Modal Specifications for Composition of Agent Behaviors

Hikmat Farhat, Guillaume Feuillade

Modal Specifications for Composition of Agent Behaviors

Hikmat Farhat[1], Guillaume Feuillade[2]
[1]*Notre Dame University-Louaize, Lebanon*
[2]*IRIT, Universite Paul Sabatier, Toulouse, France*
hikmat.farhat@acm.org, guillaume.feuillade@irit.fr

Keywords: Automated planning, behavior composition

Abstract: The goal of the behavior composition problem is to build a complex target behavior using several agent behaviors. We propose two extensions to the framework where agent behaviors are modeled by finite transition system and where the composition is done by coordinating the actions of the agents. The first extension is done by making the composition indirect: instead of choosing the actions of the agent, the composition is done by a controller issuing sets of instructions at each step. This allows to model problems where the agents behaviors are not fully controllable. The second extension is the use of modal specifications as a goal for the composition. These specifications express (infinite) sets of acceptable behaviors. We give an algorithm to solve the extended composition problem and we show that these two extensions retain the important properties of the initial framework and that the synthesis algorithm keep the same complexity.

1 INTRODUCTION

The behavior composition problem is the problem of realizing a given target behavior by putting together, in the right way, behaviors that in themselves would not suffice to achieve the target. Behaviors are an abstraction of sequences of actions made by agents. They are suitable to describe any situation where one is interested in the scheduling : for coordination of physical agents, for agent being components of some bigger agent or for online agents such a services. When it is not the case that one unique agent can achieve a goal behavior alone, then there is a need for a composition of several agents in order to build the desired behavior.

There are many different techniques for building a composition. This paper focuses on a approach based on synthesizing a special component, akin to a controller that plays the role of a mediator between the composition and any other system or user. To this extent, the problem is a synthesis problem, which is in essence different from verification problems like the ones solved by (Lomuscio et al., 2009). Another focus of the present work is to keep a polynomial complexity while having a framework and specifications expressive enough for modeling real problems.

This paper extends the work presented in (De Giacomo et al., 2013) which considers a model where agents behaviors are described by finite transition systems. This model is referred to in the litterature as the *"Roman Model"* (Hull, 2005). In their framework, the composition is obtained by the synthesis of some super-agent that has a perfect knowledge about the other agents and that decides at each step what action of the agents must be enabled or disabled. This super-agent act as a controller from the control theory of (Ramadge and Wonham, 1989). Despite having perfect knowledge and being able to control every action of the agents, some uncertainty remains about the outcome of the actions because the agents actions may be nondeterministic. This means that the controller must take into account every possible outcome for any of the actions it enables.

The work presented in this paper is intended to provide more flexibility with a more general framework while also adding a class of specifications able to express a set of constraints over the goal behavior instead of a rigid single target behavior. In the framework we propose, the controller does not choose the agent actions but instead gives a set of instruction to the agent community. The agents then execute actions tied to the given instructions. This makes the control over the actions of the agent more indirect and offers the possibility to model composition problems where the agent making the composition does not control every move of the other agents.

The extension of the notion of target behavior is done using the modal specifications of (Feuillade and

Pinchinat, 2007). A goal for the composition, expressed with modal specifications, is a set of acceptable behaviors. These behaviors are expressed by a language that states when an action is mandatory and when it is optional. These specifications also come with some nice properties such as being easy to design and modular.

We show in this paper that these two extensions have no negative impact on the complexity of the problem and we provide the synthesis algorithm. We also show that the possibility of building a structure capturing all the controllers still exist in our work as well as the existence of a most permissive controller.

In section 2 the framework is presented and followed by a discussion about the expressivity of the framework and the advantages it offers. In section 3, we present the results for the existence of a solution to the composition problem and the notion of controllability. Section 4 is dedicated to the synthesis of controllers. Finally we conclude in section 5.

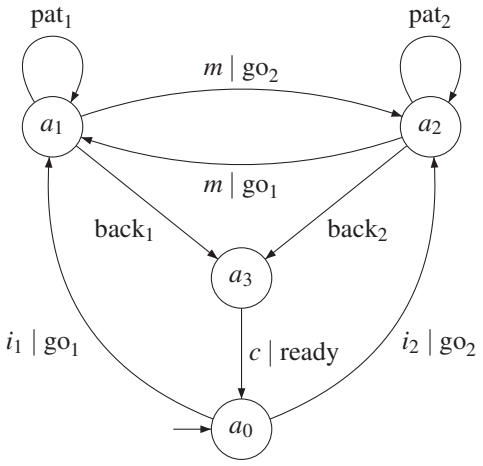

Figure 1: Drone behavior

2 FRAMEWORK

2.1 Behaviors

In this section we define two notions of behavior. First we define a generic notion of behavior based on finite state transition systems. Next we introduce the behavior model used for describing the agents. This latter model adds preconditions to the actions of the first behavior model. These preconditions represent the instructions that are given to the agent and trigger a corresponding action or even a choice between a set of actions.

