Qualitative approximate behavior composition

Yadav, Nitin; Sardina, Sebastian

Yadav, N., & Sardina, S. (2012). Qualitative approximate behavior composition. Lecture Notes in Computer Science, 7519, 450–462. https://doi.org/10.1007/978-3-642-33353-8_35
Document Version: Accepted Manuscript

Published Version: https://doi.org/10.1007/978-3-642-33353-8_35

Qualitative Approximate Behavior Composition

Nitin Yadav and Sebastian Sardina *

RMIT University, Melbourne, Australia.

Abstract. The behavior composition problem involves automatically building a controller that is able to realize a desired, but unavailable, target system (e.g., a house surveillance) by suitably coordinating a set of available components (e.g., video cameras, blinds, lamps, a vacuum cleaner, phones, etc.) Previous work has almost exclusively aimed at bringing about the desired component in its totality, which is highly unsatisfactory for unsolvable problems. In this work, we develop an approach for *approximate* behavior composition without departing from the classical setting, thus making the problem applicable to a much wider range of cases. Based on the notion of simulation, we characterize what a maximal controller and the "closest" implementable target module (optimal approximation) are, and show how these can be computed using ATL model checking technology for a special case. We show the uniqueness of optimal approximations, and prove their soundness and completeness with respect to their imported controllers.

1 Introduction

The behavior composition problem (e.g., [2, 6, 12, 19]) involves the automatic synthesis of a controller that is able to "realize" (i.e., implement) a desired, though non-existent, complex target system by suitably coordinating a collection of partially controllable available behaviors. A behavior here refers to the abstract operational model of a device or program, generally represented as a non-deterministic transition system. Thus, in a smart building setting, one may look for a controller able to coordinate the execution of a set of devices installed in a house—music and movie players, game consoles, automatic blinds and lights, radios, etc.—such that it appears as if a complex entertainment system was actually being run. A solution to the problem is called a *composition*.

The composition problem is appealing to a wide range of audiences. Indeed, with computers now present in everyday devices like mobile phones, credit cards, or places like homes, offices and factories, the trend is to build embedded complex devices from a collection of simple components. In addition, the problem can be related to several subareas of AI and CS, including web-service composition [10], reactive synthesis [14], agent-oriented programming [18], robot ecologies [15], and automated planning [8].

While the behavior composition problem has been substantially studied in an AI context lately (e.g., [6, 17, 19]), previous work has exclusively aimed at the synthesis of *complete* realisations of the desired target component—compositions that implement the desired component in its totality. This poses a major limitation in problem instances with no (exact) compositions. For such cases, a merely "no solution" outcome is extremely unsatisfactory. The need to address this shortcoming has already been noted in

* We acknowledge the support of the Australian Research Council under grant DP120100332.

previous works [19, 20]. In this paper, we develop a qualitative account of *approximate behavior composition* that caters for instances admitting no exact solutions.

Intuitively, the overarching idea is to *look for those parts of the target module that can be realized with the available modules*, and provide this as an (approximate) solution. More precisely, given a target module, the task is to identify the *closest* alternative target module that can be fully realized with the behaviors at hand—the optimal approximate target. Of course, it is expected that such alternative target will generally provide less functionalities than the original one. Indeed, some execution paths may be impossible to generate with the new target (e.g., it may no more be feasible to play video games when listening to music). Moreover, the alternative target may accommodate less "freedom" of choices in executions (e.g., when requesting to watch a movie, one may now need to commit to whether one will be playing a video game or listening to radio afterwards). Nonetheless, the user can request actions as per the alternative (approximate) target and be guaranteed her requests will always be fulfilled.

Observe that in this paper we assume a setting of *strict* uncertainty, in that the space of possibilities (behaviors' evolutions and target requests) is known, but the probabilities of these potential alternatives cannot be quantified [7]. This contrasts with our previous approach [20], which assumes all such probabilities have been specified for the domain and then looks for the "best" controller possible from a decision-theoretic perspective. Consequently, our account here can be seen as the next natural extension of the "classical" composition framework found in the literature, in that no no additional domain information is assumed. We shall discuss and compare this further in Section 6.

The rest of the paper is organized as follows. In the next two sections, we introduce the composition framework as known in the literature. Besides providing the standard notion for exact compositions (complete solutions to the problem), we also introduce the notion of *maximal compositions*, as controllers that can do as well as any other controller. After that, we develop the main contribution of our work, namely, the notion of *optimal target approximations* as the best alternative target behaviors that can be fully realized in the system at hand. We demonstrate that "importing" controllers from optimal approximations amounts to using maximal controllers (for the original target), thus providing correctness for optimal approximations. In addition, we show that the imported controllers of an optimal approximation together realize the same set of traces as those realized by maximal controllers (together as well), thereby providing a completeness result. More importantly, we prove that optimal approximations are in fact unique (up to simulation equivalence), a very interesting and unexpected property. Finally, we describe how optimal approximate targets can be computed for the special case of deterministic systems (as, for example, in the context of service composition; e.g, [2, 3]) by reducing the problem to ATL model checking, opening the door for advanced model checking tools. We close the paper with a short discussion and conclusions. An extended version of the paper, including proofs, can be found in [21].

2 The Behavior Composition Framework

In a behavior composition setting, a set of *available behaviors* are meant to jointly bring about a *virtual target behavior* [6, 17, 19]. We follow the composition framework in [17] with two minor modifications. For simplicity, we do not deal with the so-called environment, the shared space where behaviors are meant to execute. Nonetheless, all results presented here can be easily generalized to account for an environment. Second, we shall generalize target behaviors to non-deterministic transition systems.

Behaviors A behavior stands for the operational model of a program or device. In general, behaviors provide, step by step, the user a set of actions that it can perform (relative to its specification). At each step, the behavior can be instructed to execute one of the legal actions, causing the behavior to transition to a successor state, and thereby providing a new set of applicable actions.

Formally, a *behavior* is a tuple $\mathcal{B} = \langle B, \mathcal{A}, b_0, \varrho \rangle$, where:[1]

- B is the finite set of behavior's states;
- $\mathcal{A}$ is a set of actions;
- $b_0 \in B$ is the initial state;
- $\varrho \subseteq B \times \mathcal{A} \times B$ is the behavior's transition relation, where $\langle b, a, b' \rangle \in \varrho$, or $b \xrightarrow{a} b'$ in $\mathcal{B}$, denotes that action a executed in behavior state b may lead the behavior to successor state b'.

