Hierarchical planning in a supervisory control context with compositional abstraction

Juliana Vilela¹ · Richard Hill²

Received: 8 December 2020 / Accepted: 23 August 2021 / Published online: 14 September 2021
© The Author(s) 2021

Abstract

Hierarchy is a tool that has been applied to improve the scalability of solving planning problems modeled using Supervisory Control Theory. In the work of Hill and Lafortune (2016), the notion of *cost equivalence* was employed to generate an abstraction of the supervisor that, with additional conditions, guarantees that an optimal plan generated on the abstraction is also optimal when applied to the full supervisor. Their work is able to improve their abstraction by artificially giving transitions zero cost based on the sequentially-dependent ordering of events. Here, we relax the requirement on a specific ordering of the dependent events, while maintaining the optimal relationship between upper and lower levels of the hierarchy. This present paper also extends the authors' work (Vilela and Hill 2020) where we developed a new notion of equivalence based on cost equivalence and weak bisimulation that we term *priced-observation equivalence*. This equivalence allows the supervisor abstraction to be generated compositionally. This helps to avoid the explosion of the state space that arises from having to first synthesize the full supervisor before the abstraction can be applied. Here, we also show that models with artificial zero-cost transitions can be created compositionally employing the new relaxed sequential dependence definition. An example cooperative robot control application is used to demonstrate the improvements achieved by the compositional approach to abstraction proposed by this paper.

Keywords Planning · Supervisory control · Abstraction · Decomposition · Hierarchy

This article belongs to the Topical Collection: *Topical Collection on Control 2022*
Guest Editors: Joerg Raisch, Carla Seatzu and Shigemasa Takai

Part of the topical collection entitled “New approaches for supervisory control and scheduling of discrete event systems”

✉ Juliana Vilela
vilelajn@udmercy.edu

Richard Hill
hillrc@udmercy.edu

¹ Department of Electrical and Computer Engineering, University of Detroit Mercy, Detroit, MI 48221 USA

² Department of Mechanical Engineering, University of Detroit Mercy, Detroit, MI 48221 USA

1 Introduction

Planning is a problem with great importance in a wide range of fields. Generally, planning is the process of analyzing how a system changes its state based on the occurrence of actions, followed by choosing which actions to apply to the system to achieve a desired goal. Examples of planning include: path planning, navigation planning, manipulation planning, perception planning, and job-shop scheduling. Research has been done across these different notions of planning to make the process more efficient. According to Qiang (2012), an efficient planning algorithm is the “one able to find a good plan quickly”.

Focusing on the development of general algorithms for planning, Ghallab et al. (2004) proposed employing discrete-event system models (a model widely used in the field of computing). Discrete-event systems can be represented by discrete graphs, where nodes are states and transitions are actions. Based on this representation, the classical planning problem can be seen as a search for a path in a graph from the initial state to a goal state with the optimization of a criterion. However, planning problems are highly complex and it can be difficult to explicitly model a system’s behavior. To deal with this difficulty, the use of Supervisory Control Theory (SCT) has emerged as an interesting tool.

With SCT, the system and specifications are modeled separately and then tools are applied to generate the supervisor, which can achieve the most permissive closed-loop behavior that maintains safety and non-blockingness. One approach to solve planning problems is to use the supervisor as the search space to find a plan that optimizes a defined criterion, such as time or energy [(Hill and Lafortune 2017; Pena et al. 2016; Dulce-Galindo et al. 2019; Hagebring and Lennartson 2018; Ware and Su 2017; Su 2012; Fabre and Jezequel 2009), and (Bravo et al. 2018)].

The concept of synthesizing an optimal supervisor, when not directly related to the application of planning, can be seen as the solution of this type of problem [(Huang and Kumar 2007; Sengupta and Lafortune 1993; Asarin and Maler 1999; Brandin and Wonham 1994; Su et al. 2012) and (Hyun-Wook et al. 2010)]. In general, the synthesis of an optimal supervisor is beyond traditional supervisor synthesis, which only restricts the system to the set of legal behaviors. An optimal supervisor further limits the system’s behaviors to those that minimize or maximize some criterion. Most existing approaches to synthesizing an optimal supervisor rely on a monolithic approach to modeling the system, which can have prohibitive computational complexity.

Efforts have been directed to improve the scalability of planning using SCT. Modular and compositional algorithms have been applied when a system can be decomposed into sub-systems.

Su (2013) proposes a distributed approach to time optimal supervisory control, but it requires the synthesis of a global coordinator through the composition of all local components and requirements, which also can have prohibitive complexity.

Hagebring and Lennartson (2018) and Ware and Su (2017) represent their models as standard tick automata, which has the effect of dramatically increasing their model sizes, thereby reducing the applicability of the approaches for more complex problems. Hagebring and Lennartson (2018) propose the application of local optimization and abstraction to each sub-system, which removes all non-optimal local paths and merges sequences of local events into single events. This is followed by a compositional optimization, which synchronizes a sub-set of components generating a larger sub-system to which a second iteration of the local optimization is applied. This cycle is repeated until only one model remains, representing the optimal global solution by construction.

Ware and Su (2017) propose two methods for synthesizing the time-optimal accepting trace for timed automata with only controllable events based on abstraction and pruning. For the first method, the sub-systems are composed together and all local events are abstracted. When a single automaton is achieved, Dijkstra's algorithm is applied, producing the time-optimal accepting trace. The second method is a modification of the first algorithm, applying pruning instead of abstraction to reduce the size of the automata. However, when a trace is returned, it is not guaranteed to be optimal since the pruning can remove all time-optimal accepting traces.

Bravo et al. (2018) generate results specifically for production systems with symmetrically reachable idle states (states with inactive behavior). Such a system is shown to be decomposable into factors that are optimized independently and then the partial solutions are concatenated to generate the minimum makespan controllable sublanguage.

van Putten et al. (2020) propose an approach for the synthesis of a throughput-optimal supervisor for a manufacturing system. In their modeling, they use the concept of activities which express some deterministic functionality, such as moving a product, in terms of actions. With that they create a high-level model composed by a plant model, which describes the activity sequences that are available, and specifications models, that describes the order in which activities should be executed. The lower-level model has for each activity an acyclic graph that defines their constituent actions and dependencies between them. The system's behavior, including timing, is captured from the lower level by variables and guards of the automata used in the high-level models. A supervisor is synthesized for the high-level, but for that they remove all the variables related to time from the models prior to synthesis and return them once the supervisor is obtained. This procedure affects the guarantee of controllability and nonblockingness of the supervisor. To the resultant monolithic supervisor, they apply a game theoretic method, called ratio games, to find the control that optimizes the throughput.

Hill and Lafortune (2017) develop an approach for generating approximately time optimal supervisory control logic for a cooperative multi-robot system employing hierarchy and decomposition based on the results of Hill and Lafortune (2016). These works do not employ tick automata and do not rely on the identification of idle states. The work of (Hill and Lafortune 2016) proposes a hierarchical approach based on a new notion, *cost equivalence*, that aggregates states with common future costs, as shown in Fig. 1. This equivalence generates an abstraction of the supervisor that with additional conditions, such as an additive cost function and only transitions with zero cost can be "hidden," guarantees that an optimal plan found on the abstraction is also optimal when applied to the underlying full supervisor.

The work of this paper improves upon (Hill and Lafortune 2016) by developing conditions under which the abstraction can be generated more efficiently. Specifically, this paper is an extended version of the conference paper (Vilela and Hill 2020) where the proposed requirements are relaxed and additional details are provided, including a more developed example. In this work, we employ a new notion of equivalence introduced by Vilela and Hill (2020) based on *cost equivalence* and *weak bisimulation* that we term *priced-observation equivalence*. This class of equivalence is used to generate an abstraction based on the aggregation of states with futures that share the same costs and observed event labels. This new equivalence allows the abstraction to be generated compositionally, rather than first having to synthesize the full global supervisor as done by Hill and Lafortune (2016). The work of Hill and Lafortune (2016) and Vilela and Hill (2020) include as part of their process an approach for artificially reducing the cost of certain transitions to zero. In this work, we

Fig. 1 Hierarchical planning (Hill and Lafortune 2016)

provide a relaxation of the conditions to perform this transformation, and demonstrate that it can be performed compositionally.

The remainder of this paper is divided into four sections: Section 2 provides the necessary preliminary definitions, Section 3 develops the main theoretical results, Section 4 provides a detailed example, and Section 5 presents the paper's conclusions.

2 Preliminaries

This paper employs *discrete-event models* where behaviors are represented using strings of events taken from a finite alphabet Σ . Σ^* is the set of all finite strings of events in Σ , including the empty string ε , and the *concatenation of strings* $s, u \in \Sigma^*$ is written as su . A subset $L \subseteq \Sigma^*$ is called a *language*. The possible behaviors of a system are represented by its automaton generator. Specifically, this paper employs *weighted automata*, $G = (Q, \Sigma, \rightarrow, q_0, Q_m, f)$, where Q is a finite set of states, Σ is a finite set of events, $\rightarrow \subseteq Q \times \Sigma \times Q$ is the (partial) state transition relation, $q_0 \in Q$ is the initial state, $Q_m \subseteq Q$ is the finite set of marked states, and f is the cost function that represents a mapping from a transition, $Q \times \Sigma \times Q$, to a nonnegative real number, $[0, \infty)$. The cost will typically represent the expenditure of energy or time associated with a transition. The transition relation is written in infix notation, $x \xrightarrow{\sigma} y$, and is extended to traces in the natural way. The cost of a trace $s = \sigma_1\sigma_2\ldots\sigma_{n-1}$ is defined as the sum of the costs of each transition, $C(q_1 \xrightarrow{s} q_n) = \sum_{i=1}^{n-1} f(q_i \xrightarrow{\sigma_i} q_{i+1})$. G is assumed to be deterministic, therefore, if $x \xrightarrow{\sigma} y_1$ and $x \xrightarrow{\sigma} y_2$, then $y_1 = y_2$. The *generated language* of an automaton G is $L(G) := \{s \in \Sigma^* | q_0 \xrightarrow{s} q, \text{ s.t. } q \in Q\}$ and the *marked language* is $L_m(G) := \{s \in \Sigma^* | q_0 \xrightarrow{s} q, \text{ s.t. } q \in Q_m\}$.