We start by fixing a finite set of actions Act and a finite set of instruction Ins that we use all along the paper.

Generic behaviors are defined as transition systems labeled by the actions of Act. That is: a behavior $\mathcal{B}$ is a tuple $\langle B, b^0, \delta_B \rangle$ where B is a set of states, b^0 is the initial state and $\delta_B \subseteq B \times \mathrm{Act} \times B$ is a transition relation. We say that a behavior is finite when its set of states is finite.

We represent agents by their possible behaviors. In this paper agents are represented by transition systems labeled by a pair of one instruction and one action. That is: an agent $\mathcal{A}$ is a tuple $\langle A, a^0, \delta_A \rangle$ where A is a finite set of states, a^0 is the initial state and $\delta_A \subseteq A \times (\mathrm{Ins} \cup \{\varepsilon\}) \times \mathrm{Act} \times A$ is the transition relation.

For behaviors, we write $b \xrightarrow{\sigma} b'$ for $(b, \sigma, b') \in \delta_B$ and for agents, we write $a \xrightarrow{i|\sigma} a'$ for $(a, i, \sigma, a') \in \delta_A$ and $a \xrightarrow{\sigma} a'$ for $(a, \varepsilon, \sigma, a') \in \delta_A$.

Intuitively, $a \xrightarrow{i|\sigma} a'$ means that, upon receiving the instruction i in state a, agent $\mathcal{A}$ will perform action σ and enter state a'. The instruction i is the *precondition* of the action σ. The meaning of the special symbol ε is that no precondition is required for the agent to perform the action. Thus $a \xrightarrow{\sigma} a'$ means that the agent $\mathcal{A}$, in state a, will perform the action σ spontaneously and enter state a'.

Note that when $a \xrightarrow{i|\sigma} a'$ and $a \xrightarrow{i|\sigma'} a''$ then the agent may choose between the two actions σ and σ' upon recieving i. This can also be the case that the agent can execute the same action nondeterministicaly.

Example 1. *Consider a flying drone whose behavior is depicted on Figure 1. This drone can patrol two different areas named* α_1 *and* α_2 *to watch for fire starts. At any time, it may need maintenance in which case it has to go back to its base to undergo the necessary operations such as refueling for example.*

The drone starts in state a_0 *where it is ready at the base waiting for instructions. If given the instruction* i_1*, the drone will move to the area* α_1 *represented by state* a_1 *using action* go_1 *and if given the instruction* i_2*, the drone will move to the area* α_2 *represented by state* a_2 *using action* go_2*. In either* a_1 *or* a_2*, the drone will patrol the area by itself using the action* pat_1 *or* pat_2*. It may also go back to the base for maintenance using action* back_1 *or* back_2 *and entering state* a_3*. Upon receiving the instruction c, it will output the action* ready *when the maintenance is over and return to the initial state* a_0*. When in* α_1 *or* α_2*, it can be given*

the instruction m to change area.

The idea behind this agent model is that when given instructions, agent will develop some possibly infinite behavior. Note that the agents may be nondeterministic and that there is no constraint on the relation between instruction and actions: an instruction given to an agent in a given state may result in different actions or may trigger a nondeterministic action.

When several agents operate together, forming an *agent community*, they are modeled as a single bigger agent. Formally, the community of the agents $\mathcal{A}_1 = \langle A_1, a_1^0, \delta_{A_1} \rangle, \ldots, \mathcal{A}_n = \langle A_n, a_n^0, \delta_{A_n} \rangle$ is represented by the agent $\mathcal{A} = \langle A, a^0, \delta_A \rangle$ which is the asynchronous composition of the agents. That is:

- its set of state A is $A_1 \times \ldots \times A_n$
- its initial state a^0 is $(a_1^0, \ldots, a_n^0)$
- its transition relation δ_A satisfies $((a_1, \ldots, a_n), i, \sigma, (a'_1, \ldots, a'_n)) \in \delta_A$ if there exists $k \in \{1, \ldots, n\}$ such that $(a_k, i, \sigma, a'_k) \in \delta_{A_k}$ and for all $l \neq k$, $a'_l = a_l$.