Note that we allow behaviors to be non-deterministic, that is, given a state and an action, the behavior may transition to more than one state. This implies that one cannot know beforehand what actions will be available to execute after an action is performed, as the next set of applicable actions would depend on the successor state in which the behavior happens to be in. Hence, we say that non-deterministic behaviors are only *partially controllable*. A *deterministic* behavior is one where there is no state $b \in B$ and action $a \in A$ for which there exist two transitions $b \xrightarrow{a} b'$ and $b \xrightarrow{a} b''$ in $\mathcal{B}$ with $b' \neq b''$. A deterministic behavior is *fully controllable*. For the sake of legibility and easier notation, we shall assume, wlog, that behaviors capture non-terminating processes and hence do not have any terminating state with no outgoing transition.[2]

System and Enacted System A system is a collection of behaviors at disposal. Technically, an (available) *system* is a tuple $\mathcal{S} = \langle \mathcal{B}_1, \ldots, \mathcal{B}_n \rangle$, where $\mathcal{B}_i = \langle B_i, \mathcal{A}_i, b_{i0}, \varrho_i \rangle$, for $i \in \{1, \ldots, n\}$, is a behavior, called an *available behavior* in the system.

To refer to the behavior that emerges from the joint execution of behaviors in a system, we use the notion of enacted system behavior. The *enacted system behavior* of an available system $\mathcal{S}$ (as above) is a tuple $\mathcal{E}_\mathcal{S} = \langle S_\mathcal{S}, \mathcal{A}, \{1, \ldots, n\}, s_{\mathcal{S}0}, \delta_\mathcal{S} \rangle$, where:

- $S_\mathcal{S} = B_1 \times \cdots \times B_n$ is the finite set of $\mathcal{E}_\mathcal{S}$'s states; when $s_\mathcal{S} = \langle b_1, \ldots, b_n \rangle$, we denote b_i by $beh_i(s_\mathcal{S})$, for $i \in \{1, \ldots, n\}$;
- $\mathcal{A} = \bigcup_{i=1}^n \mathcal{A}_i$ is the set of actions of $\mathcal{E}_\mathcal{S}$;
- $s_{\mathcal{S}0} \in S_\mathcal{S}$ with $beh_i(s_{\mathcal{S}0}) = b_{i0}$, for $i \in \{1, \ldots, n\}$, is $\mathcal{E}_\mathcal{S}$'s initial state;
- $\delta_\mathcal{S} \subseteq S_\mathcal{S} \times \mathcal{A} \times \{1, \ldots, n\} \times S_\mathcal{S}$ is $\mathcal{E}_\mathcal{S}$'s transition relation, where $\langle s_\mathcal{S}, a, k, s'_\mathcal{S} \rangle \in \delta_\mathcal{S}$, or $s_\mathcal{S} \xrightarrow{a,k} s'_\mathcal{S}$ in $\mathcal{E}_\mathcal{S}$, iff:
  - $beh_k(s_\mathcal{S}) \xrightarrow{a} beh_k(s'_\mathcal{S})$ in $\mathcal{B}_k$; and

[1] With no shared environment in this paper, behaviors are not equipped with guard conditions (as done in [6, 19]) and the set of actions $\mathcal{A}$ are included in their definitions.

[2] As customary, e.g., in LTL verification, this can be easily achieved by introducing "fake" loop transitions.

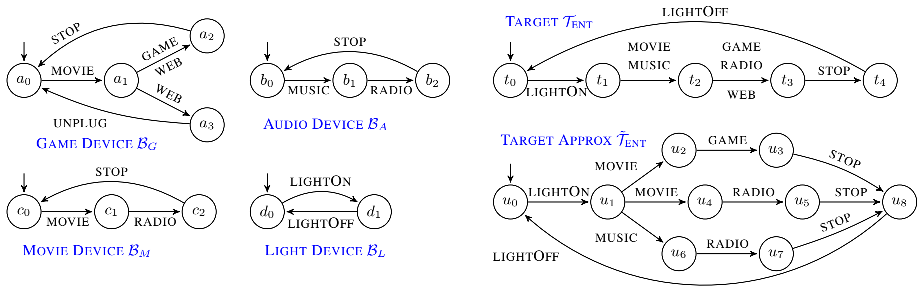

Fig. 1. A smart house scenario with four available behaviors. Target $\mathcal{T}_{\text{ENT}}$ cannot be fully realized in the system, but its optimal approximation $\tilde{\mathcal{T}}_{\text{ENT}}$ can.

- $beh_i(s_{\mathcal{S}}) = beh_i(s'_{\mathcal{S}})$, for $i \in \{1, \ldots, n\} \setminus \{k\}$.

The enacted system behavior $\mathcal{E}_{\mathcal{S}}$ is technically the asynchronous product of the available behaviors. The index k in transitions makes explicit which behavior is performing the action in the transition—all other behaviors remain still.

Target A target behavior $\mathcal{T} = \langle T, \mathcal{A}_T, t_0, \varrho_T \rangle$ is a, possibly *non-deterministic*, behavior that represents the desired functionality to be obtained (through the available system). In contrast with all previous works, we allow for *non-deterministic* target specifications. Nonetheless, the objective is *not* to capture incomplete information, and hence partial controllability, of the target module, but to be able to accommodate action requests carrying more "information." This will come handy for our account of approximation. Thus, in order to preserve the full controllability of the target, we shall consider requests in terms of *target transition*, rather than just actions.

Informally, the behavior composition task is stated as follows: Given a system $\mathcal{S}$ and a target behavior $\mathcal{T}$, is it possible to (partially) control the available behaviors in $\mathcal{S}$ in a step-by-step manner—by instructing them on which action to execute next and observing, afterwards, the outcome in the behavior used—so as to "realize" the desired target behavior. In other words, by adequately controlling the system, it appears as if one was actually executing the target module. (See next section for more details.)

As noted by De Giacomo and Sardina [6], the behavior composition problem is related to planning (under incomplete information) [8], being both synthesis tasks, though here, we look for whom to delegate the next action at each step (whatever such action happens to be at runtime), rather than what those actions should be.