Given an automaton $G = (\Sigma, Q, \rightarrow, Q_m, q_0, f)$, a state $q \in Q$ is termed *accessible* if $q_0 \xrightarrow{s} q$ for some $s \in \Sigma^*$. An automaton is accessible if every state is accessible and the accessible component of an automaton G is obtained by the operation $Ac(G)$. For the purposes of this paper, all the automata are considered accessible. The synchronous

operation of two automata G_1 and G_2 is modeled using the parallel composition operator, $\parallel$, as defined in the following.

Definition 1 The **parallel composition** of two weighted automata $G_1 = (Q_1, \Sigma_1, \rightarrow_1, q_{0,1}, Q_{m,1}, f_1)$ and $G_2 = (Q_2, \Sigma_2, \rightarrow_2, q_{0,2}, Q_{m,2}, f_2)$ is denoted $G_1 \parallel G_2 = Ac(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \rightarrow_{1,2}, (q_{0,1}, q_{0,2}), Q_{m,1} \times Q_{m,2}, f_{1,2})$, where $\rightarrow_{1,2}$ and $f_{1,2}$ are defined as follows:

- if $\sigma \in \Sigma_1 \cap \Sigma_2$, then $(q_1, q_2) \xrightarrow{\sigma}_{1,2} (q'_1, q'_2)$ and $f_{1,2}((q_1, q_2) \xrightarrow{\sigma}_{1,2} (q'_1, q'_2)) = \max(f_1(q_1 \xrightarrow{\sigma}_1 q'_1), f_2(q_2 \xrightarrow{\sigma}_2 q'_2))$;
- if $\sigma \in \Sigma_1$ and $\sigma \notin \Sigma_2$, then $(q_1, q_2) \xrightarrow{\sigma}_{1,2} (q'_1, q_2)$ and $f_{1,2}((q_1, q_2) \xrightarrow{\sigma}_{1,2} (q'_1, q_2)) = f_1(q_1 \xrightarrow{\sigma}_1 q'_1)$;
- if $\sigma \notin \Sigma_1$ and $\sigma \in \Sigma_2$, then $(q_1, q_2) \xrightarrow{\sigma}_{1,2} (q_1, q'_2)$ and $f_{1,2}((q_1, q_2) \xrightarrow{\sigma}_{1,2} (q_1, q'_2)) = f_2(q_2 \xrightarrow{\sigma}_2 q'_2)$;
- otherwise, $\rightarrow_{1,2}$ and $f_{1,2}$ are undefined.

In this definition, the function $\max$ is used to establish the cost of transitions with shared events. This decision makes sense, in particular, when costs are related to time. For example, if a task is running in parallel in two automata, the one that takes the greater amount of time determines the overall required time for both automata to complete the task.

The *natural projection* $P_i : \Sigma^* \rightarrow \Sigma_i^*$ maps strings in Σ^* to strings in Σ_i^* by erasing all events not contained in Σ_i . We can also define the *inverse projection* $P_i^{-1}(t) := \{s \in \Sigma^* : P_i(s) = t\}$. These definitions can be naturally extended to languages and then applied to provide an alternate definition of the parallel composition in terms of languages: $L(G_1 \parallel G_2) := P_1^{-1}(L(G_1)) \cap P_2^{-1}(L(G_2))$ and $L_m(G_1 \parallel G_2) := P_1^{-1}(L_m(G_1)) \cap P_2^{-1}(L_m(G_2))$. For further details on discrete-event systems and the base SCT, the reader is referred to Cassandras and Lafortune (2008).

A partition automaton, G^h , is generated from an automaton $G = (Q, \Sigma, \rightarrow, q_0, Q_m, f)$ by aggregating its states into distinct sets such that transitions inside a partition group cannot be observed. Defined more formally, a partition automaton G^h is a weighted automaton $G^h = (Q^h, \Sigma^h, \rightarrow_h, q_0^h, Q_m^h, f^h)$, where Q^h is a set of partition states $\{q_1^h, q_2^h, \dots, q_n^h\}$ such that $\cup q^h = Q$, $q_i^h \neq \emptyset$, $1 \leq i \leq |Q^h|$, and $q_i^h \cap q_j^h = \emptyset$ for $i \neq j$. The initial state of the partition automaton is defined such that $q_0 \in q_0^h$, and marked states are defined such that $q^h \in Q_m^h \Leftrightarrow \exists q \in Q_m$ s.t. $q \in q^h$. The transition relation $\rightarrow_h \subseteq Q^h \times \Sigma^h \times Q^h$ includes transitions from the original automaton that are “between” partitions. A transition $q^h \xrightarrow{\sigma}_h q'^h$ exists in G^h if and only if there exists a transition $q \xrightarrow{\sigma} q'$ in the original automaton such that $q \in q^h$ and $q' \in q'^h$. The transition’s cost in the partition automata $f^h(q^h \xrightarrow{\sigma}_h q'^h)$ will equal $f(q \xrightarrow{\sigma} q')$. This situation is represented in Fig. 2.

For brevity, we introduce the notation, $x \xRightarrow{s} y$, to represent how a string in an automaton G is observed in its corresponding partition automaton G^h . Otherwise stated, $x \xRightarrow{s} y$ with $s = \sigma_1 \sigma_2 \dots \sigma_n \in \Sigma^{h*}$, denotes the existence of $x \xrightarrow{r} y$ with exactly the order of events in s , but with an arbitrary number of “hidden” events, $\tau \in \Sigma \setminus \Sigma^h$, shuffled with the observable events in s , that is, $r = t_1 \sigma_1 t_2 \sigma_2 t_3 \dots t_n \sigma_n t_{n+1}$, where $t_i \in \{\Sigma \setminus \Sigma^h\}^*$. Further, we use the

Fig. 2 Cost of transitions in a partition automaton

following notation to capture the cost of an observed string $s \in \Sigma^{h*}$, $C(q_1 \xRightarrow{s} q_{n+1}) = \sum_{1 \leq i \leq n: \sigma_i \in \Sigma^h} f(q_i \xrightarrow{\sigma_i} q_{i+1})$.

A path in G whose transitions are visited in an order consistent with the order of the transitions in G^h , is termed a *realization* of the corresponding “high-level” path. This consistency of ordering is named *trace-cost consistency* for individual trajectories and is denoted: $q_1 \rightarrow q_n \in q_1^h \rightarrow_h q_n^h$.

In Hill and Lafortune (2016), the partition automaton is generated by aggregating states that are *cost equivalent*. This notion of equivalence requires that states must have observed futures with the same cost and consistent marking. The partitions are generated such that only transitions with *zero cost* are “hidden”. Such partition systems are termed *zero-cost reachable*. The following result from Hill and Lafortune (2016) demonstrates that when partition systems maintain this property and employ cost equivalence, then an optimal plan chosen in the abstracted automaton has at least one realization in the underlying automaton and all such realizations are optimal. We define the *optimal path* between two states of G , $q_1 \xrightarrow{00} q_n$, as the sequence of transitions between q_1 and q_n with the smallest path cost.

Theorem 1 (Hill and Lafortune (2016)) *Let $G^h = (Q^h, \Sigma^h, \rightarrow_h, q_0^h, Q_m^h, f^h)$ be a partition automaton generated by cost-equivalent abstraction of a weighted automaton $G = (Q, \Sigma, \rightarrow, q_0, Q_m, f)$ such that $\{G, G^h\}$ is zero-cost reachable. Let an optimal marked path between two states in G^h be $q_1^h \xrightarrow{00} q_k^h \in Q_m^h$. Then there exists at least one trace-cost consistent realization of this path in G , and all such realizations are optimal, $q_1 \xrightarrow{00} q_n \in q_1^h \xrightarrow{00} q_k^h$.*

Hill and Lafortune (2016) further introduce a technique to improve the amount of reduction that can be achieved for a given automaton. The idea is to create transitions that artificially have zero cost, while maintaining the overall cost of each marked path. For that, the authors introduce the concept of *sequential dependence*. A pair of events (σ_a, σ_b) is defined to be a sequentially-dependent ordering if the occurrence of the event σ_a in a marked path is always eventually followed by the event σ_b . Consider a sequentially-dependent ordering $\sigma_a \sigma_b$ in an automaton G , where σ_a is the independent event with cost $f(p \xrightarrow{\sigma_a} p') = c_a$, and σ_b is the dependent event with cost $f(q \xrightarrow{\sigma_b} q') = c_b$. If the cost of the dependent transitions are consistent in G , it is possible to perform a “*lumping*” operation, which transfers the cost of all transitions with label σ_b to transitions with label σ_a , $f(p \xrightarrow{\sigma_a} p') = c_a + c_b$ and $f(q \xrightarrow{\sigma_b} q') = 0$. The rest of the automaton G remains unaltered, resulting in the lumped automaton G^* . The property of costs being consistent for all transitions with the same event labels is referred to as *time-separability*.

Hill and Lafortune (2016) prove in their Proposition 4 that all marked paths in a lumped automaton G^* possess the same cost as in the original automaton G . This result allows an abstraction of G^* to be used for planning.

3 Main results

The main results of this paper improve upon (Hill and Lafortune 2016) by introducing a new notion of equivalence that allows the abstraction to be generated compositionally. This concept was introduced in the conference paper (Vilela and Hill 2020). Here we provide additional details and employ a definition of sequential dependence that is relaxed from the version employed in Vilela and Hill (2020) and Hill and Lafortune (2016).

The work of Hill and Lafortune (2016) relies on the existence of sequentially-dependent orderings of events to create transitions with “virtual” zero-cost to improve the amount of abstraction achieved in the generation of the high-level in the hierarchy. However, their definition of sequential dependence is restricted by a consistent ordering of the occurrence of dependent events. Based on this definition, the pair of events (σ_a, σ_b) in the automaton G presented in Fig. 3 is not considered sequentially-dependent, since if we choose an arbitrary marked path that includes the event σ_a , we can have the order $\sigma_a\sigma_b$ or $\sigma_b\sigma_a$. In this paper, we relax the sequential dependence requirement such that a specific ordering between the dependent events is not necessary. We will classify the pair of events (σ_a, σ_b) in the example of Fig. 3 to be a *sequentially-dependent pair* without requiring them to be a *sequentially-dependent ordering*. Under this new definition of sequential dependence, we also define a new lump operation. We will show that the new lump operation will not affect the overall cost of marked paths through the automaton G , maintaining the hierarchical approach to planning allowed by Theorem 1. Further, we will show that the compositional approach to generating the abstraction proposed in Vilela and Hill (2020) holds under the new relaxed sequential dependence definition.

