it. Finally, we define the task complexity as a measure of the difficulty to learn a system.

The remaining of this article is organized as follows. First we introduce the required definitions of mental models and controllability in the"background" section. In the "HMI-LTS decomposition" section, we introduce the composition operator and the decomposition properties. Finally, we show how to obtain the desired decomposition in the section "Full-control mental model decomposition".

BACKGROUND

In this section we define HMI-LTSs, mental models and the full control property. We also introduce full-control mental models, a concept that lies at the intersection of these three notions. This section is intended as a reminder of the required concepts defined in [1].

We start with the concept of labelled transition systems for human-machine interactions (HMI-LTSs) which are slightly modified labelled transition systems (LTSs). An LTS is a state transition system where each transition has an action label. LTSs interact with their environment based on this set of actions. Additionally, LTSs can have an internal τ action that cannot be observed by the environment. Two small LTSs are shown in figure 2.

DEFINITION 1 (LABELLED TRANSITION SYSTEM). *A labelled transition system (LTS) is a tuple $\langle S, \mathcal{L}, s_0, \rightarrow\rangle$ where S is a finite set of states, $\mathcal{L}$ is a finite set of labels representing visible actions, $s_0 \in S$ is the initial state and $\rightarrow \subseteq S \times (\mathcal{L} \cup \{\tau\}) \times S$ is the transition relation, where $\tau \notin \mathcal{L}$ is the label for the internal action.*

The executions of LTSs can be observed from the environment via traces. An *execution* of an LTS is a sequence of transitions $s_0 \xrightarrow{a_1} s_1 \dots s_{n-1} \xrightarrow{a_n} s_n$ where each $(s_{i-1}, a_i, s_i) \in \rightarrow$. A *trace* of an LTS is a sequence $\sigma = a_1, a_2, \dots a_n$ where each $a_i \in \mathcal{L}$ and such that there exists an execution $s_0 \xrightarrow{\tau* a_1 \tau*} s_1 \dots s_{n-1} \xrightarrow{\tau* a_n \tau*} s_n$, where $s_0 \xrightarrow{\tau* a_1 \tau*} s_1$ is itself an execution whose only observable action is a_1. For example the Lamp system of figure 1 can exhibit the trace "on, off, on, off, burn" and the trace "smash, replace, on" among infinitely many other.

HMI-LTSs refine LTS by distinguishing two kinds of actions, *commands* and *observations*. Like any I/O transition system, observations are uncontrollable outputs generated by the system and commands are controllable inputs. HMI-LTSs are exactly equivalent to Tretmans' LTS/IOs[8].

DEFINITION 2 (HUMAN-MACHINE INTERACTION LTS). *A human-machine interaction labelled transition system (HMI-LTS) is a tuple $\langle S, \mathcal{L}^c, \mathcal{L}^o, s_0, \rightarrow\rangle$ where $\langle S, \mathcal{L}^c \cup \mathcal{L}^o, s_0, \rightarrow\rangle$ is a labelled transition system, $\mathcal{L}^c$ is a finite set of command labels and $\mathcal{L}^o$ is a finite set of observation labels. The two sets $\mathcal{L}^c$ and $\mathcal{L}^o$ are disjoint and the set of visible actions is $\mathcal{L} = \mathcal{L}^c \cup \mathcal{L}^o$.*

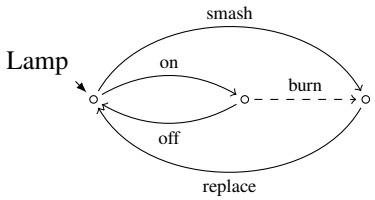

Figure 1. An HMI-LTSs model of a lamp with four commands and one observation. A lamp can be switched on and off as long as it does not burn. When burned or smashed, the lamp needs to be replaced and we are back to the starting point. This HMI-LTS is our its simplicity, it is also its own and only minimal full-contol mental model.