2.2 Modal specifications

The goal of agent composition is often given as a *target behavior* (Sardina, 2007; Balbiani et al., 2008). We propose in this paper to extend it to a set of possible target behaviors using *modal specifications* for this purpose. Modal specifications have been introduced to model objectives for control problems (Feuillade and Pinchinat, 2007). The definition we use here is the following :

Definition 1 (Modal specification)**.** *A* modal specification *is a tuple* $\mathcal{S} = \langle S, s^0, \text{May}, \text{Must} \rangle$ *where*

- S *is a set of states,*
- s^0 *is the initial state,*
- $\text{May} \subseteq S \times \text{Act} \times S$ *a* deterministic *transition function of allowed transitions,*
- $\text{Must} \subseteq \text{May}$ *a* deterministic *transition function of necessary transitions*

We say that a behavior $\mathcal{B} = \langle B, b^0, \delta_B \rangle$ *satisfies* a modal specification $\mathcal{S} = \langle S, s_0, \text{May}, \text{Must} \rangle$ if there exists a *satisfaction relation* $\rho \subseteq (B, S)$ with $(b^0, s^0) \in \rho$ and for all $(b, s) \in \rho$ and $\sigma \in \text{Act}$ we have:

(i) $(s, \sigma, s') \in \text{Must}$ implies $\exists b' \in B$ with $b \xrightarrow{\sigma} b'$ and $(b', s') \in \rho$,

(ii) $b \xrightarrow{\sigma} b'$ implies $\exists s' \in S$ with $(s, \sigma, s') \in \text{May}$ and $(b', s') \in \rho$.

The definition we use here is the one for the *modal automaton* and is different from the original definition of modal specification. However, both definitions have been proved to be equivalent in (Feuillade and Pinchinat, 2007). Remark that in order to simplify the algorithms of the paper, we require that the two transition relations of modal specification are deterministic. This is done without loss of generality, the usual determinization of automaton being easy to generalize to modal specifications.

Regarding the expressivity of modal specifications, we show that they are able to express the two usual composition objectives : simulation and bisimulation.

Let $\mathcal{B}$ be a behavior and $\mathcal{S} = \langle S, s_0, \text{May}, \text{Must} \rangle$ be a modal specification. When May = Must, one can verify that $\mathcal{B}$ satisfies $\mathcal{S}$ if and only if $\mathcal{B}$ is bisimilar to the behavior $\mathcal{B}_\mathcal{S} = \langle S, s^0, \text{Must} \rangle$. This is consequence of the fact that the satisfaction relation ρ has to be a bisimulation.

For a modal specification $\mathcal{S}$ to specify that a behavior is similar to a target behavior $\mathcal{T} = \langle T, t^0, \delta_t \rangle$ it suffices that $\mathcal{S} = \langle T \cup \{\top\}, t^0, \text{May}, \text{Must} \rangle$ where $\top$ is a special state, $\text{Must} = \delta_t$ and May is made into a complete function. That is for any $t \in T$ and $\sigma \in \text{Act}$, if $(t, \sigma, t') \in \delta_t$ for some t', then $(t, \sigma, t') \in \text{May}$ and if there is no such t', then $(t, \sigma, \top) \in \text{May}$ and finally $(\top, \sigma, \top) \in \text{May}$. The idea for this definition is that $\top$ act as a sink for any transition not in $\mathcal{T}$ and allows the behavior to include freely any transition. As a consequence, only the Must part of the specification is relevant, practically meaning that condition (ii) of the definition of satisfaction by a behavior $\mathcal{B}$ is trivial. In the end, the satisfaction relation ρ is a simulation between T and B.

Modal specifications go further than simulation and bisimulation by allowing to express more constraints. One can for example specify that if a given sequence of action occurs then some actions must occur afterward. This is particularly useful in the case of failure of one agent or in cases of uncontrollable events: when an undesired action is taken by the agent, one may want the community to take appropriate measures in reaction. The following example emphasizes this expressivity.

Example 2. *Consider the modal specification* $\mathcal{S}_d$ *of Figure 2. In this figure, the transitions of* Must *are the solid lines and the transitions of* May *are both the solid and dashed lines. This modal specification is meant to specify the behavior of a community composed of two of the drones represented in Figure 2. The objective of the community is to maintain one*