This new notion of sequential dependence is defined for a pair of events (σ_a, σ_b) , such that for an arbitrary marked path, the occurrence of one of the events in the path implies the occurrence of the other event. For example, if we have two occurrences of σ_a in a string, then we will have two occurrences of σ_b as well, without regard to their ordering. The marked path could include any permutation of these events, $\{\sigma_a\sigma_a\sigma_b\sigma_b, \sigma_a\sigma_b\sigma_a\sigma_b, \sigma_a\sigma_b\sigma_b\sigma_a, \sigma_b\sigma_a\sigma_b\sigma_a, \sigma_b\sigma_a\sigma_a\sigma_b, \sigma_b\sigma_b\sigma_a\sigma_a\}$. We will represent such sets by a permutation function. This set, for example, will be represented $Perm(\sigma_a^2, \sigma_b^2)$,

Fig. 3 Sequential dependence of the pair of events (σ_a, σ_b)

where $\sigma_a^2 = \sigma_a \sigma_a$ and $\sigma_b^2 = \sigma_b \sigma_b$. The new relaxed notion of sequential dependence is formally presented in Definition 2.

Definition 2 Consider an automaton $G = (Q, \Sigma, \rightarrow, q_0, Q_m, f)$ which possesses a pair of events (σ_a, σ_b) with corresponding natural projections $P_a : \Sigma^* \rightarrow \sigma_a^*$ and $P_b : \Sigma^* \rightarrow \sigma_b^*$. The pair of events is **sequentially dependent** in G if $\forall s \in \Sigma^*$ such that $q_0 \xrightarrow{s} q$ exists for some $q \in Q_m$ with $P_a(s) = \sigma_a^n$, then $P_b(s) = \sigma_b^n$ where $n \in \mathbb{N}$.

Based on this new sequential dependence definition, we update the definition of the lump operation in Definition 3. Because there are not specific dependent and independent events, the cost can be transferred in either direction between the events. For example, from σ_a to σ_b or from σ_b to σ_a . However, the direction of the lumping must be kept consistent throughout in the entire automaton.

Definition 3 Let there be a weighted automaton $G = (Q, \Sigma, \rightarrow, q_0, Q_m, f)$, with $\sigma_a, \sigma_b \in \Sigma$, where the pair of events (σ_a, σ_b) is *sequentially dependent*. If the pair of events are time-separable in G , with $f(p \xrightarrow{\sigma_a} p') = c_a$, and $f(q \xrightarrow{\sigma_b} q') = c_b$, then a *lumped automaton* $G^* = (Q, \Sigma, \rightarrow, q_0, Q_m, f^*)$ can be created by setting $f^*(q \xrightarrow{\sigma_b} q') = 0$, and $f^*(p \xrightarrow{\sigma_a} p') = c_a + c_b$. The rest of the automaton remains unaltered.

Now, we need to show that the cost of a marked sequence is kept unaltered in the lumped automaton G^* following this new definition of the lumping operation.

Proposition 1 Let there be an automaton $G = (Q, \Sigma, \rightarrow, q_0, Q_m, f)$ with a time-separable and sequentially-dependent pair of events (σ_a, σ_b) from which the lumped automaton $G^* = (Q, \Sigma, \rightarrow, q_0, Q_m, f^*)$ is generated. For all $s \in \Sigma^*$ such that $q_0 \xrightarrow{s} q_n$ exists for some $q_n \in Q_m$, it is true that $C(q_0 \xrightarrow{s} q_n) = C^*(q_0 \xrightarrow{s} q_n)$.

Because the cost of any marked sequence s in G^* is the same as in G , we have that an optimal plan found in G^* is also an optimal plan in G , as settled before by Hill and Lafortune (2016).

Hill and Lafortune (2016) employ the original weighted automaton G to generate a modified lumped automaton G^* , which is then used to generate the cost-equivalent, zero-cost reachable partition automaton G^h . The reduced G^h is used to find an optimal global plan (marked path). It is proven by Hill and Lafortune (2016) that all realizations in G of an optimal plan in G^h are also optimal, and that any path realized in G^* has equal cost in the original automaton G . Therefore, any realization in G of an optimal plan in $(G^*)^h$ is optimal as well (captured by Theorem 1).

This paper follows the same process of finding an optimal global plan developed by Hill and Lafortune (2016), however, it proposes a more efficient approach for generating the abstraction G^h . Given that G , generally, is composed of a set of smaller automata, $G_1 \| G_2 \| \dots \| G_n$, this paper finds conditions under which G^h can be generated compositionally, without the need to compute G first as was done in Hill and Lafortune (2016). Specifically, we show that it is possible to distribute the “lumping” operation over the parallel composition operator, and if the partitioning employs a newly defined class of equivalence, then G^h can be generated compositionally, Fig. 4. The new notion of priced-observation equivalence is stronger than the cost equivalence introduced in Hill and Lafortune (2016). Therefore, all results from Hill and Lafortune (2016) related to optimality

Fig. 4 Different approaches to create the high-level of the hierarchy

of a plan between a partition automaton and its underlying automaton are also valid under priced-observation equivalence.

With priced-observation equivalence, states must have futures with the same cost and the same observed event labels, as well as consistent marking. One can consider this equivalence a combination of cost equivalence and weak bisimulation.

Definition 4 Let there be two weighted automata, $G_1 = (Q_1, \Sigma, \rightarrow_1, q_{0,1}, Q_{m,1}, f_1)$ and $G_2 = (Q_2, \Sigma, \rightarrow_2, q_{0,2}, Q_{m,2}, f_2)$, and a set of hidden events $(\Sigma \setminus \Sigma^h)$. An equivalence relation $\approx_{PO} \subseteq Q_1 \times Q_2$ is a *priced-observation equivalence* if for any states $q_1 \in Q_1$ and $q_2 \in Q_2$, $q_1 \approx_{PO} q_2$ implies:

- (i) if $q_1 \xrightarrow{s} q'_1$ for some $s \in \Sigma^{h*}$, then $\exists q'_2 \in Q_2$ s.t. $q_2 \xrightarrow{s} q'_2$ where $C(q_1 \xrightarrow{s} q'_1) = C(q_2 \xrightarrow{s} q'_2) \wedge q'_1 \in Q_{m,1} \Rightarrow q'_2 \in Q_{m,2} \wedge q'_1 \approx_{PO} q'_2$; and
- (ii) if $q_2 \xrightarrow{s} q'_2$ for some $s \in \Sigma^{h*}$, then $\exists q'_1 \in Q_1$ s.t. $q_1 \xrightarrow{s} q'_1$ where $C(q_2 \xrightarrow{s} q'_2) = C(q_1 \xrightarrow{s} q'_1) \wedge q'_2 \in Q_{m,2} \Rightarrow q'_1 \in Q_{m,1} \wedge q'_1 \approx_{PO} q'_2$.

G_1 and G_2 are said to be priced-observation equivalent if $q_{0,1} \approx_{PO} q_{0,2}$. This follows from Definition 4 since all reachable states also will be priced-observation equivalent. As an example of priced-observation equivalence, we have the automata G and G^h in Fig. 5, where the events $\{\beta_1, \beta_2\}$ are abstracted away to generate G^h . In the abstraction, β_1 and

Fig. 5 Example of priced-observation equivalence

β_2 are replaced by the silent event τ . Two groups of priced-observation equivalent states are then generated, resulting in the automaton G^h in Fig. 5c, which is priced-observation equivalent to G , $G \approx_{PO} G^h$.

We will show now that the global partition automaton (abstraction) can be generated compositionally. This requires that priced-observation equivalence be a congruence with respect to parallel composition. We will show that this is true when a partition system $\{G, G^h\}$ is generated by a priced-observation equivalent abstraction, $G \approx_{PO} G^h$, and the shared events are kept observable.

Proposition 2 *Let there be a weighted automaton $G_1 = (Q_1, \Sigma_1, \rightarrow_1, q_{0,1}, Q_{m,1}, f_1)$ from which a partition automaton $G_1^h = (Q_1^h, \Sigma_1^h, \rightarrow_1^h, q_{0,1}^h, Q_{m,1}^h, f_1^h)$ is generated by priced-observation equivalent abstraction, $G_1 \approx_{PO} G_1^h$. For any third automaton*

$G_2 = (Q_2, \Sigma_2, \rightarrow_2, q_{0,2}, Q_{m,2}, f_2)$, if the set of observed events $\Sigma_1^h \supseteq \Sigma_1 \cap \Sigma_2$, then $G_1 \| G_2 \approx_{PO} G_1^h \| G_2$.

Proof In this proof we will establish that $G_1 \| G_2 \approx_{PO} G_1^h \| G_2$ by demonstrating that $(q_{0,1}, q_{0,2}) \approx_{PO} (q_{0,1}^h, q_{0,2})$.

- From Definition 1 we have the initial state $(q_{0,1}, q_{0,2})$ for $G_1 \| G_2$ and the initial state $(q_{0,1}^h, q_{0,2})$ for $G_1^h \| G_2$.

1) Let there be an observation $(q_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q_{1,1}, q_{1,2})$ in $G_1 \| G_2$, such that $\sigma \in \Sigma_1^h \cup \Sigma_2$ and $\Sigma_1^h \supseteq \Sigma_1 \cap \Sigma_2$. We will show that $\exists (q_{0,1}^h, q_{0,2}) \xrightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{1,2})$ in $G_1^h \| G_2$ with $C((q_{0,1}^h, q_{0,2}) \xrightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{1,2})) = C((q_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q_{1,1}, q_{1,2}))$ and that $(q_{1,1}^h, q_{1,2})$ agrees in marking with $(q_{1,1}, q_{1,2})$.