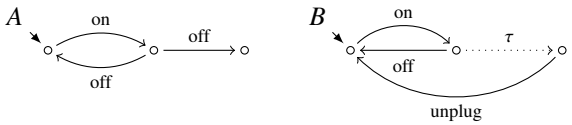

Figure 2. Two examples of nondeterministic systems. A can be turned on then off at least once, but it is impossible to say if it can be turned on again. B can be turned on and off, but it can also unobsevably change to a state where the only way to restart it is to unplug it.

HMI-LTSs are used to describe both systems and mental models. Mental models are user views of a system. They are by definition deterministic and represent the knowledge an operator has about the system he controls. It is important to note that mental models do not represent the behaviour of a user, but the behaviour of a system as seen by a user. A command in a mental model corresponds exactly to the same command on the system. The interactions between a system $\mathcal{S}$ and an operator behaving according to its mental model $\mathcal{M}$ are defined by the synchronous parallel composition $\mathcal{S} \parallel \mathcal{M}$. This distinguishes HMI-LTSs from LTS/IOs where inputs of the system must be synchronised on the outputs of the user and vice versa.

In addition, we want mental models to control systems without surprises. In particular, we want to avoid mental models to contain commands that are impossible on the system and to ignore observations that the system could produce. This motivates the introduction of the control property.

The following definition uses the s **after** σ operator that describes the set of states that can be reached from the state s after an execution whose observable trace is σ. Also, $A^c(s)$ (resp. $A^o(s)$) is the set of possible commands (resp. observations) of s. An action is possible in s if it is the first action of some trace starting at s. Moreover, an LTS is deterministic if $|s \textbf{ after } \sigma| \leq 1$ for any σ. The HMI-LTS A from figure 2 can be in two states after the trace "on, off" and is therefore not deterministic. The HMI-LTS B has two possible actions in its middle state: 'off' and 'unplug'

DEFINITION 3 (CONTROL PROPERTY). *Given two HMI-LTSs $\mathcal{S} = \langle S_{\mathcal{S}}, \mathcal{L}^c, \mathcal{L}^o, s_{0\mathcal{S}}, \rightarrow_{\mathcal{S}}\rangle$ and $\mathcal{M} = \langle S_{\mathcal{M}}, \mathcal{L}^c, \mathcal{L}^o, s_{0\mathcal{M}}, \rightarrow_{\mathcal{M}}\rangle$, $\mathcal{M}$ controls $\mathcal{S}$ if $\mathcal{M}$ is deterministic and for all traces $\sigma \in \mathcal{L}^*$ such that $s_{\mathcal{S}} \in s_{0\mathcal{S}}$ **after** σ and*

$\{s_{\mathcal{M}}\} = s_{0\mathcal{M}}$ **after** σ :

$$A^c(s_{\mathcal{S}}) \supseteq A^c(s_{\mathcal{M}}) \text{ and } A^o(s_{\mathcal{S}}) \subseteq A^o(s_{\mathcal{M}}).$$

This definition is symmetric because it allows the mental model not to know the full set of available commands while allowing the system to produce less observations than expected by the mental model. From now on, this is the formal definition we refer to when we say that a mental model controls a system.

For a given system, there always exists a mental model that contains no commands and still allows to control the system. That mental model contains only the traces of observations available from the initial state and corresponds to the mental model needed by an agent to avoid surprises when not interacting with a system. For example, you need to know that your desk phone may ring even when you do not want to interact with it. Someone who ignores that fact will be surprised whenever the phone rings.

We see that a mental model that controls a system does not necessarily explore the full range of possible behaviours of that system. When a mental-model ensures control over a system and allows to access all the available commands of the system, we say that the model fully controls the system.