Figure 1 depicts a universal home entertainment system in a smart house scenario. Target $\mathcal{T}_{\text{ENT}}$ encapsulates the desired functionality, which involves first switching on the lights when entering the room, then providing various entertainment options (e.g., listening to music, watching movies, browsing the Web, etc.), and finally stopping active modules and switching off the lights. There are four available devices installed in the house that can be used to bring about such desired behavior, namely, a game device $\mathcal{B}_G$, an audio device $\mathcal{B}_A$, a movie device $\mathcal{B}_M$, and the lights controller $\mathcal{B}_L$. Note that action WEB in the device $\mathcal{B}_G$ is non-deterministic, as it may bring the module into states a_2 or a_3. If the device happens to evolve to state a_3, then, for some reason, it is not enough to stop the device to reset it: the device needs to be completely unplugged.

3 Controllers and Compositions

Next, we formally define what constitutes a solution for a behavior composition problem. In doing so, we shall not only look at the problem from a binary perspective — solvable vs unsolvable–but instead provide a *qualitative* account of "optimal" solutions. From now on, let $\mathcal{S} = \langle \mathcal{B}_1, \ldots, \mathcal{B}_n \rangle$ be an available system and $\mathcal{T} = \langle T, \mathcal{A}, t_0, \varrho_T \rangle$ be a target behavior to be realized on $\mathcal{S}$.

Controller A controller is a component able to activate, stop, and resume any of the available behaviors, and to instruct them to execute an (allowed) action. The controller has *full observability* on the available behaviors; that is, it can keep track (at runtime) of their current states—if details have to be hidden, this can be done by means of nondeterminism within the abstract behaviors exposed.

To formally define controllers and solutions, we rely on the notions of traces and histories. A *trace* for a given enacted system $\mathcal{E}_\mathcal{S} = \langle S_\mathcal{S}, \mathcal{A}, \{1, \ldots, n\}, s_{\mathcal{S}0}, \delta_\mathcal{S} \rangle$ is a, possibly infinite, sequence of the form $s^0 \xrightarrow{a^1,k^1} s^1 \xrightarrow{a^2,k^2} \cdots$ such that *(i)* $s^0 = s_{\mathcal{S}0}$; and *(ii)* $s^j \xrightarrow{a^{j+1},k^{j+1}} s^{j+1}$ in $\mathcal{E}_\mathcal{S}$, for all $j > 0$. A *history* is just a finite prefix $h = s^0 \xrightarrow{a^1,k^1} \cdots \xrightarrow{a^\ell,k^\ell} s^\ell$ of a trace. We denote s^ℓ by *last*(h), ℓ by $|h|$ (i.e., the length of h), and sequence $a^1 \cdot \ldots \cdot a^\ell$ as $[h]$ (i.e., the projection on actions). Traces and histories can also be defined for a behavior $\mathcal{B}$ in a similar fashion: behavior traces have the form $s^0 \xrightarrow{a^1} s^1 \xrightarrow{a^2} \cdots$ such that *(i)* $s^0 = b_0$; and *(ii)* $s^j \xrightarrow{a^{j+1}} s^{j+1}$ in $\mathcal{B}$, for all $j > 0$. We use $\mathcal{H}_\mathcal{S}$ and $\mathcal{H}_\mathcal{B}$ to denote the set of system histories (i.e., histories of $\mathcal{E}_\mathcal{S}$) and histories of behavior $\mathcal{B}$, respectively.

A *controller* for target $\mathcal{T}$ on system $\mathcal{S}$ is a partial function $C : \mathcal{H}_\mathcal{S} \times (T \times \mathcal{A} \times T) \mapsto \{1, \ldots, n\}$, which, given a system history $h \in \mathcal{H}_\mathcal{S}$ and a requested target transition $\langle t, a, t' \rangle \in \varrho_T$, returns the index of an available behavior to which the action a is delegated for execution. For legibility, we shall write $C(h, t_1 \xrightarrow{a} t_2)$ to compactly denote $C(h, t_1, a, t_2)$. Note here the slight departure form previous notions of controllers (e.g., [6, 17, 19]), in that a controller now receives a complete target transition as the next request, not just an action. While this has no impact when dealing with deterministic targets, it guarantees full controllability for nondeterministic ones.

Intuitively, a controller (fully) realizes a target behavior if for every trace (i.e., run) of the target, at every step, the controller returns the index of an available behavior that can perform the requested action. Formally, one first defines when a controller C *realizes a trace* of the target $\mathcal{T}$. Though not required for this paper, the reader is referred to [6, 17] for details on how to formally characterize trace realization. We denote $\Delta^C_{\langle \mathcal{S}, \mathcal{T} \rangle}$ the set of traces of $\mathcal{T}$ that controller C is able to realize in system $\mathcal{S}$. Then, a controller C realizes the target behavior $\mathcal{T}$ *iff* it realizes all its traces. In that case, C is said to be an *exact composition* for target $\mathcal{T}$ on system $\mathcal{S}$.

Now, suppose we are given a target behavior $\mathcal{T}$ and an available system $\mathcal{S}$, and that, as expected in many domains, there is no exact composition for $\mathcal{T}$ on $\mathcal{S}$—the target cannot be *completely* realized in the system. This is indeed the case in our example, as there is no exact composition for $\mathcal{T}_{\text{ENT}}$ in the house system. Merely returning a negative "no solution" outcome is highly unsatisfactory. The question then is: what does it mean for a controller C_1 to achieve "a better realization" of $\mathcal{T}$ on $\mathcal{S}$ than controller C_2?

To answer such a question in a qualitative manner, we rely on the extent at which controllers are able to honour arbitrary long set of target requests. We say that controller C_1 *dominates* controller C_2, denoted $C_1 \geq C_2$, *iff* $\Delta^{C_2}_{\langle \mathcal{S},\mathcal{T}\rangle} \subseteq \Delta^{C_1}_{\langle \mathcal{S},\mathcal{T}\rangle}$—$C_1$ can honour all request sequences that C_2 can honour, and possibly more. As usual, $C_1 > C_2$ is equivalent to $C_1 \geq C_2$ but $C_2 \not\geq C_1$, that is, $\Delta^{C_2}_{\langle \mathcal{S},\mathcal{T}\rangle} \subset \Delta^{C_1}_{\langle \mathcal{S},\mathcal{T}\rangle}$. A controller C is said to be a *maximal composition* (for a target on a system) *iff* for every other controller C', if $C' \geq C$, then $C \geq C'$ (or equivalently $C' \not> C$). In other words, maximal compositions are those for which there is no other controller that can realize strictly more runs of the target behavior in the system. We use $\textsc{MaxComp}(\mathcal{S},\mathcal{T})$ to denote the set of all maximal compositions for target $\mathcal{T}$ on system $\mathcal{S}$.