The observation $(q_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q_{1,1}, q_{1,2})$ implies the existence of the sequence $(q_{0,1}, q_{0,2}) \xrightarrow{t_1}_{1,2} (q'_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q'_{1,1}, q_{1,2}) \xrightarrow{t_2}_{1,2} (q_{1,1}, q_{1,2})$ where $t_i \in ((\Sigma_1 \setminus \Sigma_1^h) \setminus \Sigma_2)^*$, since by assumption $\Sigma_1^h \supseteq \Sigma_1 \cap \Sigma_2$. Analysing this sequence of transitions by parts we have:

1.i) $(q_{0,1}, q_{0,2}) \xrightarrow{t_1}_{1,2} (q'_{0,1}, q_{0,2})$: We know that $t_1 \in ((\Sigma_1 \setminus \Sigma_1^h) \setminus \Sigma_2)^*$, then invoking Definition 1, we have that $\exists q_{0,1} \xrightarrow{t_1}_1 q'_{0,1}$, but $\nexists q_{0,2} \xrightarrow{t_1}_2 q_{1,2}$. By construction of G_1^h , we have that $q_{0,1} \approx_{PO} q'_{0,1}$ and that $q_{0,1} \approx_{PO} q_{0,1}^h$. Therefore, by transitivity, $q_{0,1} \approx_{PO} q'_{0,1} \approx_{PO} q_{0,1}^h$. Furthermore, by Definition 1, $(q_{0,1}^h, q_{0,2}) \in Q_{m,1}^h \times Q_{m,2} \iff (q_{0,1}, q_{0,2}) \in Q_{m,1} \times Q_{m,2}$.

1.ii) $(q'_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q'_{1,1}, q_{1,2})$: Since $\sigma \in \Sigma_1^h \cup \Sigma_2$, there are three possible cases:

1.ii.a) $\sigma \in \Sigma_1^h \setminus \Sigma_2$: By Definition 1 we know that $\exists q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1}$ and $\nexists q_{0,2} \xrightarrow{\sigma}_2 q_{1,2}$ and $f_{1,2}((q'_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q'_{1,1}, q_{0,2})) = f_1(q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1})$. By assumption $G_1 \approx_{PO} G_1^h$, we have $q'_{0,1} \approx_{PO} q_{0,1}^h$. Given $\exists q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1}$ and invoking Definition 4, we have that $\exists q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h$ with $f_1^h(q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h) = f_1(q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1})$ and $q_{1,1}^h \in Q_{m,1}^h \iff q'_{1,1} \in Q_{m,1}$. Therefore, by Definition 1, in $G_1^h \| G_2$ $\exists (q_{0,1}^h, q_{0,2}) \xrightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{0,2})$ with cost $f_{1,2}((q_{0,1}^h, q_{0,2}) \xrightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{0,2})) = f_1^h(q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h) = f_1(q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1}) = f_{1,2}((q'_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q'_{1,1}, q_{0,2}))$ and $(q_{1,1}^h, q_{0,2}) \in Q_{m,1}^h \times Q_{m,2} \iff (q'_{1,1}, q_{0,2}) \in Q_{m,1} \times Q_{m,2}$.

1.ii.b) $\sigma \in \Sigma_1^h \cap \Sigma_2$: By Definition 1 we know that $\exists q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1}$ and $\exists q_{0,2} \xrightarrow{\sigma}_2 q_{1,2}$ and $f_{1,2}((q'_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q'_{1,1}, q_{1,2})) = \max(f_1(q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1}), f_2(q_{0,2} \xrightarrow{\sigma}_2 q_{1,2}))$. Since $\exists q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1}$ and $q_{0,1} \approx_{PO} q'_{0,1} \approx_{PO} q_{0,1}^h$, invoking Definition 4 provides that $\exists q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h$ in G_1^h with cost $f_1^h(q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h) = f_1(q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1})$ and $q_{1,1}^h \in Q_{m,1}^h \iff q'_{1,1} \in Q_{m,1}$. Therefore, by Definition 1, we have that $\exists (q_{0,1}^h, q_{0,2}) \xrightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{1,2})$ in $G_1^h \| G_2$ with cost $f_{1,2}((q_{0,1}^h, q_{0,2}) \xrightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{1,2})) = \max(f_1^h(q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h), f_2(q_{0,2} \xrightarrow{\sigma}_2 q_{1,2})) = \max(f_1(q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1}), f_2(q_{0,2} \xrightarrow{\sigma}_2 q_{1,2})) = f_{1,2}((q'_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q'_{1,1}, q_{1,2}))$ and $(q_{1,1}^h, q_{1,2}) \in Q_{m,1}^h \times Q_{m,2} \iff (q'_{1,1}, q_{1,2}) \in Q_{m,1} \times Q_{m,2}$.

1.ii.c) $\sigma \in \Sigma_2 \setminus \Sigma_1^h$: This case follows from logic similar to that employed in **1.ii.a)**

1.iii) $(q'_{1,1}, q_{1,2}) \xrightarrow{t_2}_{1,2} (q_{1,1}, q_{1,2})$: We know that $t_2 \in ((\Sigma_1 \setminus \Sigma_1^h) \setminus \Sigma_2)^*$ thus this case follows from logic similar to that employed in **1.i)**.

- So far we have shown that given $(q_{0,1}, q_{0,2}) \xRightarrow{\sigma}_{1,2} (q_{1,1}, q_{1,2})$ in $G_1 \| G_2$, then $\exists (q_{0,1}^h, q_{0,2}) \xRightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{1,2})$ in $G_1^h \| G_2$ with $C((q_{0,1}^h, q_{0,2}) \xRightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{1,2})) = C((q_{0,1}, q_{0,2}) \xRightarrow{\sigma}_{1,2} (q_{1,1}, q_{1,2}))$ and $(q_{1,1}^h, q_{1,2})$ and $(q_{1,1}, q_{1,2})$ agree in marking. Applying the preceding logic over subsequent observations, we end up with the result that all observed sequences that exist in $G_1 \| G_2$ also exist in $G_1^h \| G_2$ with the same cost and marking.

2) The next step is to show the opposite direction. Let there be an observation $(q_{0,1}^h, q_{0,2}) \xRightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{1,2})$ in $G_1^h \| G_2$, such that $\sigma \in \Sigma_1^h \cup \Sigma_2$ and $\Sigma_1^h \supseteq \Sigma_1 \cap \Sigma_2$.

The observation $(q_{0,1}^h, q_{0,2}) \xRightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{1,2})$ implies the existence of the sequence $(q_{0,1}^h, q_{0,2}) \xrightarrow{t_1 \sigma t_2}_{1^h,2} (q_{1,1}^h, q_{1,2})$, where $t_i \in ((\Sigma_1 \setminus \Sigma_1^h) \setminus \Sigma_2)^*$, therefore, the observation $(q_{0,1}^h, q_{0,2}) \xRightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{1,2})$ implies that $(q_{0,1}^h, q_{0,2}) \xrightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{1,2})$ in $G_1^h \| G_2$.

We will now show the existence of the observation in $G_1 \| G_2$ in a similar manner to direction **1)** by proving the existence of a sequence $(q_{0,1}, q_{0,2}) \xrightarrow{t_1}_{1,2} (q'_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q'_{1,1}, q_{1,2}) \xrightarrow{t_2}_{1,2} (q_{1,1}, q_{1,2})$ with consistent cost and marking.

2.i) $(q_{0,1}, q_{0,2}) \xrightarrow{t_1}_{1,2} (q'_{0,1}, q_{0,2})$: By assumption $G_1^h \approx_{PO} G_1$, thus $q_{0,1}^h \approx_{PO} q_{0,1}$. However, in G_1 there are hidden events, given $t_1 \in ((\Sigma_1 \setminus \Sigma_1^h) \setminus \Sigma_2)^*$, then $\exists q_{0,1} \xrightarrow{t_1}_1 q'_{0,1}$ with $q_{0,1} \approx_{PO} q'_{0,1}$, by construction. By transitivity, $q_{0,1}^h \approx_{PO} q'_{0,1}$. Since $t_1 \notin \Sigma_2$, Definition 1 then provides that $\exists (q_{0,1}, q_{0,2}) \xrightarrow{t_1}_{1,2} (q'_{0,1}, q_{0,2})$ and $(q_{0,1}^h, q_{0,2}) \in Q_{m,1}^h \times Q_{m,2} \iff (q_{0,1}, q_{0,2}) \in Q_{m,1} \times Q_{m,2}$.

2.ii) $(q'_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q_{1,1}, q_{1,2})$: Since $\sigma \in \Sigma_1^h \cup \Sigma_2$, there are three possible cases:

2.ii.a) $\sigma \in \Sigma_1^h \setminus \Sigma_2$: By Definition 1 we know that $\exists q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h$ and $\nexists q_{0,2} \xrightarrow{\sigma}_2 q_{1,2}$ and $f_{1,2}((q_{0,1}^h, q_{0,2}) \xrightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{1,2})) = f_1^h(q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h)$. By assumption $G_1^h \approx_{PO} G_1$, we have $q_{0,1}^h \approx_{PO} q'_{0,1}$. Given $\exists q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h$ and invoking Definition 4, we have that $\exists q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1}$ with $f_1(q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1}) = f_1^h(q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h)$ and $q'_{1,1} \in Q_{m,1} \iff q_{1,1}^h \in Q_{m,1}^h$. Therefore, by Definition 1, in $G_1 \| G_2$ $\exists (q'_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q'_{1,1}, q_{0,2})$ with cost $f_{1,2}((q'_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q'_{1,1}, q_{0,2})) = f_1(q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1}) = f_1^h(q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h) = f_{1,2}((q_{0,1}^h, q_{0,2}) \xrightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{0,2}))$ and $(q'_{1,1}, q_{0,2}) \in Q_{m,1} \times Q_{m,2} \iff (q_{1,1}^h, q_{0,2}) \in Q_{m,1}^h \times Q_{m,2}$.

2.ii.b) $\sigma \in \Sigma_1^h \cap \Sigma_2$: By Definition 1 we know that $\exists q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h$ and $\exists q_{0,2} \xrightarrow{\sigma}_2 q_{1,2}$ and $f_{1,2}((q_{0,1}^h, q_{0,2}) \xrightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{1,2})) = \max(f_1^h(q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h), f_2(q_{0,2} \xrightarrow{\sigma}_2 q_{1,2}))$. Since $\exists q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h$ and $q_{0,1}^h \approx_{PO} q'_{0,1}$, invoking Definition 4 provides that $\exists q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1}$ in G_1 with cost $f_1(q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1}) = f_1^h(q_{0,1}^h \xrightarrow{\sigma}_{1^h} q_{1,1}^h)$ and $q'_{1,1} \in Q_{m,1} \iff q_{1,1}^h \in Q_{m,1}^h$. Therefore, by Definition 1, we have that $\exists (q'_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q'_{1,1}, q_{1,2})$ in $G_1 \| G_2$ with

cost $f_{1,2}((q'_{0,1}, q_{0,2}) \xrightarrow{\sigma}_{1,2} (q'_{1,1}, q_{1,2})) = \max(f_1(q'_{0,1} \xrightarrow{\sigma}_1 q'_{1,1}), f_2(q_{0,2} \xrightarrow{\sigma}_2 q_{1,2})) = \max(f_1^h(q_{0,1} \xrightarrow{\sigma}_{1^h} q_{1,1}^h), f_2(q_{0,2} \xrightarrow{\sigma}_2 q_{1,2})) = f_{1^h,2}((q_{0,1}^h, q_{0,2}) \xrightarrow{\sigma}_{1^h,2} (q_{1,1}^h, q_{1,2}))$ and $(q'_{1,1}, q_{1,2}) \in Q_{m,1} \times Q_{m,2} \iff (q_{1,1}^h, q_{1,2}) \in Q_{m,1}^h \times Q_{m,2}$.

