Theoretical Computer Science 274 (2002) 117–149

Theoretical Computer Science

www.elsevier.com/locate/tcs

Branching time controllers for discrete event systems

P. Madhusudan[a], P.S. Thiagarajan[b,*,1]

[a]*Institute of Mathematical Sciences, C.I.T. Campus, Taramani, Chennai, India*
[b]*Chennai Mathematical Institute, 92 G.N. Chetty Road, T. Nagar, Chennai, India*

Abstract

We study the problem of synthesizing controllers for discrete event systems in a branching time framework. We use a class of labelled transition systems to model both plants and specifications. We use first simulations and later bisimulations to capture the role of a controller; the controlled behaviour of the plant should be related via a simulation (bisimulation) to the specification. For both simulations and bisimulations we show that the problem of checking if a pair of finite transition systems – one modelling the plant and the other the specification – admits a controller is decidable in polynomial time. We also show that the size of the controller, if one exists, can be bounded by a polynomial in the sizes of the plant and the specification and can be effectively constructed in polynomial time. Finally, we prove that in the case of simulations, the problem of checking for the existence of a controller is undecidable in a natural concurrent setting. © 2002 Published by Elsevier Science B.V.

Keywords: Discrete-event systems; Controller synthesis; Supervisor synthesis; Simulations; Bisimulations; Asynchronous transition systems

1. Introduction

We study the problem of synthesizing controllers for discrete event systems. In informal terms, one is given an open discrete event system called a plant which consists of a system and its environment. One then specifies the desired patterns of interaction between the system and its environment. The problem then is to find a controller which will restrict the behaviour of the plant in such a way that the controlled behaviour meets the specification. Two characteristic features of the controller are that it is allowed to restrict only the actions of the system – and not those of the environment – and that

* Corresponding author.
E-mail address: pst@cmi.ac.in (P.S. Thiagarajan).
[1] This research was sponsored by the IFCPAR project 1502-1.

it should not introduce any fresh deadlocks. Typical questions that arise are

- Given finite descriptions of the plant and the specification is it decidable that there exists a controller ?
- In case there is a controller is it always the case that there is a finite controller?
- How big need the controller be – in case it exists – relative to the sizes of the plant and the specification?

A substantial amount of knowledge is available about this problem in the linear time framework. Here the behaviour of the plant will consist of L_P, a suitable collection of (finite or infinite) sequences. One then specifies the desired behaviour by another collection of sequences L_S. The problem then is to come up with a controller such that $L_{PC} \subseteq L_S$ where L_{PC} is the constrained language generated by the plant–controller combination.

This line of work goes back to a realization problem formulated by Church [6], later solved elegantly by Büchi and Landweber [5]. During the past decade there has been a vigorous revival of this area both from computer science and control-theoretic perspectives. For the computer science literature we refer the reader to [27, 28] and the references therein. As for the control-theoretic flavoured works, starting from [29], various problems associated with partial observability, controllability and hierarchical control have been addressed as evidenced in [14, 13, 15, 32].

In the present work, the key point of departure is that we study the controller synthesis problem in a branching time setting. Our main motivation is to admit specification mechanisms that are more flexible than the automata-theoretic means adopted in the linear time framework. We uniformly describe both plants and specifications as certain kinds of labelled transition systems. We then advocate the use of simulations and bisimulations to capture the requirement that the plant–controller combination meets its specification. As a result, behavioural properties that can be only stated in a branching time setting become available as specifications (see [20]). A typical example of such a property is the existence of a home state: there is a home state H which can be (potentially) reached from every intermediate state of every computation.

As for related work, the synthesis problem has been studied in a branching time setting using the failure semantics model of processes [25, 26]. A pre-order relates the behaviour of the plant–controller to the specification. However their setup is very different. In their setting, the nondeterminism arises due to abstraction and not due to the hiding of the environment's actions. Consequently, their controllers cannot distinguish between the nondeterministic choices made in the plant. In our setting the nondeterminism (on the labels of events) is purely due to the hiding of the environment's responses and the controller can discern between the nondeterministic choices made. A nice feature of [26] is that it deals with partial descriptions via the use of internal events. Extension of our work to handle partial descriptions is yet to be achieved. Yet another piece of related work is [1] where the branching time temporal logic CTL is used for specifications. The notion of a controller is however quite weak in that controllers are required to be memoryless. Further, the emphasis is on complexity-theoretic lower and upper bounds.