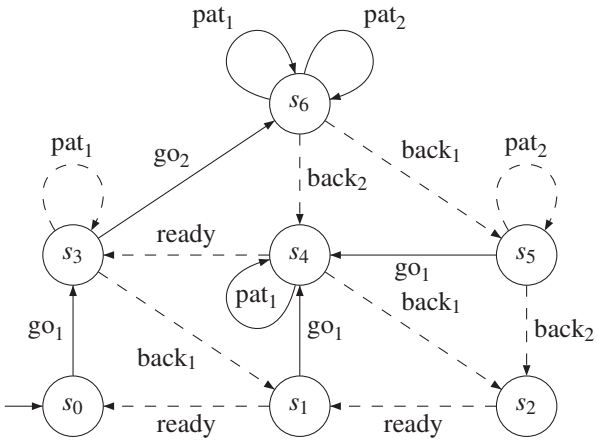

Figure 2: Drone composition specification $\mathcal{S}_d$

drone patrolling in each of the two area whenever possible. If one drone must return to the base for maintenance, the agents must ensure that the remaining drone patrols in the area α_1 since it is the most sensible one. The modeling of these constraints is done by the specification $\mathcal{S}_d$. The key states of the specification can be understood as follow :

- *state s_0 is the initial state. Both drones are ready and at the base.*
- *in state s_6, there is one drone in each area.*
- *in state s_5, the drone which was in area α_1 is back to base. The other one is still in area α_2*
- *in state s_4, one drone is at the base and the other is in area α_1.*

The Must *transition ensures that when both drone are available they are sent in both area. It also ensures that whenever the drone in area α_1 is back to base, the drone in area α_2 is sent to area α_1 (transition $(s_5, \mathrm{go}_1, s_4)$). When a drone is in position, the* Must *transition obligates it to patrol the area. The* May *transition specify which transitions are forbidden: each transition not in* May *should not happen. This is the case for example for the transition go_2 in state s_6 which would send the drone in area α_1 to area α_2.*

Another interest of modal specification is their underlying logical background: in (Feuillade and Pinchinat, 2007) it is shown that they correspond to a logic called the *conjunctive nu-calculus*[1]. This brings one useful property: one can combine several modal specifications with an `and` operator. In practice, this is done in our setting by merging the initial state of each modal specification and then applying the determinization algorithm for the May transition. This is useful for adding simple constraints to a more complex specification. For example the constraint "every action α must be followed by an action β" can be modeled by a two states modal specification and added to a global specification with the `and` operator.

The underlying logical setting allows for an implicit declaration of the modal specification. We emphasize that this is an important improvement over bisimulation/simulation based techniques where all the states and transition of the behavior have to be given explicitly.

One likely practical use of modal specification is to express some behavior target the compositon has to simulate and then add some further constraints using additional independent specifications.

[1]The conjunctive nu-calculus is a fragment of the mu-calculus without disjunction and with only greatest fixpoints

2.3 Control problem

In our setting, the composition is done by a special agent. This agent has a perfect knowledge of the current state of the other agents and a perfect recall of the previous events although we will show that this latter knowledge is not mandatory. Given this knowledge, the agent elects a set of instruction to be given to the agent community. These instructions are not given to any particular agent but to the community, and any agent may respond to the instruction if there is an available action in its current state whose precondition matches the instruction. The set of instructions given to the community must be chosen such that the resulting behavior respects the specification.

We present the agent in charge of the composition as a controller since one may consider it removes instructions from the whole set Ins thus restraining the available behaviors of the community. This setting is different from the usual control settings because the controller operates here in an indirect way by issuing instructions.

We first define the notion of history. Let $\mathcal{A} = \langle A, a^0, \delta_A \rangle$ be an agent community. An *history* is a finite sequence of transitions $(a_0 \xrightarrow{\sigma_0} a_1 \ldots \xrightarrow{\sigma_{k-1}} a_k)$ where

- the history begins with a_0 which is a^0,
- for all $0 \leq i \leq k-1$, $a_i \xrightarrow{i|\sigma_i} a_{i+1}$ for some $i \in \mathrm{Ins} \cup \{\varepsilon\}$.

We use $\mathcal{H}_\mathcal{A}$ for the set of histories for $\mathcal{A}$.

A *controller* C for the community $\mathcal{A}$ is a function $C : \mathcal{H}_{\mathcal{A}} \to 2^{\text{Ins}}$.

The *controlled behavior* of an agent or community of agents $\mathcal{A}$ by a controller C for the specification S is the behavior $\mathcal{B}_{\mathcal{A}}^{C} = \langle B, b^0, \delta_B \rangle$ such that

- $B = \mathcal{H}_{\mathcal{A}}$
- b^0 is (a^0), the empty history
- $h \xrightarrow{\sigma} h'$ in $\mathcal{B}_{\mathcal{A}}^{C}$ if and only if h' is h augmented with the transition $a_k \xrightarrow{\sigma_k} a_{k+1}$ where a_k is the last state of h and
  - either $a_k \xrightarrow{\sigma_{k+1}} a_{k+1}$ is a transition of $\mathcal{A}$
  - or there exists $i \in C(h)$ such that $a_k \xrightarrow{i|\sigma_k} a_{k+1}$ in $\mathcal{A}$

The controlled behavior $\mathcal{B}_{\mathcal{A}}^{C}$ is the unfolding of $\mathcal{A}$ where any transitions with a non-empty precondition is kept (without the precondition) only if the controller allows it, i.e. when the controller outputs the corresponding instruction.