2.ii.c) $\sigma \in \Sigma_2 \setminus \Sigma_1^h$: This case follows from logic similar to that employed in **2.ii.a**).

2.iii) $(q'_{1,1}, q_{1,2}) \xrightarrow{t_2}_{1,2} (q_{1,1}, q_{1,2})$: This case follows from logic similar to that employed in **2.i**).

- Applying this same logic to subsequent observations, we end up with the result that all observed sequences in $G_1^h \| G_2$ also exist in $G_1 \| G_2$, with the same cost and marking.
- Based on **1**) and **2**) and invoking Definition 4, we have that $(q_{0,1}, q_{0,2}) \approx_{PO} (q_{0,1}^h, q_{0,2})$, thus $G_1 \| G_2 \approx_{PO} G_1^h \| G_2$. $\square$

The fact that priced-observation equivalence is a congruence with respect to the parallel composition is an important result that allows the global partition automaton G^h to be generated compositionally as proven in the following theorem.

Theorem 2 *Let there be two weighted automata $G_1 = (Q_1, \Sigma_1, \rightarrow_1, q_{0,1}, Q_{m,1}, f_1)$ and $G_2 = (Q_2, \Sigma_2, \rightarrow_2, q_{0,2}, Q_{m,2}, f_2)$. If the partition automata are generated by a priced-observation equivalent abstraction and $\Sigma^h \supseteq \Sigma_1 \cap \Sigma_2$, then $G^h = (G_1 \| G_2)^h \approx_{PO} G_1^h \| G_2^h$.*

Proof Generating a partition automaton for G_1 using priced-observation equivalence, we have $G_1 \approx_{PO} G_1^h$. By Proposition 2, we then have $G_1 \| G_2 \approx_{PO} G_1^h \| G_2$. Generating a partition automaton for G_2 , again using priced-observation equivalence, we have $G_2 \approx_{PO} G_2^h$. Again invoking Proposition 2 with G_1^h as the third automaton, we have $G_1^h \| G_2 \approx_{PO} G_1^h \| G_2^h$ since parallel composition is commutative. By transitivity, we end up with $G_1 \| G_2 \approx_{PO} G_1^h \| G_2^h$. By construction, $G \approx_{PO} G^h$, so $G_1 \| G_2 \approx_{PO} (G_1 \| G_2)^h$. Therefore, again by transitivity, we have $G^h = (G_1 \| G_2)^h \approx_{PO} G_1^h \| G_2^h$. $\square$

Now we desire that the relaxed version of sequential dependence be preserved over parallel composition. Proposition 3 shows that this is true if the pair of events is sequentially dependent in all automata that have those events in their alphabet.

Proposition 3 *Let there be two weighted automata $G_1 = (Q_1, \Sigma_1, \rightarrow_1, q_{0,1}, Q_{m,1}, f_1)$ and $G_2 = (Q_2, \Sigma_2, \rightarrow_2, q_{0,2}, Q_{m,2}, f_2)$, where G_1 has a sequentially-dependent pair of events (σ_a, σ_b) . The parallel composition $G_1 \| G_2$ will also have the sequentially-dependent pair of events (σ_a, σ_b) , if:*

- $\sigma_a, \sigma_b \in (\Sigma_1 \cap \Sigma_2)$ and G_2 also has the sequentially-dependent pair of events (σ_a, σ_b) ; or;
- $\sigma_a, \sigma_b \in (\Sigma_1 \setminus \Sigma_2)$.

Proof Let $\Sigma = \Sigma_1 \cup \Sigma_2$, $P_a : \Sigma^* \rightarrow \sigma_a^*$, $P_b : \Sigma^* \rightarrow \sigma_b^*$, $P_{ab} : \Sigma^* \rightarrow \{\sigma_a, \sigma_b\}^*$, $P_1 : \Sigma^* \rightarrow \Sigma_1^*$, and $P_2 : \Sigma^* \rightarrow \Sigma_2^*$. It is given that the pair of events (σ_a, σ_b) is sequentially-dependent in G_1 and $\sigma_a, \sigma_b \in \Sigma_1$.

- Following Definition 2, we first need to show that $\forall s \in L_m(G_1 \| G_2)$ for which $P_a(s) = \sigma_a^n$, we have that $P_b(s) = \sigma_b^n$.

- Let $s \in L_m(G_1 \| G_2) = P_1^{-1}(L_m(G_1)) \cap P_2^{-1}(L_m(G_2))$, and $P_a(s) = \sigma_a^n$. Thus, $s \in P_1^{-1}(L_m(G_1))$ and $s \in P_2^{-1}(L_m(G_2))$.
- Since $s \in P_1^{-1}(L_m(G_1))$, $\exists s_1 \in L_m(G_1)$ such that $P_1(s) = s_1$.
- Also since $s \in P_2^{-1}(L_m(G_2))$, $\exists s_2 \in L_m(G_2)$ such that $P_2(s) = s_2$.
- Given $\sigma_a \in \Sigma_1$ and $P_1(P_a(s)) = \sigma_a^n$, then $P_a(P_1(s)) = P_a(s_1) = \sigma_a^n$.
- G_1 has the sequentially-dependent pair of events (σ_a, σ_b) , so given $s_1 \in L_m(G_1)$ and $P_a(s_1) = \sigma_a^n$, from Definition 2, $P_b(s_1) = \sigma_b^n$. Also, $P_{ab}(s_1) = r_1 \in \text{Perm}(\sigma_a^n, \sigma_b^n)$.
- Now analyzing the two possible cases:
 - i) for the case $\sigma_a, \sigma_b \in \Sigma_1 \cap \Sigma_2$: from the assumption $P_a(s) = \sigma_a^n$, we have $P_a(P_2(s)) = P_a(s_2) = \sigma_a^n$ since $\sigma_a \in \Sigma_2$. The pair of events (σ_a, σ_b) is also sequentially dependent in G_2 , then from Definition 2, $P_b(s_2) = \sigma_b^n$ and $P_{ab}(s_2) = r_2 \in \text{Perm}(\sigma_a^n, \sigma_b^n)$. Since $P_{ab}(P_1^{-1}(s_1)) = r_1$ and $P_{ab}(P_2^{-1}(s_2)) = r_2$, $P_{ab}(P_1^{-1}(s_1)) \cap P_{ab}(P_2^{-1}(s_2)) = \epsilon$ if $r_1 \neq r_2$, or $P_{ab}(P_1^{-1}(s_1)) \cap P_{ab}(P_2^{-1}(s_2)) = r_1$, if $r_1 = r_2$. Therefore, $P_{ab}(P_1^{-1}(s_1)) \cap P_{ab}(P_2^{-1}(s_2))$ equals ϵ or r_1 .
 - ii) for $\sigma_a, \sigma_b \in \Sigma_1 \setminus \Sigma_2$, $P_{ab}(s_2) = \epsilon$, and $P_{ab}(P_2^{-1}(s_2)) = (\sigma_a, \sigma_b)^*$. Therefore, $P_{ab}(P_1^{-1}(s_1)) \cap P_{ab}(P_2^{-1}(s_2)) = r_1$
- Considering both i) and ii), $P_{ab}(P_1^{-1}(s_1)) \cap P_{ab}(P_2^{-1}(s_2)) \subset \{\epsilon, r_1\}$. From a property of natural projection, $P_{ab}(P_1^{-1}(s_1) \cap P_2^{-1}(s_2)) \subseteq P_{ab}(P_1^{-1}(s_1)) \cap P_{ab}(P_2^{-1}(s_2)) \subset \{\epsilon, r_1\}$.
- Because $s \in P_1^{-1}(s_1) \cap P_2^{-1}(s_2)$ we have $P_{ab}(s) \subset \{\epsilon, r_1\}$. This provides that $P_{ab}(s) = \epsilon$ or $P_{ab}(s) = r_1$.
- Since it is given $P_a(s) = \sigma_a^n$, this removes the possibility of $P_{ab}(s) = \epsilon$. Therefore, $P_{ab}(s) = r_1$.
- Since $r_1 \in \text{Perm}(\sigma_a^n, \sigma_b^n)$, we have that $P_b(s) = \sigma_b^n$. Therefore, the pair of events (σ_a, σ_b) is sequentially-dependent in $G_1 \| G_2$. $\square$

The sequential dependence property can then be employed to generate the lumped automaton G^* as defined in Definition 3. Specifically, it is desired to perform the “lumping” compositionally. Conditions under which the lumping operation can be distributed over parallel composition, $G^* = G_1^* \| G_2^*$, are presented in Propositions 4 and 5, which correspond to cases i) and ii) of Proposition 3, respectively.