DEFINITION 4 (FULL-CONTROL PROPERTY). *Given two HMI-LTSs* $\mathcal{S} = \langle S_{\mathcal{S}}, \mathcal{L}^c, \mathcal{L}^o, s_{0\mathcal{S}}, \rightarrow_{\mathcal{S}} \rangle$ *and* $\mathcal{M} = \langle S_{\mathcal{M}}, \mathcal{L}^c, \mathcal{L}^o, s_{0\mathcal{M}}, \rightarrow_{\mathcal{M}} \rangle$*,* $\mathcal{M}$ *is a full-control mental model for* $\mathcal{S}$*, which is denoted* $\mathcal{M}$ **fc** $\mathcal{S}$*, if* $\mathcal{M}$ *is deterministic and for all traces* $\sigma \in \mathcal{L}^*$ *such that* $s_{\mathcal{S}} \in s_{0\mathcal{S}}$ **after** σ *and* $\{s_{\mathcal{M}}\} = s_{0\mathcal{M}}$ **after** σ *:*

$$A^c(s_{\mathcal{S}}) = A^c(s_{\mathcal{M}}) \text{ and } A^o(s_{\mathcal{S}}) \subseteq A^o(s_{\mathcal{M}}).$$

A full-control mental model is therefore a deterministic HMI-LTS representing the required information for an operator to interact with a system to the full extent of its possibilities, and without surprises. Full-control mental models are minimal if they have a minimal number of states compared to other full-control mental models of the same system. Also, being *full-control deterministic* is the property of all the systems for which there exists a full-control mental model.

Minimal full-control mental models are important because they represent the minimal model that a perfect operator should learn. Compact training material and user guides should describe a minimal full-control mental model. As already stated the introduction, different algorithms exist to generate such models.

HMI-LTS DECOMPOSITION

While minimal full-control mental models are perfect in terms of control, they are inefficient when training operators as they require to be completely mastered before using a system. We provide a way to split huge models into smaller ones that can be learned independently and reassembled to form larger models.

In this section, we define a new merge operator that combines two HMI-LTSs and we claim that this operator is a natural way to encode the increase of knowledge arising from learning new partial models. We provide a finite decomposition of any HMI-LTS into in a set of basic HMI-LTSs.

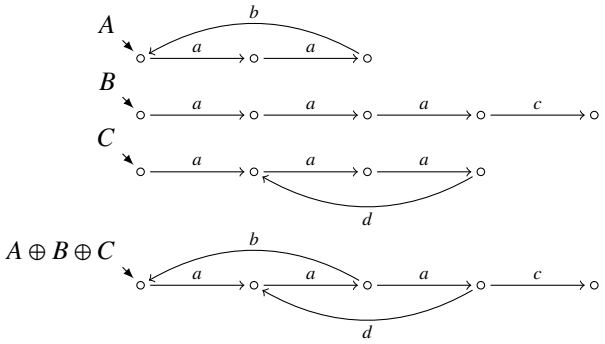

Figure 3. Example of the merge operation on three HMI-LTSs

HMI-LTS merging

Merging two HMI-LTSs produces a third HMI-LTS much like the traditional choice operator, except that common prefixes are merged. This is a kind of lazy choice, as the final behaviour does not commit to behave like the first or the second operand until a decision is required. Notice that the definition does not rely on observations and commands. This definition can therefore be generalized to LTSs.

DEFINITION 5 (MERGE). *The* merge *of two deterministic HMI-LTSs* $A = \langle S_A, \mathcal{L}^c_A, \mathcal{L}^o_A, s_{0A}, \rightarrow_A \rangle$ *and* $B = \langle S_B, \mathcal{L}^c_B, \mathcal{L}^o_B, s_{0B}, \rightarrow_B \rangle$*, denoted* $A \oplus B$*, is an HMI-LTS* $\langle S, \mathcal{L}^c, \mathcal{L}^o, s_0, \rightarrow \rangle$ *where* $\mathcal{L}^c = \mathcal{L}^c_A \cup \mathcal{L}^c_B$*,* $\mathcal{L}^o = \mathcal{L}^o_A \cup \mathcal{L}^o_B$ *and* S *is a partition of* $S_A \uplus S_B$ *such that*