Example 3. *Let us consider again the example composed of two drones from Figure 1. Let us consider the controller that is initially issuing all instructions* $\{i_1, i_2, m, c\}$. *Two of these instruction are without any effect, they are m and c, the other two enable the transition* go_1 *and* go_2 of each drone. *This means that the controlled behavior has 4 transitions in its initial state (the empty history) being the two transition for the two drones.*

At some point, if the controller issues the empty set of instructions while the first drone is in state a_1 *and the second is in state* a_2, *there is still 4 uncontrollable transitions in the corresponding controlled behavior state, they are* $\text{pat}_1, \text{back}_1, \text{pat}_2$ *and* back_2.

Now we can state the composition problem in our framework.

Composition Problem. *Given an agent community* $\mathcal{A}$ *and a modal specification* $\mathcal{S}$, *does there exist a controller* C *for* $\mathcal{A}$ *such that the behavior* $\mathcal{B}_{\mathcal{A}}^{C}$ *satisfies* $\mathcal{S}$.

In Section 3 we show that the notion of controllability is the key to solving this problem and we provide an algorithm to compute controllability. However, when the answer to this problem is positive one usually wants the composition to be effectively computed. This is the object of Section 4 where we show that from the output of the controllability algorithm it is easy to build a specific controller but also to construct a structure that captures all the controllers and thus allows to switch from a controller to another during the execution.

2.4 Framework discussion

As stated in the introduction, the framework proposed in this paper extends the one of (De Giacomo et al., 2013) in two directions: the model for the agents and their relation with the controller is more general and the modal specifications offers a better expressivity for the goals of the composition. The contribution offered by modal specifications having already been highlighted, we aim the discussion here toward the contribution brought by the instruction-based model of agents.

In the more general version of the Roman model, the control over the agent is done by the controller selecting an agent and one action of this agents for matching one transition of the target behavior. This means that the controller have a total control over the actions of the agents except from the fact that some actions of some agent may be nondeterministic. In the framework we propose, the Roman model corresponds precisely to the special case where all the actions are subject to a precondition ; the preconditions for an action σ in the agent $\mathcal{A}_k$ being some instruction σ_k. That is, every transition is of the form $a \xrightarrow{\sigma_k|\sigma} a'$.

Because of space constraints, we do not include in the framework proposed in this paper the notion of environment and its effect upon the agents. We ensure the reader that it can be included without modifying the synthesis algorithm and the complexity results. The environment has to be considered as embedded in the agents in the framework of this paper. In practice, the environement may be responsible for many occurences of nondeterminism, in particular whenever an agent has two different possible actions when given an identical instruction.

The improvements over the Roman model offered by our framework are: first, the actions without preconditions are uncontrollable. It is reasonable that in an agent setting, some action cannot be controlled. Some effects of uncontrollable actions can be simulated by nondeterminism but this is not the case where some agent may become unresponsive to some instruction after an uncontrollable action. However, this mechanism is particularly useful for embedding the possible failure of some agent component in the objective of composition and thus making sure there is a correct answer for this failure in the solution.

Second, an agent in a specific state and given a specific instruction may respond with some different actions. This can be a consequence of the fact that the agent is an abstraction of the real agent, the fact that the instruction leave the possibility for the agent to choose autonomously its response to the instruction, or the fact that the environement may alter the

available response to the instruction at the given state.

Finally, an instruction may be given to a set of agents and not to one particular agent. This allows one to represent systems where the agents are more autonomous about their organization. This can model for example a set of elevators in the same building: when a user calls the elevator, she does not know which elevator will answer her call.

Related works also include the framework of (Balbiani et al., 2008) where the agents are controlled by communications. The main difference with this paper is that in their framework the communications are actions. This allows to consider asynchronous communications with the drawback that this causes an exponential blow-up in the composition algorithm because the communication must be removed for checking the simulation with the target behavior.

3 CONTROLLABILITY

In this section we present a notion of controllability inspired by the similar notion from control theory. Controllability captures the set of positions in the execution where the control has a solution and practically gives an optimal answer to the composition problem.