Proposition 4 *Let there be two weighted automata $G_1 = (Q_1, \Sigma_1, \rightarrow_1, q_{0,1}, Q_{m,1}, f_1)$ and $G_2 = (Q_2, \Sigma_2, \rightarrow_2, q_{0,2}, Q_{m,2}, f_2)$ with sequentially dependent pair of events (σ_a, σ_b) such that $\sigma_a, \sigma_b \in (\Sigma_1 \cap \Sigma_2)$. If G_1 and G_2 are time-separable for σ_a and σ_b , with transition costs such that $f_1(q_{a,1} \xrightarrow{\sigma_a} q'_{a,1}) \geq f_2(q_{a,2} \xrightarrow{\sigma_a} q'_{a,2})$ and $f_1(q_{b,1} \xrightarrow{\sigma_b} q'_{b,1}) \geq f_2(q_{b,2} \xrightarrow{\sigma_b} q'_{b,2})$, then $G^* = (G_1 \| G_2)^* = G_1^* \| G_2^*$.*

Proof Let there be a pair of events (σ_a, σ_b) that is sequentially dependent in G_1 and G_2 , where $\sigma_a, \sigma_b \in (\Sigma_1 \cap \Sigma_2)$. It follows from Proposition 3 Case i), that the pair of events (σ_a, σ_b) is also sequentially dependent in $G = G_1 \| G_2$. We wish to show that the cost of corresponding transitions with labels σ_a and σ_b are the same in $(G_1 \| G_2)^*$ as in $G_1^* \| G_2^*$. Letting f^* be the cost function for $(G_1 \| G_2)^*$ and $f_{1,2}^*$ the cost function for $G_1^* \| G_2^*$, we will show that $f^*((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q'_{a,2})) = f_{1,2}^*((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q'_{a,2}))$ and $f^*((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q'_{b,2})) = f_{1,2}^*((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q'_{b,2}))$, for all such transitions.

i) We begin with $(G_1 \| G_2)^*$. Consider transitions corresponding to σ_a $q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1}$ in G_1 and $q_{a,2} \xrightarrow{\sigma_a}_2 q_{a,2}'$ in G_2 . Noting that $\sigma_a, \sigma_b \in (\Sigma_1 \cap \Sigma_2)$ and following Definition 1, we have the transition $(q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q'_{a,2})$ in $G_1 \| G_2$, with cost $f((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q'_{a,2})) = \max(f_1(q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1}), f_2(q_{a,2} \xrightarrow{\sigma_a}_2 q'_{a,2}))$. By assumption that $f_1(q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1}) \geq f_2(q_{a,2} \xrightarrow{\sigma_a}_2 q'_{a,2})$, we then have that $f((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q'_{a,2})) = f_1(q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1})$. Following the same logic for σ_b transitions $q_{b,1} \xrightarrow{\sigma_b}_1 q'_{b,1}$ in G_1 and $q_{b,2} \xrightarrow{\sigma_b}_2 q_{b,2}'$ in G_2 , we have that $f((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q'_{b,2})) = f_1(q_{b,1} \xrightarrow{\sigma_b}_1 q'_{b,1})$.

Applying the lump operation of Definition 3 to generate $(G_1 \| G_2)^*$, lumping the cost from σ_b to σ_a , we have that $f^*((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q'_{a,2})) = f((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q'_{a,2})) + f((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q'_{b,2})) = f_1(q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1}) + f_1(q_{b,1} \xrightarrow{\sigma_b}_1 q'_{b,1})$, and $f^*((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q'_{b,2})) = 0$.

ii) Now we analyze $G_1^* \| G_2^*$. The costs associated with the transitions related to σ_a and σ_b , lumping the cost from σ_b to σ_a , for G_1^* are as follows: $f_1^*(q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1}) = f_1(q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1}) + f_1(q_{b,1} \xrightarrow{\sigma_b}_1 q'_{b,1})$ and $f_1^*(q_{b,1} \xrightarrow{\sigma_b}_1 q'_{b,1}) = 0$; similarly for G_2^* : $f_2^*(q_{a,2} \xrightarrow{\sigma_a}_2 q'_{a,2}) = f_2(q_{a,2} \xrightarrow{\sigma_a}_2 q'_{a,2}) + f_2(q_{b,2} \xrightarrow{\sigma_b}_2 q'_{b,2})$ and $f_2^*(q_{b,2} \xrightarrow{\sigma_b}_2 q'_{b,2}) = 0$.

Noting that $\sigma_a, \sigma_b \in (\Sigma_1 \cap \Sigma_2)$ and invoking Definition 1, the transition costs in $G_1^* \| G_2^*$ are: $f_{1,2}^*((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q'_{a,2})) = \max(f_1^*(q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1}), f_2^*(q_{a,2} \xrightarrow{\sigma_a}_2 q'_{a,2}))$, by assumption $f_1(q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1}) \geq f_2(q_{a,2} \xrightarrow{\sigma_a}_2 q'_{a,2})$ and $f_1(q_{b,1} \xrightarrow{\sigma_b}_1 q'_{b,1}) \geq f_2(q_{b,2} \xrightarrow{\sigma_b}_2 q'_{b,2})$, therefore, $f_1(q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1}) + f_1(q_{b,1} \xrightarrow{\sigma_b}_1 q'_{b,1}) \geq f_2(q_{a,2} \xrightarrow{\sigma_a}_2 q'_{a,2}) + f_2(q_{b,2} \xrightarrow{\sigma_b}_2 q'_{b,2})$, and thus, $f_{1,2}^*((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q'_{a,2})) = f_1^*(q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1}) = f_1(q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1}) + f_1(q_{b,1} \xrightarrow{\sigma_b}_1 q'_{b,1})$ and $f_{1,2}^*((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q'_{b,2})) = 0$.

- Comparing results from i) and ii), we have $f^*((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q'_{a,2})) = f_{1,2}^*((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q'_{a,2})) = f_1(q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1}) + f_1(q_{b,1} \xrightarrow{\sigma_b}_1 q'_{b,1})$ and $f^*((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q'_{b,2})) = f_{1,2}^*((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q'_{b,2})) = 0$.
- The same logic is valid if we lump the cost of σ_a to σ_b .
- Having demonstrated that the transitions have the same cost, in general, we have demonstrated that $G^* = (G_1 \| G_2)^* = G_1^* \| G_2^*$. $\square$

Proposition 5 Let there be two weighted automata $G_1 = (Q_1, \Sigma_1, \rightarrow_1, q_{0,1}, Q_{m,1}, f_1)$ and $G_2 = (Q_2, \Sigma_2, \rightarrow_2, q_{0,2}, Q_{m,2}, f_2)$ with $\sigma_a, \sigma_b \in \Sigma_1 \setminus \Sigma_2$. If the pair of events (σ_a, σ_b) is sequentially dependent and time-separable in G_1 , then $G^* = (G_1 \| G_2)^* = G_1^* \| G_2^*$.

Proof Let there be the pair of events (σ_a, σ_b) that are sequentially dependent in G_1 , where $\sigma_a, \sigma_b \in \Sigma_1 \setminus \Sigma_2$. We wish to show that the cost of corresponding transitions with labels σ_a and σ_b are the same in $(G_1 \| G_2)^*$ as in $G_1^* \| G_2^*$. Letting f^* be the cost function for $(G_1 \| G_2)^*$ and $f_{1,2}^*$ the cost function for $G_1^* \| G_2^*$, we will show that $f^*((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q'_{a,2})) = f_{1,2}^*((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q'_{a,2}))$ and $f^*((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q'_{b,2})) = f_{1,2}^*((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q'_{b,2}))$, for all such transitions.

i) We begin with $(G_1 \parallel G_2)^*$. Consider the transitions $q_{a,1} \xrightarrow{\sigma_a} q'_{a,1}$ and $q_{b,1} \xrightarrow{\sigma_b} q'_{b,1}$ in G_1 . Noting that $\sigma_a, \sigma_b \in \Sigma_1 \setminus \Sigma_2$ and following Definition 1, we have the transitions $(q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q_{a,2})$ and $(q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q_{b,2})$ in $G_1 \parallel G_2$, with costs $f((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q_{a,2})) = f_1(q_{a,1} \xrightarrow{\sigma_a} q'_{a,1})$ and $f((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q_{b,2})) = f_1(q_{b,1} \xrightarrow{\sigma_b} q'_{b,1})$. Applying the lump operation of Definition 3, lumping the cost of σ_b to σ_a , to generate $(G_1 \parallel G_2)^*$, we have $f^*((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q_{a,2})) = f_1(q_{a,1} \xrightarrow{\sigma_a} q'_{a,1}) + f_1(q_{b,1} \xrightarrow{\sigma_b} q'_{b,1})$ and $f^*((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q_{b,2})) = 0$.

ii) Now we analyze $G_1^* \parallel G_2^*$. The costs associated with the transitions of G_1^* maintaining the same order of lumping done before, are as follows: $f_1^*(q_{a,1} \xrightarrow{\sigma_a} q'_{a,1}) = f_1(q_{a,1} \xrightarrow{\sigma_a} q'_{a,1}) + f_1(q_{b,1} \xrightarrow{\sigma_b} q'_{b,1})$ and $f_1^*(q_{b,1} \xrightarrow{\sigma_b} q'_{b,1}) = 0$. Since $\sigma_a, \sigma_b \notin \Sigma_2$, and following Definition 1, we have $f_{1,2}^*((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q_{a,2})) = f_1^*(q_{a,1} \xrightarrow{\sigma_a} q'_{a,1}) = f_1(q_{a,1} \xrightarrow{\sigma_a} q'_{a,1}) + f_1(q_{b,1} \xrightarrow{\sigma_b} q'_{b,1})$ and $f_{1,2}^*((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q_{b,2})) = f_1^*(q_{b,1} \xrightarrow{\sigma_b} q'_{b,1}) = 0$.

- Comparing results from **i)** and **ii)**, we have $f^*((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q_{a,2})) = f_{1,2}^*((q_{a,1}, q_{a,2}) \xrightarrow{\sigma_a} (q'_{a,1}, q_{a,2})) = f_1(q_{a,1} \xrightarrow{\sigma_a} q'_{a,1}) + f_1(q_{b,1} \xrightarrow{\sigma_b} q'_{b,1})$ and $f^*((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q_{b,2})) = f_{1,2}^*((q_{b,1}, q_{b,2}) \xrightarrow{\sigma_b} (q'_{b,1}, q_{b,2})) = 0$.
- The same logic is valid if we lump the cost of σ_a to σ_b .
- Having demonstrated that the transitions have the same cost, in general, we have demonstrated that $G^* = (G_1 \parallel G_2)^* = G_1^* \parallel G_2^*$. $\square$

The conditions of Propositions 4 and 5 might seem restrictive, however, many systems are composed of plant components with disjoint event sets where all interaction takes place through specifications. In this case, the transitions within the specifications may be assumed to have zero cost with each event (and its cost) “owned” by a single plant component. In such a situation, the required conditions would be satisfied to distribute the lump operation over parallel composition as defined in Definition 1.