Consider the following two controllers for our smart house. Whereas controller C_1 allocates all requests to the light device $\mathcal{B}_L$, controller C_2 delegates media and light requests to the audio $\mathcal{B}_A$ and light $\mathcal{B}_L$ devices, respectively. Then, C_1 realizes just one target trace, that is, $\Delta^{C_1}_{\langle \mathcal{S},\mathcal{T}\rangle} = \{t_0 \xrightarrow{\text{LIGHTON}} t_1\}$. On the other hand, C_2 realizes such a trace as well as trace $t_0 \xrightarrow{\text{LIGHTON}} t_1 \xrightarrow{\text{MOVIE}} t_2 \xrightarrow{\text{RADIO}} t_3 \xrightarrow{\text{STOP}} t_4$ (and all its prefixes). Therefore, $\Delta^{C_1}_{\langle \mathcal{S},\mathcal{T}\rangle} \subset \Delta^{C_2}_{\langle \mathcal{S},\mathcal{T}\rangle}$ and $C_2 > C_1$ holds. The reader may notice that even better controllers than C_2 exist when all four behaviors are used.

As expected, whenever a behavior composition problem admits an exact composition—the target is fully realizable—the set of exact compositions coincides with that of maximal compositions. When full realizations are impossible, though, maximal compositions capture the best controllers that one could hope for.

4 Target Approximation

Whereas maximal compositions, as defined above, provide a way of handling instances with no exact solution, they do not convey useful insights on how well such instances can be solved. Even if we are given the set of traces that a maximal composition realizes, it will be difficult to reconstruct what it means in terms of the problem specification. As a consequence, using a maximal non-exact composition may yield dead-end executions where no further actions can be honoured. What is more, while there are various techniques to construct exact compositions (e.g., [6, 16, 19]), it is far from clear how to build maximal composition controllers.

So, in this section, we will look at "approximation" from a different perspective that is arguably more intuitive and computationally more amenable than dealing with controller functions, namely, we are concerned with what parts of the target can in fact be brought about. More concretely, we are interested in the following task:

> *Given an available system $\mathcal{S}$ and a target behavior $\mathcal{T}$, find an (approximate) target behavior $\tilde{\mathcal{T}}$ that can be fully realized on $\mathcal{S}$ (by some controller $C_{\tilde{\mathcal{T}}}$) and such that $\tilde{\mathcal{T}}$ is "as close as possible" to the original target behavior $\mathcal{T}$.*

We call this the *approximate behavior composition problem*. Once an approximate target $\tilde{\mathcal{T}}$ is obtained, one may either use such new target directly or consider "importing" its exact compositions into the original target module $\mathcal{T}$. Hopefully, in the latter case, the imported controllers will turn out to be the best possible controllers for the original target. These are arguably the main ideas of our work and what we shall develop below. Before doing so, we should point out that defining approximate targets based merely on trace/language inclusion is not sufficient. While two targets may yield exactly the same sequences of requests, one may accept an exact composition while the other may not. In our smart house scenario, for instance, the two sequences LIGHTON · MOVIE · GAME · STOP and LIGHTON · MOVIE · RADIO · STOP may be realized by the same controller for the approximation $\tilde{\mathcal{T}}_{\mathrm{ENT}}$, but not for the original target $\mathcal{T}_{\mathrm{ENT}}$.

In order to capture approximate targets, we make use of the formal notion of *simulation* [13]. A simulation relation captures the similarity in the behavior of two transition systems. Intuitively, a (transition) system S_1 "simulates" another system S_2 if S_1 is able to *match* all of S_2's moves. We make this precise for our (target) behaviors as follows. Let $\mathcal{T}_i = \langle T_i, \mathcal{A}, t_{i0}, \varrho_i \rangle$, where $i \in \{1, 2\}$, be two target behaviors. A *simulation relation* of $\mathcal{T}_2$ by $\mathcal{T}_1$ is a relation *Sim* $\subseteq T_2 \times T_1$ such that $\langle t_2, t_1 \rangle \in$ *Sim* implies that for every transition $\langle t_2, a, t_2' \rangle \in \varrho_2$ in $\mathcal{T}_2$, there exists a transition $\langle t_1, a, t_1' \rangle \in \varrho_1$ in $\mathcal{T}_1$ such that $\langle t_2', t_1' \rangle \in$ *Sim*. We say that a state $t_2 \in T_2$ is *simulated* by a state $t_1 \in T_1$ (or t_1 simulates t_2), denoted $t_2 \preceq t_1$, *iff* there exists a simulation relation *Sim* of T_2 by T_1 such that $\langle t_2, t_1 \rangle \in$ *Sim*. Observe that relation $\preceq$ is itself a simulation relation (of $\mathcal{T}_2$ by $\mathcal{T}_1$), and in fact, it is the largest simulation relation, in that all simulation relations are contained in it. Informally, $t_2 \preceq t_1$ means that t_1 in $\mathcal{T}_1$ can "mimic" all moves of t_2 in $\mathcal{T}_2$, and that this property is propagated in their corresponding successor states. We say that a target behavior $\mathcal{T}_1$ *simulates* target behavior $\mathcal{T}_2$, denoted $\mathcal{T}_2 \preceq \mathcal{T}_1$, if it is the case that $t_{20} \preceq t_{10}$, that is, their initial states are in simulation and, as a result, $\mathcal{T}_1$ can always mimic $\mathcal{T}_2$ from the start. In our example, t_2 and t_1 in $\mathcal{T}_{\mathrm{ENT}}$ simulate states u_4 and u_1, respectively, in $\tilde{\mathcal{T}}_{\mathrm{ENT}}$ (i.e., $u_4 \preceq t_2$ and $u_1 \preceq t_1$), but not the other way around (i.e., $t_2 \not\preceq u_4$ and $t_1 \not\preceq u_1$). Two targets are said to be *simulation equivalent*, denoted $\mathcal{T}_1 \sim \mathcal{T}_2$, whenever they simulate each other.