Given $\mathcal{A} = \langle A, a^0, \delta_A \rangle$ a communicating agent and $\mathcal{S} = \langle S, s_0, \text{May}, \text{Must} \rangle$ a modal specification, it is convenient to define, for each pair (a,s) of states of $\mathcal{A}$ and $\mathcal{S}$ the notion of *acceptable instruction* as the set of instructions that only produce transitions that are allowed by the specification. Formally the set of acceptable instructions is the subset $\text{AI}(a,s)$ of Ins such that for each $i \in \text{AI}(a,s)$ there is some transition $a \xrightarrow{i|\sigma} a'$ only if $(s,\sigma,s') \in \text{May}$ for some s'.

3.1 Controllability

Let $\mathcal{A} = \langle A, a^0, \delta_A \rangle$ be a communicating agent and $\mathcal{S} = \langle S, s_0, \text{May}, \text{Must} \rangle$ be a modal specification. The notion of controllability of a state a of $\mathcal{A}$ w.r.t a state s of $\mathcal{S}$ captures the fact that there is a solution to the controller synthesis problem starting in a for satisfying the specification from state s. Formally the set of controllable pairs of states in $A \times S$ is the largest relation $\rho \in A \times S$ with, for all $(a,s) \in \rho$

(i) for all $\sigma \in \text{Act}$, if $a \xrightarrow{\sigma} a'$ for some $a' \in A$ then there exists s' such that $(s,\sigma,s') \in \text{May}$ and $(a',s') \in \rho$

(ii) there exists a subset E of $\text{AI}(a,s)$ such that

- for all $i \in E$, if $a \xrightarrow{i|\sigma} a'$ for some $a' \in A$ and $\sigma \in \text{Act}$, then there exists s' such that $(s,\sigma,s') \in \text{May}$ and $(a',s') \in \rho$
- for all $(s,\sigma,s') \in \text{Must}$ either $a \xrightarrow{\sigma} a'$ or there is $c \in E$ such that $a \xrightarrow{c|\sigma} a'$

We say that a state a of $\mathcal{A}$ is *controllable* w.r.t the state s of $\mathcal{S}$ if $(a,s) \in \rho$. We say that $\mathcal{A}$ is *controllable* w.r.t $\mathcal{S}$ if $(a^0,s^0) \in \rho$.

Note that if one considers only item (i) and forgets communications, ρ is the biggest simulation between $\mathcal{A}$ and $\mathcal{S}$ where $\mathcal{S}$ is seen as an agent with May as transition function.

The following result show that controllability answers the composition problem.

Theorem 1. *There exists a controller for $\mathcal{A}$ w.r.t the modal specification $\mathcal{S}$ if and only if $\mathcal{A}$ is controllable w.r.t $\mathcal{S}$.*

Due to space constraints we do not include the proof in this paper. The idea is that the ρ relation that is computed here is the relation for the satisfaction of $\mathcal{S}$ and that the sets E are the outputs of some controller.

3.2 Algorithm for controllability

Our algorithm for computing controllability is very similar to the one for computing the biggest simulation relation between transition systems. The principle is to build the controllability relation ρ over $A \times S$ and at the same time for each pair (a,s), to build a set $E(a,s) \subseteq \text{Ins}$ which is the E set of item (ii) in the definition of controllability.

The algorithm first assigns every element of $S \times A$ to the relation ρ, and assigns the maximal set of instructions for each such element (which is the set of acceptable instruction). The next step is the iteration of a procedure for removing "bad" states until a fix-point is reached.

Controllability ($\mathcal{A}$,$\mathcal{S}$)
`Init.` : $\rho = A \times S$ and $E(a,s) = \text{AI}(a,s)$[2]
`Iterate until fix point : foreach` $(a,s) \in \rho$ `do : remove` (a,s) `from` ρ `if either of these condition is met:`

- `there exist` $a \xrightarrow{\sigma} a'$ `and there is no` s' `such that` $(s,\sigma,s') \in \text{May}$ `and` $(a',s') \in \rho$
- `there exists` $(s,\sigma,s') \in \text{Must}$ `and there is no` $i \in E(a,s) \cup \{\varepsilon\}$ `such that` $a \xrightarrow{i|\sigma} a'$ `and` $(a',s') \in \rho$.

[2]Note that the computation of AI is local to pairs of states.

Note that both items of the algorithm are done locally. Obviously, the algorithm terminates, and the bound on the number of iterations is given by the size of the set $A \times S$. If we consider that $\mathcal{A}$ is the agent community composed by $\mathcal{A}_1, \ldots \mathcal{A}_n$ then the algorithm is exponential in n. If we fix n, then the algorithm is polynomial in the size of the set $A \times S$. This complexity is the same as the one for the Roman Model and matches the lower bound of (Muscholl and Walukiewicz, 2008).