1. s_0 *contains at least* $\{s_{0A}, s_{0B}\}$*;*
2. S *is the finest partition of* $S_A \uplus S_B$ *such that for all* $(m, a, m') \in \rightarrow_A$ *and* $(n, a, n') \in \rightarrow_B$ *with* $m, n \in X$ *for some* $X \in S$*, there exists* $Y \in S$ *such that* $\{m', n'\} \subseteq Y$*; and*
3. $\rightarrow$ *is the set of transitions* (X, a, Y) *for which there exists* $x \in X$ *and* $y \in Y$ *such that* $(x, a, y) \in \rightarrow_A$ *or* $(x, a, y) \in \rightarrow_B$*.*

In this definition, S is always well defined. It can be computed by starting with a complete partition where each state is a different element and merging all the states that do not respect the required criterion. This process stops when the criterion is enforced and this happens within a finite number of steps as it must end when the partition contains only one element with all the states in it. The merge of two deterministic HMI-LTS is unique, but this is not necessarily the case in general.

An example of the action of the merge operator is given in figure 3. This example uses the fact that the merge operator is associative. The operator is also commutative. While commutativity can be assumed from the symmetry of the definition, commutativity is more complex and the demonstration is left to the reader.

We can show that the result of merging two deterministic HMI-LTSs is deterministic. Indeed, as the two operands of

the merge are deterministic, they cannot contain τ transitions and so their merge is free of τ transitions too. Let's assume that the result contains a state X such that there exists two transitions with the same label a leading to different states Y and Y'. This means that there exists $(m, a, m') \in \rightarrow_A$ and $(n, a, n') \in \rightarrow_B$ such that $m, n' \in X$, $m' \in Y$ and $n' \in Y'$ which violates the property on S. Namely, m' and n' must belong to the same state Y. The resulting HMI-LTS can contain no τ transitions and no fork where a transition with a same label leads to two different states. This is sufficient to prove that it is deterministic.

The HMI-LTS $A \oplus B$ can switch his behaviour from A to B provided A can reach a state that was merged with a state of B. This conversely holds from B to A. If the HMI-LTS can switch from A to B and from B to A, then it can alternate its behaviour arbitrarily often. We can see that this operator is different from the traditional choice operator because it is more than the union of the traces. It allows to build complex behaviours from two simple models. In figure 3, we can see that the trace $a, a, a, d, a, b, a, a, c$ was not possible on the different models but is valid on their merge.

This operator is useful because the set of traces of a merge is always larger or equal to the union of the traces of the merged transitions systems. This means that the possible behaviours of a merge can be richer than the union of the behaviours of its operands. This is needed to ensure that the decomposition of a big system is a small set of small systems. By comparison, the behaviours of a choice are exactly the union of the behaviours of its operands. For synchronous parallel composition, the resulting behaviours are the intersection of the behaviours of the two operands if we synchronise on the union of the alphabets.

Furthermore, the merge operator enforces the interpretation of HMI-LTSs as scenarios. When a scenario loops or terminates, the system is assumed to have returned in a state equivalent to the initial one. In particular, the scenario is assumed to be repeatable infinitely often unless explicitly stated. When an HMI-LTS loops to a given state, it should that the system has returned to a state that is completely equivalent to the initial one for controllability purposes.

The merge operator is therefore great for decomposing systems and is a natural way to encode how mental models grow when learning new ones.

Basic HMI-LTSs

In this section, we explore the decomposition induced by the merge operator on the HMI-LTSs.

The merge operator naturally defines a partial order relation on the HMI-LTSs. The merge order is such that $A \leq_\oplus B$ if and only if $A \oplus B = B$. The strict partial order relation also requires A to be different from B. We can intuitively see that it well defined because merging HMI-LTSs can only increases the set of described behaviours and the merge order captures this.