We then argue that a qualitative comparison of target approximations can be achieved based on their simulation "hierarchy" (see that $\preceq$ is a pre-order). We say that a target behavior $\tilde{\mathcal{T}}$ *approximates* target $\mathcal{T}$ on system $\mathcal{S}$ (or $\tilde{\mathcal{T}}$ is an approximation of $\mathcal{T}$ on $\mathcal{S}$) *iff* $\tilde{\mathcal{T}} \preceq \mathcal{T}$ and there is an exact composition for $\tilde{\mathcal{T}}$ on $\mathcal{S}$ (i.e., $\tilde{\mathcal{T}}$ is simulated by $\mathcal{T}$ and it can be fully realized on available system $\mathcal{S}$).

Despite being fully solvable, an approximation will generally provide "*less*" than the original target. First, an approximation may be missing certain executions altogether. In the smart house scenario, approximation $\tilde{\mathcal{T}}_{\mathrm{ENT}}$ does not account for the action sequence LIGHTON · MUSIC · GAME · STOP · LIGHTOFF. Second, an approximation may require the user to commit earlier to future possible request choices. In that sense, a user of target $\tilde{\mathcal{T}}_{\mathrm{ENT}}$ needs to decide when requesting MOVIE in state u_1 if she will later play a GAME or listen to RADIO. Notice such extra "temporal" information is not required at state t_1 in original target $\mathcal{T}_{\mathrm{ENT}}$. It is exactly to accommodate this feature that we have departed from the standard view of deterministic targets.

Of course, between full realization and the trivial empty approximation, there lies a whole spectrum of approximating targets. Among these, we are interested in those that are "closest" to the original target, in that the minimum possible is given up. We say that a target behavior $\tilde{\mathcal{T}}$ is an *optimal approximate* of target $\mathcal{T}$ on system $\mathcal{S}$ *iff*:

1. $\tilde{\mathcal{T}}$ is an approximation of $\mathcal{T}$ on $\mathcal{S}$; and

2. there is no target behavior $\tilde{\mathcal{T}}'$ that approximates $\mathcal{T}$ on $\mathcal{S}$ such that $\tilde{\mathcal{T}} \prec \tilde{\mathcal{T}}'$, that is, $\mathcal{T}$ cannot be approximated by a strictly more general target module.

Intuitively, an optimal target approximation is a maximal representation of those aspects of the original target that can be completely implemented. When the target behavior does admit a full realization in the system, the optimal approximation is then expected to represent the target module in all its extent.

Theorem 1. *Suppose there is an exact composition for target $\mathcal{T}$ on system $\mathcal{S}$. Then, $\tilde{\mathcal{T}}$ is an optimal approximation of $\mathcal{T}$ on $\mathcal{S}$ iff $\tilde{\mathcal{T}} \sim \mathcal{T}$.*

Importantly, there can only be one way of optimally approximating a given target.

Theorem 2. *An optimal approximation $\tilde{\mathcal{T}}$ of a target $\mathcal{T}$ on a system $\mathcal{S}$ is unique upto simulation equivalence.*

We observe that, for non-deterministic transition systems, simulation is a stronger measure of equivalence than language inclusion [9]. Therefore, if a target $\tilde{\mathcal{T}}$ approximates another target $\mathcal{T}$, then the action request sequences resulting from the traces of $\tilde{\mathcal{T}}$ will be a subset of those produced by $\mathcal{T}$. It follows then that if $C_{\tilde{\mathcal{T}}}$ is an exact composition for $\tilde{\mathcal{T}}$, then $C_{\tilde{\mathcal{T}}}$ ought to be able to handle a subset of $\mathcal{T}$'s request sequences.

4.1 Imported Controllers

In contrast with maximal controllers, optimal approximations are specified in the *same* language as the original problem. The user can thus decide to request actions as per the new (approximate) target with guaranteed full realizability. Nonetheless, one may still ask in which sense these solutions are "correct." To answer that, we show that using an exact composition for an optimal approximation amounts to using a maximal composition for the original target. To that end, we define what it means to "import" a controller $C_{\mathcal{T}'}$ designed for one target module $\mathcal{T}'$ into another target module $\mathcal{T}$.

We start by defining the family of functions that are meant to explain sequences of action requests in a target. Informally, the function $\textsc{Expl}_{\mathcal{T}}(\sigma)$ outputs a history of the target $\mathcal{T}$ compatible with the given sequence of actions σ. Formally, a function $\textsc{Expl}_{\mathcal{T}} : \mathcal{A}^* \mapsto \mathcal{H}_{\mathcal{T}}$ is a <u>*target explanatory*</u> function for a target $\mathcal{T}$ if for any action sequence $\sigma = a^1 \cdot \ldots \cdot a^\ell \in \mathcal{A}^*$, with $\ell \geq 0$, it is the case that $\textsc{Expl}_{\mathcal{T}}(\sigma) = t^0 \xrightarrow{a^1} \cdots \xrightarrow{a^\ell} t^\ell \in \mathcal{H}_{\mathcal{T}}$. In general, there will be many of such functions, since the same sequence of action requests can arise from different runs of a non-deterministic target. For instance, sequence $\textsc{lightOn} \cdot \textsc{movie}$ can be explained in two ways on target $\tilde{\mathcal{T}}_{\text{ENT}}$, namely, via histories $u_0 \xrightarrow{\textsc{lightOn}} u_1 \xrightarrow{\textsc{movie}} u_2$ and $u_0 \xrightarrow{\textsc{lightOn}} u_1 \xrightarrow{\textsc{movie}} u_4$.