4 CONTROLLER SYNTHESIS

Since a controller chooses a set of instructions based upon some history, it is convenient to introduce a notion of *labeled history* where the controller follows the history on the specification at the same time as on the agents. Formally, given an agent $\mathcal{A} = \langle A, a^0, \delta_A \rangle$ and a history $h \in \mathcal{H}_\mathcal{A}$ with $h = (a_0 \xrightarrow{\sigma_0} a_1 \ldots \xrightarrow{\sigma_{k-1}} a_k)$, the *labeling* of h with a modal specification $\mathcal{S} = \langle S, s_0, \text{May}, \text{Must} \rangle$ is the history $h_\mathcal{S} = ((a_0, s_0) \xrightarrow{\sigma_0} (a_1, s_1) \ldots \xrightarrow{\sigma_{k-1}} (a_k, s_k))$ where $s_0 = s^0$ and $(s_i, \sigma_i, s_{i+1}) \in \text{May}$ for $0 \leq i \leq k-1$.

Remark that such labeling is unique because the May transition of the specification is deterministic. From this point on, we define controllers over labeled history instead of regular histories since there is a one on one correspondence. We use $\mathcal{H}_\mathcal{A}^\mathcal{S}$ for the set $\mathcal{H}_\mathcal{A}$ labeled by $\mathcal{S}$.

We say that a controller is *memoryless* when it does only consider the last state of the labeled history for deciding its set of instruction. Thus a memoryless controller is a function $C : A \times X \to 2^{\text{Ins}}$. In practice, a memoryless controller follows the transitions of $\mathcal{A}$ on $\mathcal{S}$ but only keep in memory the last pair of states.

4.1 Most permissive controller

Modal specifications, when used to specify control objective, have the property to have a biggest solution in the sense of the inclusion of behaviors. This property remains true in the composition setting. The consequence is that, whenever an instance of the composition problem has a solution, then amongst all the controllers, there is one that produces the largest controlled behavior. Consider the following definition.

Definition 2 (Most permissive controller)**.** *The* most permissive controller *for $\mathcal{A}$ and $\mathcal{S}$ is the controller* $\text{PC} : A \times S \to 2^{\text{Ins}}$ *where $c \in \text{PC}(a,s)$ if and only if*

- $c \in \text{AI}(a,s)$,
- *for all $a' \in A$, $s' \in S$ and $\sigma \in$ Act such that $a \xrightarrow{c|\sigma} a'$ and $(s, \sigma, s') \in$ May, a' is controllable w.r.t s'.*

Remark that this controller is memoryless. The next theorem states that the controller PC is indeed the most permissive controller in the sense that it is the one that gives the most instructions at each step.

Theorem 2. *Let $\mathcal{A} = \langle A, a^0, \delta_A \rangle$ be a communicating agent and $\mathcal{S} = \langle S, s_0, \text{May}, \text{Must} \rangle$ be a modal specification. If there exists a controller C for $\mathcal{A}$ w.r.t $\mathcal{S}$ then the most permissive controller* PC *satisfies, for each labeled history $h_\mathcal{S}$ ending on the pair of states (a,s), that $C(h) \subseteq \text{PC}(a,s)$.*

The synthesis of the most permissive controller is the synthesis of the maximal controllable subset of $A \times S$. Since the algorithm we provided in previous section for controllability works as a greatest fixpoint, is computes exactly this maximal controllable subset. As a consequence, the controller is the set E given by the algorithm for controllability.

4.2 Controller generator

The concept of controller generator has been introduced in (De Giacomo et al., 2013); the key idea is to build not a single controller but a structure capturing the set of all controllers which are solutions of the initial problem. The structure captures all the memoryless controller, but also allows to switch at any time from one to another thus capturing all the controllers in a finite way.