Furthermore, due to the definition of the merge order, the set of deterministic HMI-LTSs is lattice-structured. Indeed, any two HMI-LTSs A and B are (upper) bounded by $A \oplus B$. Therefore, there exists minimal elements called atoms. These are the HMI-LTSs with only one transition. If the lattice was atomistic, we would be able to generate generate all the deterministic HMI-LTSs by merging some of its atoms. This is not the case as we can see in figure 4. There is no way to obtain the graph B by merging HMI-LTSs with only one transistion. However, there exists a larger family of HMI-LTSs that can generate all the deterministic HMI-LTSs, we call them *basic* HMI-LTSs.

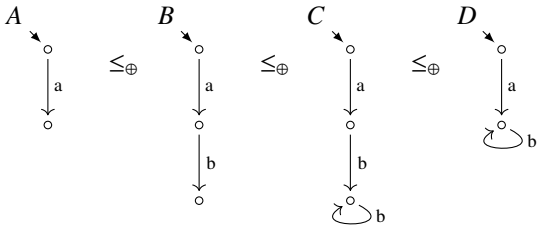

Figure 4. Illustration of the order relation on basic HMI-LTSs. For example, we have $C \leq_\oplus D$ because $C \oplus D = D$.

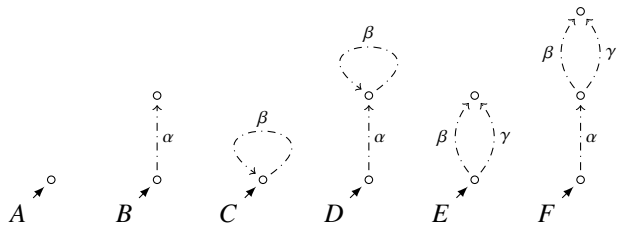

Figure 5. Different shapes of basic HMI-LTSs. They can be A) empty, B) sequences, C) loops, D) lassos and E,F) tulips with and without stem. Dotted lines represent any oriented sequence of states and transitions. All these shapes are degenerated tulips where action sequences α, β and γ can be empty

DEFINITION 6 (BASIC HMI-LTS). *A deterministic HMI-LTS A is* basic *if it cannot be decomposed into two strictly smaller HMI-LTSs. That is, for all HMI-LTS X, Y such that $X \oplus Y = A$, either $X = A$ or $Y = A$.*

It turns out that such basic HMI-LTS take the form of be single loops, single sequences, lassos or tulips. Loops and sequences can be seen as degenerated lassos with no stem or no loop. The fully degenerated lasso is the HMI-LTS with no transitions at all. Finally, a tulip is a branching HMI-LTS where the two branches reunite in the last state. Like lassos, they may have no stem. All these shapes are drawn in figure 5.

Any finite deterministic HMI-LTS can be decomposed into a finite set of basic HMI-LTS. This arises from the fact that any HMI-LTS is the merge of a basic HMI-LTS and another HMI-LTS strictly smaller than the previous one. Were it not the case, that HMI-LTS would be basic itself. By induction on the remaining HMI-LTS, we show that it eventually reduces to the empty HMI-LTS after a finite number of basic HMI-LTS removal. All the removed basic elements form a set that we call the decomposition of the HMI-LTS.

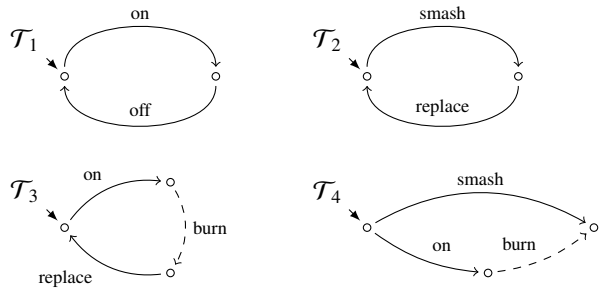

Figure 6. All the basic HMI-LTSs of the Lamp mental model defined in figure 1. $\mathcal{T}_1$ is the only mental model that does not control the Lamp system of figure 1. $\mathcal{T}_{1,2,3}$ are loops and $\mathcal{T}_4$ is a tulip. Amongst the three loops, $\mathcal{T}_1$ represents the fact that the lamp can be switched on and off forever, $\mathcal{T}_2$ that it can be smashed and replaced forever and $\mathcal{T}_3$ that a lamp can be turned on and replaced when it burns to turn it on again. Being a tulip, $\mathcal{T}_4$ has a different meaning. It expresses the fact that smashing a lamp is equivalent to turn it on and observe it burn.