Using target explanatory functions, we next characterize the set of so-called *induced* controllers. Suppose we have a controller $C_{\mathcal{T}'}$ for a target $\mathcal{T}'$ (on a system $\mathcal{S}$). An induced controller (from controller $\mathcal{C}_{\mathcal{T}'}$) for a target behavior $\mathcal{T}$ is one that handles requests from $\mathcal{T}$ as if they were requests issued as per module $\mathcal{T}'$. Recall that a controller for a system $\mathcal{S}$ outputs the behavior index to which a given transition-action request is delegated to at a certain system history. Formally, then, we say that $C_{\mathcal{T}}^{\mathcal{T}'}$ is an <u>*induced controller*</u> (from controller $C_{\mathcal{T}'}$ on target $\mathcal{T}'$) for target $\mathcal{T}$ over system $\mathcal{S}$

if there exists a target explanatory function $\textsc{Expl}_{\mathcal{T}'}(\cdot)$ for $\mathcal{T}'$ such that for every system history $h \in \mathcal{H}_{\mathcal{S}}$ and transition $t_1 \xrightarrow{a} t_2$ in $\mathcal{T}$, the following holds (recall that $[h]$ denotes the sequence of actions in history h):

$$C_{\mathcal{T}}^{\mathcal{T}'}(h, t_1 \xrightarrow{a} t_2) = \begin{cases} \mathcal{C}_{\mathcal{T}'}(h, t_1' \xrightarrow{a} t_2') & \textsc{Expl}_{\mathcal{T}'}([h] \cdot a) = t^0 \xrightarrow{a^1} \cdots \xrightarrow{a^{|h|}} t_1' \xrightarrow{a} t_2' \\ \text{undefined} & \textsc{Expl}_{\mathcal{T}'}([h] \cdot a) \text{ is undefined} \end{cases}$$

That is, $\mathcal{T}$'s request $t_1 \xrightarrow{a} t_2$ is delegated at history h as controller $C_{\mathcal{T}'}$ would delegate request $t_1' \xrightarrow{a} t_2'$ from target $\mathcal{T}'$ if h's requests leave target $\mathcal{T}'$ in state t_1' and the current requested action a is indeed explained by transition request $t_1' \xrightarrow{a} t_2'$ in $\mathcal{T}'$. When there is no explanation in the $\mathcal{T}'$—$\textsc{Expl}(\cdot)$ is undefined—the induced controller is left undefined. Note that different ways of explaining original target's sequences of requests (i.e., different explanatory functions) yield different induced controllers.

Finally, an *imported* controller is a maximal (i.e., non-strictly dominated) controller within the family of induced controllers—the "best" induced controllers. Technically, the set of *imported controllers* from C on $\mathcal{T}$ into target $\mathcal{T}'$, denoted $\Omega_{\langle C,\mathcal{T}\rangle}^{\mathcal{T}'}$ is the set of all controllers $\hat{C}$ for $\mathcal{T}'$ such that *(i)* $\hat{C}$ is an induced controller from C on target $\mathcal{T}$ for $\mathcal{T}'$; and *(ii)* there is no other induced controller C' such that $C' > \hat{C}$.

First, we show that better target approximations amount to better, or more precisely "never worse," imported controllers.

Theorem 3. *Let $\tilde{\mathcal{T}}_1$ and $\tilde{\mathcal{T}}_2$ be two target approximations of target $\mathcal{T}$ on system $\mathcal{S}$, and let $\tilde{C}_1$ and $\tilde{C}_2$ be exact compositions of $\tilde{\mathcal{T}}_1$ and $\tilde{\mathcal{T}}_2$, resp. Suppose also that $\tilde{\mathcal{T}}_2 \preceq \tilde{\mathcal{T}}_1$ (i.e, $\tilde{\mathcal{T}}_1$ simulates $\tilde{\mathcal{T}}_2$). Then, for every controller $C_1 \in \Omega_{\langle \tilde{C}_1,\tilde{\mathcal{T}}_1\rangle}^{\mathcal{T}}$, there is no controller $C_2 \in \Omega_{\langle \tilde{C}_2,\tilde{\mathcal{T}}_2\rangle}^{\mathcal{T}}$ such that $C_2 > C_1$ holds.*

In other words, if $\tilde{\mathcal{T}}_1$ is as good an approximation as $\tilde{\mathcal{T}}_2$, then $\tilde{\mathcal{T}}_1$'s imported controllers will not be worse than those imported from $\tilde{\mathcal{T}}_2$. More importantly, the next result demonstrates that importing controllers from an *optimal* approximation yields maximal compositions (for the original target being approximated), and that, together, they account for every trace of the original target that could ever be realized. In other words, $\Omega_{\langle \tilde{C},\tilde{\mathcal{T}}\rangle}^{\mathcal{T}}$ is sound and "complete."

Theorem 4. *Let $\tilde{\mathcal{T}}$ be an optimal approximation of target $\mathcal{T}$ on system $\mathcal{S}$, and $\tilde{C}$ be an exact composition for $\tilde{\mathcal{T}}$. Then,*

- *For all $C \in \Omega_{\langle \tilde{C},\tilde{\mathcal{T}}\rangle}^{\mathcal{T}}$, it holds that $C \in \textsc{MaxComp}(\mathcal{S},\mathcal{T})$; and*
- *$\bigcup_{C \in \Omega_{\langle \tilde{C},\tilde{\mathcal{T}}\rangle}^{\mathcal{T}}} \Delta_{\langle \mathcal{S},\mathcal{T}\rangle}^{C} = \bigcup_{C \in \textsc{MaxComp}(\mathcal{S},\mathcal{T})} \Delta_{\langle \mathcal{S},\mathcal{T}\rangle}^{C}$, that is, all imported controllers account together for all realizable target traces.*

These two results are important in that they establish the relationship between approximating the target and optimizing its controller: optimizing targets implies optimizing controllers. A direct and expected consequence of Theorems 1 and 4 is that if the optimal approximation is simulation equivalent to the target, then every imported controller from such approximation is in fact an exact composition.

5 Computing Optimal Approximations for Deterministic Systems

Various techniques have been used to actually solve classical behavior composition problems, including PDL satisfiability [6], direct search-based approaches [19], LTL/ATL synthesis [5, 16], and computation of special kind of simulation relations [3, 17]. Unfortunately, all those techniques synthesize *exact* composition controllers. In the context of our work, we are interested in *computing optimal target approximations* instead. We show how this can be effectively done for the special case of *deterministic* available behaviors, as in the case of service composition [2, 3].

De Giacomo and Felli [5] has shown that the controller generator (i.e., a structure representing all exact compositions) can be synthesised by resorting to Alternating-time Temporal Logic (ATL) model checking. ATL [1] is a logic for reasoning about the ability of group of agents (i.e., coalitions) in multi-agent game structures. The advantages of reducing the composition problem to that of ATL reasoning is that it provides access to some of the most advanced model checking techniques and tools, such as MCMAS [11], that are in active development within the agent community.