Definition 3 (Controller Generator)**.** *Let $\mathcal{A} = \langle A, a^0, \delta_A \rangle$ be a communicating agent and $\mathcal{S} = \langle S, s_0, \text{May}, \text{Must} \rangle$ be a modal specification. The* controller generator *for $\mathcal{A}$ and $\mathcal{S}$ is a partial mapping* $\text{CG} : A \times S \to 2^{2^{\text{Ins}}}$ *defined by : for all $a \in A$ and $s \in S$ where a is controllable w.r.t s,* $\text{CG}(a,s)$ *is the set of subsets C of* Ins *where*

(i) $C \subseteq \text{PC}(a,s)$

(ii) for each $\sigma \in$ Act such that $(s, \sigma, s') \in$ Must for some s', we have

- *either $\exists a' \in A$ with $a \xrightarrow{\sigma} a'$*
- *or $\exists a' \in A, \exists i \in C$ such that $a \xrightarrow{i|\sigma} a'$*

At each pair (a,s) where a is controllable w.r.t s, the controller generator associates a set of sets of instructions. Each of these sets is a correct set of instructions in the sense that it fulfill the local requirements of the specification and only allow transitions into some controllable states. Note that the set $\text{CG}(a,s)$ is a lattice whose greatest element is $\text{PC}(a,s)$ but that may not have a least element. The following theorem ensures that the controller generator captures all solutions of the composition problem.

Theorem 3. *Let $\mathcal{A} = \langle A, a^0, \delta_A \rangle$ be a communicating agent and $\mathcal{S} = \langle S, s_0, \text{May}, \text{Must} \rangle$ be a modal specification. If $\mathcal{A}$ is controllable w.r.t $\mathcal{S}$ then each controller C satisfies foreach labeled history $h \in \mathcal{H}_{\mathcal{A}}^{\mathcal{S}}$ ending on (a,s) that $C(h_{\mathcal{S}}) \in \text{CG}(a,s)$.*

The synthesis of the controller generator is done using the following algorithm:

- synthesize the most permissive controller PC using the algorithm for controllability
- foreach pair $(a,s) \in A \times S$ where $\text{PC}(a,s)$ is defined, compute the minimal subsets satisfying the item (ii) of the definition. The lattice $\text{CG}(a,s)$ is the set of elements included in $\text{PC}(a,s)$ and including any of these minimal subsets.

The complexity of the algorithm is bounded by the one of the algorithm for controllability multiplied by $2^{|\text{Ins}|}$. This remains exponential in the number of agents and polynomial in number of elements in $A \times S$ if we don't consider the number of elements in Ins as a parameter.

5 CONCLUSION

This paper extends the framework of the Roman Model with the notion of instructions. This indirect mean of control introduces uncontrollability in several different ways into the models. This allows one to tackle new composition problems where some agent may not behave exactly as predicted or asked to as well as problems where the agents are keeping some autonomy. We have also extended the goal of the composition from a rigid given behavior into a specification that can be more lenient than bisimulation but also more strict than simulation. We believe this is necessary whenever uncontrollable events are considered in the model. In fact, when one may request a behavior to at least include a target behavior like simulation do, it is suitable to be able to add some bounds to the behaviors that are not requested but produced by the mean of a permissive controller or by uncontrollable actions. We also stress the fact that a more lenient specification may allow to obtain more solutions for difficult problems. This is particularly true for planning under the eventuality of failure.

Both extensions we proposed are free in the sense that they have a cost neither in complexity nor in the existence of a controller-generator. We also believe that the expression of the composition of the problem in safety games is still possible but this point will need further investigations. The next natural step is drop perfect information and introduce some form of partial observation. The question being: is it possible to introduce partial observation without getting into the usual exponential blowup.

REFERENCES

Balbiani, P., Cheikh, F., and Feuillade, G. (2008). Composition of interactive web services based on controller synthesis. In *International Workshop on Web Service Composition and Adaptation(WSCA'08)*, pages 521–528, Honolulu, USA. IEEE.

De Giacomo, G., Patrizi, F., and Sardina, S. (2013). Automatic behavior composition synthesis. *Artificial Intelligence Journal*, 196:106–142.

Feuillade, G. and Pinchinat, S. (2007). Modal specifications for the control theory of discrete event systems. *Discrete Event Dynamic Systems*, 17(2):211–232.

Hull, R. (2005). Web services composition: a story of models, automata, and logics. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*, pages xxx–xxi vol.1.

Lomuscio, A., Qu, H., and Raimondi, F. (2009). Mcmas: A model checker for the verification of multi-agent systems. In Bouajjani, A. and Maler, O., editors, *CAV*, volume 5643 of *Lecture Notes in Computer Science*, pages 682–688. Springer.

Muscholl, A. and Walukiewicz, I. (2008). A lower bound on web services composition. In *Proceedings of the international conference on Foundations of Software Science and Computation Structures (FoSSaCS 07)*, pages 274–286. Springer.

Ramadge, P. and Wonham, W. (1989). The control of discrete event systems. *Proceedings of the IEEE*, 77:81–98.

Sardina, S. (2007). Automatic synthesis of new behaviors from a library of available behaviors. In *In Proc. of IJCAI 2007*, pages 1866–1871.