A decomposition is non-redundant if it does not contain two elements such that one is strictly smaller than some other with respect to the merge order defined above. The decomposition algorithm just sketched always produces a non-redundant decomposition because each basic HMI-LTS contains actions that were not part of the previously removed basic elements, and that are removed with it. For example, the decomposition of the Lamp system into $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4\}$ as shown in figure 6 is non redundant.

A decomposition is minimal if no other decomposition of the same HMI-LTS contains less basic elements. The size of a minimal decomposition is called the *complexity* of an HMI-LTS. Minimal decompositions of the Lamp system contain exactly three elements so its complexity is 3. With the basic HMI-LTSs defined in figure 6, we see that $\mathcal{T}_4$ states the equivalence of the "smash" and "on, burn" traces. This implies that $\mathcal{T}_2$ is equivalent to $\mathcal{T}_3$ in a set containing $\mathcal{T}_4$. The two minimal decompositions of the Lamp system are $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_4\}$ and $\{\mathcal{T}_1, \mathcal{T}_3, \mathcal{T}_4\}$.

We know how to decompose an HMI-LTS into basic elements, and that decomposition gives us a measure of the complexity of that HMI-LTS.

FULL-CONTROL MENTAL MODEL DECOMPOSITION

In this section, we show that it is possible to build a set of tasks that each control a given model and can be combined into a full-control mental model.

The main idea is to decompose a full-control mental model of the system into basic subgraphs. It appears that basic subgraphs can be completed to form tasks that can control the system. This means that the completed basic subgraphs of a full-control mental model of a system form a set of independent compatible mental models that can be merged to reproduce the behaviour of the full-control mental model.

Basic subgraphs

A subgraph of a graph $\mathcal{G}$ is a graph that contains some of the edges of $\mathcal{G}$. This notion can be extended to HMI-LTSs.

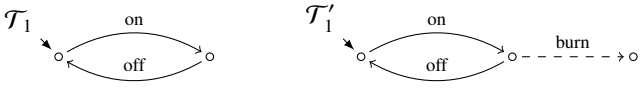

Figure 7. The observation completion of $\mathcal{T}_1$ with respect to the only minimal full-control mental model of Lamp, which is Lamp itself, is $\mathcal{T}'_1$. It contains all the observation transitions from Lamp reachable from $\mathcal{T}_1$. The other basic models $\mathcal{T}_{i,i\neq 1}$ need not be completed. The interpretation of $\mathcal{T}'_1$ is that the lamp can be switched on and off forever as long as it does not burn. When the burning occurs, either the objective is reached or the user is stuck. To unblock the situation, the user can for example read the manual to increase its knowledge of the system or ask a more experienced user.

DEFINITION 7 (SUBGRAPH). *Given two HMI-LTS $\mathcal{T} = \langle S_\mathcal{T}, \mathcal{L}^c, \mathcal{L}^o, s_{0\mathcal{T}}, \rightarrow_\mathcal{T}\rangle$ and $\mathcal{M} = \langle S_\mathcal{M}, \mathcal{L}^c, \mathcal{L}^o, s_{0\mathcal{M}}, \rightarrow_\mathcal{M}\rangle$, $\mathcal{T}$ is a subgraph of $\mathcal{M}$, denoted $\mathcal{T} \subseteq \mathcal{M}$, if $S_\mathcal{T} \subseteq S_\mathcal{M}$, $s_{0\mathcal{T}} = s_{0\mathcal{M}}$ and $\rightarrow_\mathcal{T} \subseteq \rightarrow_\mathcal{M}$*

Given two subgraphs $\mathcal{T}$ and $\mathcal{T}'$ of an HMI-LTS $\mathcal{M}$, we have the nice property that their merge $\mathcal{T} \oplus \mathcal{T}'$ is a subgraph of $\mathcal{M}$ up to a relabelling of the states. That is, the merge of two subgraphs of $\mathcal{M}$ is isomorphic to some subgraph of $\mathcal{M}$. This can be seen from the fact that $\oplus$ merges states that can be reached with the same traces, and that states must correspond to the same state of $\mathcal{M}$ as $\mathcal{M}$ is deterministic.