ATL formulae are built by combining propositional formulas, the usual temporal operators—namely, $\bigcirc$ ("in the next state"), $\Box$ ("always"), $\Diamond$ ("eventually"), and $\mathcal{U}$ ("strict until")—and a *coalition path quantifier* $\langle\langle A \rangle\rangle$ taking a set of agents A as parameter. Intuitively, an ATL formula $\langle\langle A \rangle\rangle\phi$, where A is a set of agents, holds in an ATL structure if by suitably choosing their moves, the agents in A *can force* ϕ *true*, no matter how other agents happen to move. The semantics of ATL is defined in so-called *concurrent game structures* where, at each point, all agents simultaneously choose their moves from a finite set, and the next state deterministically depends on such choices.

In order to reduce a behavior composition problem to an ATL model checking problem, De Giacomo and Felli [5] basically define an ATL structure $\mathcal{M}_{\mathcal{S},\mathcal{T}}$ with one agent per available and target behavior, and one distinguished agent *contr* representing the controller. A state $\langle b_1, \ldots, b_n, t_s, a, t_d, k \rangle$ in such a model encodes the current state b_i of each available behavior, the current state t_s of the target, the current action a being requested by the target, the next target state t_d given the request, and the index of the available behavior to which the last action was delegated to. The initial states of $\mathcal{M}_{\mathcal{S},\mathcal{T}}$ encode all possible initial configurations of the composition framework—initial states for all behaviors and a legal initial request. Also, the structure is made to encode all legal evolutions of the composition instance. The task then involves model checking the special formula $\varphi = \langle\langle \mathit{contr} \rangle\rangle \Box (\bigwedge_{i=1,\ldots,n} \mathit{state}_i \neq \mathit{error}_i)$ (against structure $\mathcal{M}_{\mathcal{S},\mathcal{T}}$),[3] which states that the controller agent has a strategy so that none of the n available behaviors end up in an error state. A behavior arrives to a distinguished "error"state if it is ever delegated an action that it cannot perform. As a result, the controller agent ought to make sure it always delegates actions in the right way so as to satisfy every potential request, that is, it has to solve the composition problem. Finally, De Giacomo and Felli [5, Definition 2 & Theorems 3 and 4] show how to extract a correct controller generator—a structure representing all exact compositions—from the set of *winning states* $[\varphi]_{\mathcal{M}_{\mathcal{S},\mathcal{T}}}$, namely, all those states q in $\mathcal{M}_{\mathcal{S},\mathcal{T}}$ such that $q \models \varphi$. Intuitively, a winning state for

[3] We note that [5] deals with final states where the composition execution may stop. For simplicity, we have not dealt with final configurations here, but one can easily accommodate them.

them is one in which the current request is legally honored to some available behavior and *all* corresponding successor states are winning.

Surprisingly, it turns out that one can readily adapt De Giacomo and Felli's reduction to actually synthesize an optimal approximation for a, possibly non-solvable, *deterministic* composition problem (and to extract the corresponding controller generator). Though it looks counter-intuitive, the key for this is to *include the target behavior in the coalition* so that the joint-strategy also includes selecting which transition from the actual target may be requested. In other words, we are instead to model check the following formula against structure $\mathcal{M}_{\mathcal{S},\mathcal{T}}$:

$$\tilde{\varphi} = \langle\langle \mathsf{contr}, \mathsf{tgt} \rangle\rangle \Box (\bigwedge_{i=1,\ldots,n} \mathit{state}_i \neq \mathit{error}_i).$$

In this case, a winning state in $[\tilde{\varphi}]_{\mathcal{M}_{\mathcal{S},\mathcal{T}}}$ is one in which the target requests actions such that the controller can (always) legally honor them to an available behavior, and has *some* corresponding successor winning state. Observe here the implicit existential quantification on the requests, as compared with the universal quantification implied in De Giacomo and Felli [5]'s encoding for exact composition synthesis.

Intuitively, the idea behind formula $\tilde{\varphi}$, as opposed to formula φ, is that the coalition is now in control of what can be requested (and what should not be). This suggests that the coalition has the ability to select which parts of the target can be executed without driving the available system into an "error" state (due to an impossible fulfilment of a request). It follows then that one can extract an *optimal* approximation from the *maximal* winning set $[\tilde{\varphi}]_{\mathcal{M}_{\mathcal{S},\mathcal{T}}}$, as the following result demonstrates.

Theorem 5. *Let $\mathcal{S} = \langle \mathcal{B}_1, \ldots, \mathcal{B}_n \rangle$ be a system and $\mathcal{T} = \langle T, \mathcal{A}, t_0, \varrho_T \rangle$ a target module. Then, behavior $\hat{\mathcal{T}} = \langle \hat{T}, \mathcal{A}, \hat{t_0}, \hat{\varrho} \rangle$ is an optimal approximation for $\mathcal{T}$ on $\mathcal{S}$, where:*

- $\hat{T} = \{\langle b_1, \ldots, b_n, t_s \rangle \mid \langle b_1, \ldots, b_n, t_s, a, t_d, k \rangle \in [\tilde{\varphi}]_{\mathcal{M}_{\mathcal{S},\mathcal{T}}}\} \cup \{\hat{t_0}\}$*;*
- $\hat{t_0} = \langle b_{10}, \ldots, b_{n0}, t_0 \rangle$ *is the initial state of* $\hat{\mathcal{T}}$*;*
- $\hat{\varrho}(\langle b_1, \ldots, b_n, t_s \rangle, a, \langle b'_1, \ldots, b'_n, t_d \rangle)$ iff *for some action $a' \in \mathcal{A}$, and indexes $k, k' \in \{1, \ldots, n\}$, it is the case that:*
  - $\langle b_1, \ldots, b_n, t_s, a, t_d, k \rangle, \langle b'_1, \ldots, b'_n, t'_s, a', t'_d, k' \rangle \in [\tilde{\varphi}]_{\mathcal{M}_{\mathcal{S},\mathcal{T}}}$*; and*
  - $\langle b_1, \ldots, b_n, t_s, a, t_d, k \rangle$ *may transition to* $\langle b'_1, \ldots, b'_n, t'_s, a', t'_d, k' \rangle$ *in* $\mathcal{M}_{\mathcal{S},\mathcal{T}}$*.*

It is not hard to see that the controller generator [17] for $\hat{\mathcal{T}}$ can be extracted by keeping those behavior delegations that transition a winning game state into another winning state in $\mathcal{M}_{\mathcal{S},\mathcal{T}}$. In terms of computational complexity, the model checking task on ATL can be done in polynomial time wrt to the size of the game structure [1]. Since the size of such space is exponential on the number of available behaviors, computing the optimal approximation can be done in exponential time (for deterministic systems). Observe that, in the worst case, the approximation problem itself is (at least) exponential, as it subsumes the classical behavior composition problem (which is known to be EXPTIME-complete even under deterministic behaviors). Indeed, in order to check if a problem has an exact composition one can compute its optimal approximation and test (in polynomial time) if it is simulation equivalent with the original target.