Figure 6 provides an example that illustrates Proposition 4. The automaton G is composed of two component automata, such that $G = G_1 \parallel G_2$. The pair of events (σ_a, σ_b) is sequentially dependent in G_1 and G_2 . For the transitions with labels σ_a and σ_b , each instance has greater cost in G_2 than in G_1 . As predicted, $G_1^* \parallel G_2^*$ results in the same automaton as $(G_1 \parallel G_2)^*$.

Now that we have shown it is possible to distribute the lump operation over parallel composition and that the partition automaton can be generated compositionally, we can combine these results to present the main theorem of this paper. This result requires that events shared between two weighted automata, G_1 and G_2 , cannot be hidden. Furthermore, if the shared events are sequentially dependent, then all transitions for each dependent pair of events must have greater or equal cost in one of the two automata. Under these conditions, the abstracted automaton $(G^*)^h$ can be generated compositionally $(G^*)^h \approx_{PO} (G_1^*)^h \parallel (G_2^*)^h$, without the need to first construct the full unabstracted automaton G . This significantly improves the scalability of the results from Hill and Lafortune (2016) by making it possible to generate the abstracted model, while avoiding the explosion of the state space that can arise from the construction of the global model G .

Fig. 6 Example: Distribution of lump operation over parallel composition

Theorem 3 Let there be two weighted automata $G_1 = (Q_1, \Sigma_1, \rightarrow_1, q_{0,1}, Q_{m,1}, f_1)$ and $G_2 = (Q_2, \Sigma_2, \rightarrow_2, q_{0,2}, Q_{m,2}, f_2)$. For each sequentially-dependent pair of events (σ_a, σ_b) , either let $\sigma_a, \sigma_b \in \Sigma_1 \setminus \Sigma_2$ or $\sigma_a, \sigma_b \in (\Sigma_1 \cap \Sigma_2)$, and let G_1 and G_2 be time-separable for σ_a and σ_b with costs: $f_1(q_{a,1} \xrightarrow{\sigma_a}_1 q'_{a,1}) \geq f_2(q_{a,2} \xrightarrow{\sigma_a}_2 q'_{a,2})$ and $f_1(q_{b,1} \xrightarrow{\sigma_b}_1 q'_{b,1}) \geq f_2(q_{b,2} \xrightarrow{\sigma_b}_2 q'_{b,2})$. Let G^* be the lumped automaton generated from $G = G_1 \parallel G_2$. If the partition automata are generated by priced-observation equivalent abstraction and $\Sigma^h \supseteq \Sigma_1 \cap \Sigma_2$, then $(G^*)^h \approx_{PO} (G_1^*)^h \parallel (G_2^*)^h$.

Proof By assumption, each arbitrary sequentially-dependent pair of events (σ_a, σ_b) with $\sigma_a, \sigma_b \in (\Sigma_1 \cap \Sigma_2)$ has corresponding transitions with costs that satisfy

$f_1(q_{a,1} \xrightarrow{\sigma_a} q'_{a,1}) \geq f_2(q_{a,2} \xrightarrow{\sigma_a} q'_{a,2})$ and $f_1(q_{b,1} \xrightarrow{\sigma_b} q'_{b,1}) \geq f_2(q_{b,2} \xrightarrow{\sigma_b} q'_{b,2})$. Invoking Proposition 4, we have $G^* = G_1^* \| G_2^*$. Similarly, each arbitrary sequentially-dependent pair of events (σ_a, σ_b) with $\sigma_a, \sigma_b \in \Sigma_1 \setminus \Sigma_2$, also provides that $G^* = G_1^* \| G_2^*$ by Proposition 5. Thus $(G^*)^h = (G_1^* \| G_2^*)^h$. Applying Theorem 2 then provides that for a priced-observation equivalent abstraction with $\Sigma^h \supseteq \Sigma_1 \cap \Sigma_2$, we have $(G^*)^h = (G_1^* \| G_2^*)^h \approx_{PO} (G_1^*)^h \| (G_2^*)^h$. $\square$

4 Computational examples

In order to illustrate the advantages of hierarchical planning using the compositional approach to abstraction proposed in this paper, we will apply the technique to the example of cooperative robot control introduced in Hill and Lafortune (2017).

In the approach of this paper, a planning technique is applied to the abstraction of a set of automata composed together. There are no requirements on how these automata are generated, but it is likely in many cases that these automata will capture the set of safe and nonblocking behaviors of the system, in other words, the system's supervised behavior. More specifically, the idea is to create a model for the problem by generating a classic supervisor based on the controllable and nonblocking sublanguage which provides the possible legal behaviors of the system. In the hierarchical planning proposed by Hill and Lafortune (2016), an abstraction of the supervisor is created based on their notion of cost equivalence, and this abstraction becomes the search space for the planning technique. A limitation with their work is that you still need to create the full model (supervisor) before generating the abstraction, which can be computationally prohibitive. The proposed approach of this paper makes it possible to create the abstracted model for hierarchical planning in a compositional manner. In order to apply this approach, we will first generate a set of modular supervisors, rather than generating the monolithic supervisor. In the following, the models and monolithic and compositional approaches to abstraction are described for our motivating example.

Let's first consider a scenario with two robots, A and B , four tasks, and four regions, as illustrated in Fig. 7.

Each robot is physically capable of performing only one task at a time. This behavior is captured for robot A by the automaton $R_{A_{tasks}}$ shown in Fig. 8a, where each task consists of a start event $a\#s$ and a finish event $a\#f$. The analogous model is built for robot B , $R_{B_{tasks}}$, where the start events have the form $b\#s$ and the finish events have the form $b\#f$. There is a requirement on ordering precedence between the tasks as illustrated by the automata $Spec_{tasks_{1,2}}$ and $Spec_{tasks_{3,4}}$, shown in Fig. 9. The automaton $Spec_{tasks_{1,2}}$ represents that task 1 must be performed prior to task 2 and by the same robot. This could represent, for example, that each odd-numbered task involves picking up an item at some location, while the corresponding even-numbered task involves dropping the item at a different location. Each transition for starting a task ($a\#s, b\#s$) possesses the cost associated with performing that task (distance to that location), while all transitions for finishing a task ($a\#f, b\#f$) have zero cost. Additionally, all start events are controllable and all finish events are uncontrollable.

We also model how the robots move between regions (numbered 5, 6, 7, 8), as well as the region in which each task is located. From its current position, a robot can move to any adjacent region, but it cannot move diagonally between regions. The automaton in Fig. 8b is for robot A , where events $a\#e$ signify the entry into a region. Likewise, an automaton

Fig. 7 Cooperative control of multiple robots: Scenario with 2 robots, 4 tasks, and 4 regions [(Hill and Lafortune 2017)]

is built for robot B , with a change in the initial state since robot B starts in region 8. As a restriction, it is specified we cannot have more than one robot in the same region at the same time, as modeled in Fig. 10. All entry events ($a\#e, b\#e$) are controllable and could be assigned a cost based on the distance from the preceding location to the point of region entry.

The plant model related to task completion is $G_{tasks} = R_{A_{tasks}} \parallel R_{B_{tasks}}$, and the plant model related to robot movement is $G_{regions} = R_{A_{geo}} \parallel R_{B_{geo}}$. The overall plant model is $G = G_{tasks} \parallel G_{regions}$. The full model for the specifications is $E = Spec_{1,2} \parallel$

Fig. 8 Models for Robot A

Fig. 9 Specification for task precedence

$Spec_{3,4} \parallel Spec_{regions}$. The desired behavior for the whole system is $K = L_m(G \parallel E)$. The monolithic supervisor is obtained by $S = SupC(K, L_m(G))$.

Fig. 10 Specification for region avoidance, $Spec_{regions}$

Table 1 Size of automata generated in the monolithic approach for a scenario of 2 robots, 4 tasks, and 4 regions

Monolithic Approach Step	# of states	# of transitions
$G = G_{tasks} \parallel G_{regions}$	400	2400
$K = G \parallel E$	672	2512
$S = SupC(K, L_m(G))$	672	2512
$S^{(a\#e, b\#e, a\#f, b\#f)}$	16	32

In the supervisor S , entry events will always follow a start event. Sub-trajectories through the supervisor automaton have the form that a task is started, some regions are possibly entered, then the task is completed. Such a pattern almost implies sequential dependence, except that the robot could take different paths to a task (it could enact different region entry events). If the entry events had cost zero and became candidates to be abstracted, it would greatly increase the potential for abstraction and would improve the decoupling between the high-level planning and the low-level planning. A reasonable approximation is to treat the entry events as having zero cost and treat the start events as having cost equal to the entire straight-line distance to the task. In such a case, the entry events become candidates to be abstracted while maintaining priced-observation equivalence and zero-cost reachability with respect to the original automaton.

Based on this approximation, we can consider finish and entry events, both having cost zero, to be candidates for abstraction. For an arbitrary automaton A , we will denote an abstraction A^h more specifically as $A^{(\sigma_1, \sigma_2, \dots, \sigma_n)}$ to signify that the events $(\sigma_1, \sigma_2, \dots, \sigma_n)$ have been hidden. In the monolithic approach, the high-level abstraction is generated from the monolithic supervisor $S = SupC(K, L_m(G))$ by abstracting all finish and entry events at once $S^h = S^{(a\#f, b\#f, a\#e, b\#e)}$.

In order to apply Theorem 2 to generate our high-level abstraction compositionally, we first generate a set of modular supervisors that meet all of the provided specifications for our system of robots. We will employ the approach of Hill and Tilbury (2008) to generate the modular supervisors incrementally. Consider the following steps as an example for the scenario with two robots, four tasks and four regions:

- $S_{avoid} = SupC(L_m(G_{regions} \parallel Spec_{regions}), L_m(G_{regions}))$;
- $S_{avoid}^h = S_{avoid}^{(a\#e, b\#e)}$;

Table 2 Size of automata generated in the compositional approach for a scenario of 2 robots, 4 tasks, and 4 regions

Compositional Approach Step	# of states	# of transitions
$G_{regions}$	16	96
S_{avoid}	12	56
G_{12}	25	16
S_{12}	48	136
G_{34}	36	108
S_{34}	32	56
$S_{avoid}^{(a\#e, b\#e, a\#f, b\#f)} \parallel S_{12}^{(a\#f, b\#f)}$	4	20
$S_{avoid}^{(a\#e, b\#e, a\#f, b\#f)} \parallel S_{12}^{(a\#f, b\#f)} \parallel S_{34}^{(a\#f, b\#f)}$	16	32

Table 3 Comparison between monolithic and compositional approaches for abstraction

Scenario	Biggest Step Monolithic		Biggest Step Compositional		Abstracted Supervisor	
	# states	# transitions	# states	# transitions	# states	# transitions
2R4T4R	672	2512	48	136	16	32
2R4T9R	4,032	22,656	72	400	16	32
2R8T9R	147,456	892,928	768	2,624	256	1,024
3R8T9R	495,432	3,708,432	4,725	23,220	625	3,000
3R10T9R	*	*	23,625	124,650	3,125	18,750

- $G_{12} = G_{tasks} \| S_{avoid}^h$;
- $S_{12} = SupC(L_m(G_{12} \| Spec_{tasks_{1,2}}), L_m(G_{12}))$;
- $G_{34} = S_{12}^{(a1f, a2f, b1f, b2f)}$;
- $S_{34} = SupC(L_m(G_{34} \| Spec_{tasks_{3,4}}), L_m(G_{34}))$.