Therefore, any HMI-LTS can be decomposed into a non-redundant finite set of its basic subgraphs.

Tasks

Starting from a full-control mental model $\mathcal{M}$ of a system $\mathcal{S}$, we can decompose it into a set of basic HMI-LTSs. However, these basic HMI-LTSs do not necessarily control $\mathcal{S}$. To achieve this property, they need to be completed with respect to observations. This holds because of the symmetric nature of the control property. A mental model that controls a system must accept all the observations of that system, but is allowed to ignore commands.

If we call $\rightarrow^o_\mathcal{S}$ the transition relation of $\mathcal{S}$ restricted to observations, then any basic subgraph of a full-control mental model can be completed with $\rightarrow^o_\mathcal{S}$ in order to control $\mathcal{S}$. Of course, only the connected component reachable from the initial state should be kept after the completion. Figure 7 shows the completion of the basic HMI-LTS $\mathcal{T}_1$ from figure 6.

DEFINITION 8 (OBSERVATION COMPLETION). *Given an HMI-LTS $\mathcal{M} = \langle S, \mathcal{L}^c, \mathcal{L}^o, s_0, \rightarrow^c \cup \rightarrow^o\rangle$ and one subgraph $\mathcal{T} = \langle S_\mathcal{T}, \mathcal{L}^c, \mathcal{L}^o, s_0, \rightarrow_\mathcal{T}\rangle$ of $\mathcal{M}$, the* observation completion *of $\mathcal{T}$ is an HMI-LTS $\mathcal{T}'$ such that $\mathcal{T}'$ is the connected component of $\langle S, \mathcal{L}^c, \mathcal{L}^o, s_0, \rightarrow_\mathcal{T} \cup \rightarrow^o\rangle$ reachable from s_0.*

The observation completion of any subgraph of a full-control mental model $\mathcal{M}$ controls the intended system. Indeed, such a completed subgraph cannot prevent observations from occurring as the full-control mental model does not, and the completed graph has all the observations from the system. In particular, the observation completion of basic subgraphs of full-control mental models of a system $\mathcal{S}$ control that system $\mathcal{S}$. These elements also have the nice property of merging

into completed subgraphs of $\mathcal{M}$ that themselves have control over $\mathcal{S}$.

DEFINITION 9 (BASIC TASK).
Given a full-control mental models $\mathcal{M} = \langle S, \mathcal{L}^c, \mathcal{L}^o, s_0, \rightarrow \rangle$ where $\rightarrow = \rightarrow^c_{\mathcal{M}} \cup \rightarrow^o_{\mathcal{M}}$, a basic task is a mental model $\mathcal{T} = \langle S_{\mathcal{T}}, \mathcal{L}^c, \mathcal{L}^o, s_0, \rightarrow_T \rangle$ such that $\rightarrow_T = \rightarrow^o_{\mathcal{M}} \cup \rightarrow_b$ and $\langle S_{\mathcal{T}}, \mathcal{L}^c, \mathcal{L}^o, s_0, \rightarrow_b \rangle$ is a basic subgraph of $\mathcal{M}$.

With this definition, we can state that any full-control deterministic system is fully controlled by the merge of a set of basic tasks. As an example, $\mathcal{T}'_1$, $\mathcal{T}_2$ and $\mathcal{T}4$ form such a set from figures 6 and 7 form such a set.