The full details of the ATL encoding, together with an implementation in MCMAS of our running example, can be found in [21].

6 Discussion

We have proposed a qualitative framework for approximate behavior composition in which the task is to find the *closest* possible target module that can be implemented with the available modules. To that end, we relied on the formal notion of simulation and that of imported controllers for the specification of the problem, and on ATL model checking for actual computation of solutions for the special case of deterministic systems. To our knowledge, this is the first account that is able to accommodate behavior composition instances with no complete solutions—arguably the most common ones—while still remaining within the original problem formulation.

Initially, the work of Girard and Pappas [9] appeared to be extremely related to our objectives, as it proposes a notion of transition system approximation based on the notion of simulation. However, their work differs in *what* is being approximated. In the most general notion of simulation, only some aspects of states are observable and two states in simulation are meant to coincide on their observable aspects. In Girard and Pappas's account, an approximate transition system is allowed to differ on such observables up to some extent: s simulates s' implies s can (always) replicate all moves of s' and s's observation is "similar" to that of s'. It follows then that the approximating transition system *must* still be able to mimic *all* actions of the approximated system. In our framework, there is no notion of state observations (every state has the same observations) and hence we only focus on the similarities of states in terms of the potential behavior they can generate. We believe though that one can use their account of approximation when performing composition *within a shared environment* (as in [6, 19]), so as to allow the environment to evolve "close enough" to what is necessary.

Confronted with a behavior composition problem instance admitting no complete solution (i.e., no exact composition) one can, of course, think of other approaches orthogonal to the one developed here. For example, one may look for additional available behavior modules or enhancement of existing ones with new capabilities that will recover exactness. In some cases, simply adding extra "copies" of existing modules could be enough. Thus, installing an extra video camera in the house may turn the problem solvable. One could also consider a framework where essential and optional functionalities can be specified, and look for controllers that fully realize the former ones while optimizing the latter ones. We shall focus on these ideas on future work, as well as on generalizing the actual synthesis techniques from Section 5 to nondeterministic systems, possibly relying on more expressive games using GR(1) formulas [4].

The only approach, as far as we know, to deal with unsolvable composition instances is the one we pursued previously in [20] within a decision-theoretic framework. There, the idea is to look for a controller that maximizes the *"expected realizability"* of the target behavior. There are however two major differences with our current proposal. First, their controller may in some runs yield dead-end situations, that is, states from where no further target request can be fulfilled. Under our framework, the user (of the target) can never arrive to those "error" situations, as the optimal approximation is always fully implementable. Second, in our work we kept the strict uncertainty setting from the composition problem found in the literature—no extra knowledge of the domain is assumed to be available. We note that it is well known that *strict* uncertainty cannot always be reduced to a setting where the uncertainty can be measured [7]. Nonetheless, it would be interesting to be able to accommodate extra domain knowledge *when available*.

References

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, (49):672–713, 2002.

[2] P. Balbiani, F. Cheikh, and G. Feuillade. Composition of interactive web services based on controller synthesis. In *Proc. of SERVICES*, pages 521–528, 2008.

[3] D. Berardi, F. Cheikh, G. De Giacomo, and F. Patrizi. Automatic service composition via simulation. *International Journal of Foundations of Computer Science*, 19(2):429–452, 2008.

[4] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Saar. Synthesis of reactive(1) designs. *Journal of Computer and System Sciences*, pages 1–28, 2011.

[5] G. De Giacomo and P. Felli. Agent composition synthesis based on ATL. In *Proc. of AAMAS*, pages 499–506, 2010.

[6] G. De Giacomo and S. Sardina. Automatic synthesis of new behaviors from a library of available behaviors. In *Proc. of IJCAI*, pages 1866–1871, 2007.

[7] S. French. *Decision Theory: An Introduction to the Mathematics of Rationality*. Ellis Horwood, 1986.

[8] M. Ghallab, D. S. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers Inc., 2004.

[9] A. Girard and G. Pappas. Approximation metrics for discrete and continuous systems. *Automatic Control, IEEE Transactions on*, 52(5):782–798, 2007.

[10] R. Hull. Web services composition: A story of models, automata, and logics. In *Proc. of SCC*, pages 18–19, 2005.

[11] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In *Proc. of CAV*, pages 682–688, 2009.

[12] Y. Lustig and M. Y. Vardi. Synthesis from component libraries. In *Proc. of FOSSACS*, volume 5504 of *LNCS*, pages 395–409, 2009.

[13] R. Milner. An algebraic definition of simulation between programs. In *Proc. of IJCAI*, pages 481–489, 1971.

[14] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. of POPL*, pages 179–190, 1989.

[15] A. Saffiotti and M. Broxvall. PEIS ecologies: Ambient intelligence meets autonomous robotics. In *Proc. of the International Conference on Smart Objects and Ambient Intelligence*, pages 275–280, 2005.

[16] S. Sardina and G. De Giacomo. Realizing multiple autonomous agents through scheduling of shared devices. In *Proc. of ICAPS*, pages 304–312, 2008.

[17] S. Sardina, F. Patrizi, and G. De Giacomo. Behavior composition in the presence of failure. In *Proc. of KR*, pages 640–650, 2008.

[18] Y. Shoham. Agent-oriented programming. *Artificial Intelligence Journal*, 60:51–92, 1993.

[19] T. Stroeder and M. Pagnucco. Realising deterministic behaviour from multiple non-deterministic behaviours. In *Proc. of IJCAI*, pages 936–941, 2009.

[20] N. Yadav and S. Sardina. Decision theoretic behavior composition. In *Proc. of AAMAS*, pages 575–582, 2011.

[21] N. Yadav and S. Sardina. Qualitative approximate behavior composition. Available from CoRR `http://arxiv.org/abs/1207.3863`, 2012.