The conjunction of these modular supervisors, $S_{avoid} \| S_{12} \| S_{34}$, is guaranteed by construction to meet all the safety specifications and to be nonblocking by Hill and Tilbury (2008). Therefore, for this example, we can consider that the overall supervisor S is composed of these modular supervisors. Now, given $S = S_{avoid} \| S_{12} \|$

S_{34} and based on Theorem 2, we have that the abstraction of the supervisor S can be performed compositionally (based on the transitivity of the equivalence). Thus, the high-level of the hierarchy can be built compositionally, resulting in $S^h \approx_{PO} S_{avoid}^{(a\#e, b\#e, a\#f, b\#f)} \| S_{12}^{(a\#f, b\#f)} \| S_{34}^{(a\#f, b\#f)}$.

For this example scenario, the size of necessary steps to create the abstracted upper level of the hierarchy are presented in Table 1 for the monolithic approach and in Table 2 for the compositional approach. The monolithic approach requires the creation of an automaton with 672 states and 2512 transitions, while in the compositional approach the largest automaton that is generated has 48 states and 136 transitions. This is a reduction of 93% in the number of states and 95% in the number of transitions. Both approaches result in an upper-level abstraction with 16 states and 32 transitions. Since all the events abstracted (entry and finish events) have cost zero and the abstractions have priced-observation equivalence, it is possible to solve the high-level control of the multiple robots for this scenario in the upper-level abstraction as proposed in Theorem 1.

Both approaches, monolithic and compositional, were applied to different scenarios of the cooperative robot control application proposed in Hill and Lafortune (2017), where the notation #R#T#R indicates the number of robots, number of tasks, and number of regions. These results are displayed in Table 3, which shows the size of the biggest automaton created during the monolithic and compositional approaches, as well the final abstracted supervisor size.

5 Conclusion

The work of Hill and Lafortune (2016) developed a hierarchical approach to planning in order to reduce the complexity of planning operations, specifically, to accelerate the on-line planning that may need to occur in reaction to unpredictable events or new information.

However, the step of first generating the global supervisor before the necessary abstraction can be applied restricts the size of systems that can be addressed by this approach. The present paper is an extension of the conference paper (Vilela and Hill 2020), which demonstrates that under certain conditions it is possible to generate the abstraction of a system compositionally using the new notion of priced-observation equivalence. This equivalence guarantees that an optimal plan found in the abstraction is also optimal in the global supervisor as proven by Theorem 1 of Hill and Lafortune (2016).

In an effort to improve the amount of reduction obtained by the abstraction, (Hill and Lafortune 2016) proposed an operation that creates transitions with artificial zero cost, if they are part of a sequentially-dependent ordering of events, without changing the overall cost of a marked path. However, there are cases where there is a sequential dependence relation between a pair of events, but they do not follow a specific ordering as required by Hill and Lafortune (2016). In this paper, a relaxation of the sequential dependence property was proposed to include these cases. It was shown that the overall cost of marked paths in the resulting lumped automaton based on the relaxed sequential dependence is kept unaltered, maintaining the optimal relation between upper and lower levels proposed by Hill and Lafortune (2016).

The present paper also demonstrates that the relaxation of the sequential dependence concept does not affect the results of Vilela and Hill (2020). Specifically, it is proven that under this relaxation, it is still possible to generate the abstraction of a system compositionally.

The results compiled in Table 3 for different scenarios of the cooperative robot control example of Hill and Lafortune (2017) demonstrates the improvement in scalability achieved by the compositional approach proposed by this paper as compared to the monolithic approach to hierarchy generation employed by Hill and Lafortune (2016). For a scenario with two robots, four tasks, and four regions, employing the compositional approach provides a reduction of 93% in the number of states and 95% in the number of transitions of the largest automaton built in the process of generating the abstraction as compared to using the monolithic approach. For large scenarios, like the one consisting of three robots, ten tasks, and nine regions, the synthesis of a global supervisor is prohibitively large, precluding the possibility of even applying the monolithic approach of Hill and Lafortune (2016). The proposed compositional approach, however, is able to address the scenario and the largest automaton that it needs to construct has 23,625 states and 124,650 transitions.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Asarin E, Maler O (1999) As soon as possible: Time optimal control for timed automata. In: Vaandrager FW, van Schuppen JH (eds) Hybrid Systems: Computation and Control. Springer, Berlin, pp 19–30
- Brandin BA, Wonham WM (1994) Supervisory control of timed discrete-event systems. IEEE Trans Autom Control 39(2):329–342

- Bravo HJ, Pena PN, Alves LV, Takahashi RH (2018) Factorization-based approach for computing a minimum makespan controllable sublanguage. *IFAC-PapersOnLine* 51(7):19–24. 14th IFAC Workshop on Discrete Event Systems WODES 2018
- Cassandras C, Lafortune S (2008) Introduction to discrete event systems, 2nd edn. Springer, New York
- Dulce-Galindo JA, Santos MA, Raffo GV, Pena PN (2019) Autonomous navigation of multiple robots using supervisory control theory. In: 2019 18th European Control Conference (ECC), pp 3198–3203
- Fabre E, Jezequel L (2009) Distributed Optimal Planning: An Approach by Weighted Automata Calculus. 2009 IEEE Conference on Decision and Control and 28th Chinese Control Conference (section III)
- Ghahlab M, Nau D, Traverso P (2004) Automated Plannig: theory and practice, 1st edn. Elsevier, San Francisco
- Hagebring F, Lennartson B (2018) Compositional optimization of discrete event systems. In: 2018 IEEE 14th International Conference on Automation Science and Engineering - CASE, pp 849–856
- Hill R, Tilbury D (2008) Incremental hierarchical construction of modular supervisors for discrete-event systems. *Int J Control* 81:1364–1381
- Hill RC, Lafortune S (2016) Planning under abstraction within a supervisory control context. In: 2016 IEEE 55th Conference on Decision and Control (CDC), pp 4770–4777
- Hill RC, Lafortune S (2017) Scaling the formal synthesis of supervisory control software for multiple robot systems. In: 2017 American Control Conference (ACC), pp 3840–3847
- Huang J, Kumar R (2007) Optimal nonblocking directed control of discrete event systems. In: 2007 American Control Conference, pp 4285–4290
- Hyun-Wook J, Jae-Ho A, Jun-Sang P, Jun-Han O, Lim J (2010) Task planning for service robots with optimal supervisory control. In: 2010 IEEE Conference on Robotics, Automation and Mechatronics, pp 303–308. <https://doi.org/10.1109/RAMECH.2010.5513172>
- Pena PN, Costa TA, Silva RS, Takahashi RHC (2016) Control of Flexible Manufacturing Systems under model uncertainty using Supervisory Control Theory and evolutionary computation schedule synthesis. *Inf Sci* 329:491–502
- Qiang Y (2012) Intelligent Planning: A Decomposition and Abstraction Based Approach, 1st edn. Springer Publishing Company, Berlin
- Sengupta R, Lafortune S (1993) A deterministic optimal control theory for discrete event systems. In: Proceedings of 32nd IEEE Conference on Decision and Control, vol 2, pp 1182–1187
- Su R (2012) Abstraction-based Synthesis of Timed Supervisors for Time-Weighted Systems. In: Proceedings of the 11th International Workshop on Discrete Event Systems (WODES 2012), pp 128–134
- Su R (2013) Coordinated distributed time optimal supervisory control. In: 2013 American Control Conference, pp 905–910
- Su R, van Schuppen JH, Rooda JE (2012) The synthesis of time optimal supervisors by using heaps-of-pieces. *IEEE Trans Autom Control* 57(1):105–118
- van Putten BJC, van der Sadden B, Reniers M, Voeten J, Schiffelers R (2020) Supervisor synthesis and throughput optimization of partially-controllable manufacturing systems. *Discrete Event Dynamic Systems*
- Vilela J, Hill R (2020) A compositional approach to abstraction for planning problems. In: 2020 Workshop on Discrete Event Systems (WODES)
- Ware S, Su R (2017) Time optimal synthesis based upon sequential abstraction and its application to cluster tools. *IEEE Trans Autom Sci Eng* 14(2):772–784

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Juliana Vilela received the bachelor degree in Control and Automation Engineering in 2011 and a M.S. degree in Electrical Engineering in 2016 from Federal University of Minas Gerais, Belo Horizonte, Brazil. She is currently working towards the Ph.D degree in Electrical Engineering at the University of Detroit Mercy. Her current research interests are in the areas of Discrete Event Systems, Supervisory Control Theory, Planning, Optimization and Robotics.

Richard Hill is a Professor of Mechanical Engineering and the Assistant Dean for Research and External Initiatives in the College of Engineering & Science at University of Detroit Mercy. Dr. Hill received the B.S. degree in Mechanical Engineering, *summa cum laude*, from the University of Southern California in 1998, and the M.S. degree in Mechanical Engineering from the University of California, Berkeley in 2000. From 2000 to 2002, he worked at Lockheed Martin Corporation on satellite attitude determination and control. He then spent two years as a high school math and science teacher. In 2008 he received the Ph.D. degree in Mechanical Engineering and the M.S. degree in Applied Mathematics from the University of Michigan, Ann Arbor. In 2008 Dr. Hill joined the faculty of the Mechanical Engineering Department at Detroit Mercy. His research focuses on scaling and optimizing supervisory control logic for complex discrete-event systems through modular and hierarchical techniques. Dr. Hill also has interest in vehicle control applications, engineering education, and diversifying the STEM pipeline.