THEOREM 1 (TASK DECOMPOSITION). *Any finite fc-deterministic HMI-LTS $\mathcal{S}$ can be decomposed into a finite set $T = \{\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_n\}$ of basic tasks such that*

- *each $\mathcal{T}_i$ controls $\mathcal{S}$;*
- *for each subset $I \subset \{1, 2, \ldots n\}$ of indices, the partial merge $\bigoplus_{i \in I} \mathcal{T}_i$ of elements of T controls $\mathcal{S}$; and*
- *the complete merge $\bigoplus_{i=1}^{n} \mathcal{T}_i$ has full-control over $\mathcal{S}$.*

PROOF. By definition, any fc-deterministic HMI-LTS $\mathcal{S}$ has at least one minimal full-control mental model $\mathcal{M}$. We have shown that such a full-control mental model can be decomposed into a finite set of basic tasks which are completed basic subgraphs. Because these elements are completed subgraphs they have control over $\mathcal{S}$ and any partial merge of these elements have too. As the elements are the completion of the decomposition of $\mathcal{M}$ into basic HMI-LTS, their full merge will be exactly $\mathcal{M}$, and therefore fully controls $\mathcal{S}$. This proves that there exists a decomposition of $\mathcal{S}$ meeting the required properties. □

Task-complexity

The decomposition of an fc-deterministic system into a set of tasks is far from unique. Indeed, there exists an infinity of full-control mental models for a given system, and for each full-control mental model there may exist multiple decompositions into basic tasks.

Nevertheless, we define the task-complexity of a system as the size of the smallest set of tasks that can be merged into a full-control mental model of that system. This metric measures the number of small tasks that an operator needs to learn before being able to control all the features of the system.

This metric is different from both the number of states and the number of transitions which are the most common measures of transition systems.

CONCLUSION

We have defined the merge operation that represents how a human augments its mental model by leaning new mental models. We have shown that this operation is more natural than the parallel synchronisation operation and more powerful than the classical choice operation.

We have shown how tasks can be split into simple operations. In particular, we have shown that the full-control mental model of a system is itself a composition of basic tasks. With this decomposition, we have defined a measure of the complexity of HMI-LTSs based on tasks.

We know how to generate decompositions of a system into tasks, and we know that there exists a minimal decomposition, but we do not yet know how to generate minimal decompositions. In the near future, we expect to work on an algorithm capable of computing one.

This works can be used to validate system design by detecting irreducible large tasks. But, more importantly, this works lays the ground for automated generation of user manuals and assisted generation of other training material. With small well-defined tasks, it is possible to decompose manuals into self-contained chapters.

REFERENCES

1. Sébastien Combéfis. 2013. *A Formal Framework for the Analysis of Human-Machine Interactions*. Vol. 459. Presses universitaires de Louvain.
2. Sébastien Combéfis, Dimitra Giannakopoulou, Charles Pecheur, and Michael Feary. 2011a. A formal framework for design and analysis of human-machine interaction. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*. IEEE, 1801–1808.
3. Sébastien Combéfis, Dimitra Giannakopoulou, Charles Pecheur, and Michael Feary. 2011b. Learning system abstractions for human operators. In *Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering*. ACM, 3–10.
4. Sébastien Combéfis and Charles Pecheur. 2009. A bisimulation-based approach to the analysis of human-computer interaction. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*. ACM, 101–110.
5. Philippe A Palanque, Rémi Bastide, and Valérie Sengès. 1995. Validating interactive system design through the verification of formal task and system models.. In *EHCI*. 189–212.
6. Peter J Ramadge and W Murray Wonham. 1987. Supervisory control of a class of discrete event processes. *SIAM journal on control and optimization* 25, 1 (1987), 206–230.
7. John Rushby. 2002. Using model checking to help discover mode confusions and other automation surprises. *Reliability Engineering & System Safety* 75, 2 (2002), 167–177.
8. Jan Tretmans. 2008. Model Based Testing with Labelled Transition Systems. In *Formal Methods and Testing*, Robert M. Hierons, Jonathan P. Bowen, and Mark Harman (Eds.). Lecture Notes in Computer Science, Vol. 4949. Springer Berlin Heidelberg, 1–38.