There is a neighbouring body of work (see for instance [9, 18]) which has a similar flavour as the controller synthesis problem and uses techniques similar to those we discuss in this paper. This body of work has to do with equation solving in a process algebraic domain. The simplest problem setting is one where one is given a system A and a specification B both presented as terms in a process algebra, say CCS. The problem is to come up with a CCS term X such that $A|X$ is bisimilar to B. To consider an extreme example, suppose A is the process *nil* which does nothing. Then $X = B$ will be accepted as solution to the equation $A|X = B$. Thus, the crucial difference between the work reported here and the work on equation solving in process algebras is that our controllers – unlike the unknown term X in the process algebra setting – can only restrict the behaviour of the plant; it is not allowed to contribute any new behavioural possibilities.

Finally, the sequence of results concerning module checking reported in [16, 17] has a bearing on our work. In particular, it suggests an important and nontrivial extension. It will be convenient to discuss this in more detail in the concluding section.

In the next section we formulate the model of a plant using transition systems with two layers of labelling on the transitions. This turns out to be a convenient way of capturing the usual two-person game associated with the plant as well as the plant–controller interaction. We use the same class of transition systems to capture specifications. We then define simulations which are behaviour preserving homomorphisms, in the usual way. A controller is then required to restrict the system's actions so that the restricted behaviour of the plant can be related to the specification via a simulation. We advocate simulations because they are a good starting point for this study of branching time specifications. They can be used to capture restricted kinds of safety properties. For a detailed survey of simulations as specifications we refer the reader to the paper by Lynch and Vaandrager [20].

An important lesson derived from existing literature is that a richer class of controllers can be obtained by allowing the controller to make use of memory of the past to achieve its goal. Hence, we demand that a controller should be such that the *unfolding* of the plant–controller combination is related to the (unfolding of the) specification via a simulation. Clearly, the set of possible domains of such simulations is an infinite collection of infinite objects even when both the plant and specification are finite objects. Consequently the task of deciding the existence of a controller is not trivial.

In Sections 3 and 4 we show that the problem of deciding if a pair of finite systems (a plant and a specification) admits a controller is decidable in time which is polynomial in the sizes of the plant and specification. We also show that the size of the controller, whenever one exists, can be bounded from above by a similar polynomial. A point worth noting here is our transition systems are deterministic with respect to an alphabet of events. But the events will have an additional layer of action labels and the simulations are required to preserve only action labels. Consequently the domain of a simulation, relative to the action labels will be, in all nontrivial instances, nondeterministic.

In Section 5 we extend the techniques of the previous two sections to tackle the case of bisimulations. To our knowledge, bisimulations have never been considered as a specification mechanism in the supervisory control problem, though it has been used as a technique to solve the classical controller synthesis problem [3]. Surprisingly, the time complexity and the size of the controller (when one exists) still have polynomial upper bounds. It turns out that a crucial computational step in the decision procedure can be efficiently reduced to a maximal matching problem which is known to be solvable in polynomial time [7].

In Section 6 we enrich our transition systems with some concurrency information in a standard way [31]. We then show that in this richer setting, the problem of determining if there is a controller such that the unfolding of the plant–controller combination can be related to the specification via a simulation is undecidable. (The undecidability result reported in [28] is quite far removed from the present setting.) At present we do not know if our undecidability result goes through in the presence of bisimulations, though a recent result [11] suggests that this might be the case, as we will explain in the conclusion.

2. The model

It will be convenient to work with deterministic transition systems that have an additional layer of labelling. Through the rest of the paper we fix a finite set of labels Σ and let a, b range over Σ.

Definition 2.1. A Σ-labelled deterministic transition system is a structure $TS = (Q, E, T, q_{\text{in}}, \varphi)$ where

- Q is a set of states.
- E is a set of events.
- $T \subseteq Q \times E \times Q$ is a deterministic transition relation. In other words, if $(q, e, q') \in T$ and $(q, e, q'') \in T$ then $q' = q''$.
- $q_{\text{in}} \in Q$ is the initial state.
- $\varphi : E \longrightarrow \Sigma$ is a labelling function.

We use T to denote the set of transitions instead of the usual notation $\longrightarrow$ because the simulation maps we consider will operate on both states and transitions. Let $t = (q, e, q') \in T$. We will often write $q \xrightarrow{e} q'$ instead of $(q, e, q') \in T$. Sometimes we shall write $q \xrightarrow[a]{e} q'$ to indicate that $\varphi(e) = a$. Abusing notation $\varphi(ev(t))$ will be shortened to $\varphi(t)$. In all such cases the concerned transition system will be clear from the context.

From now, we shall refer to Σ-labelled deterministic transition systems as just transition systems.

Let $TS = (Q, E, T, q_{\text{in}}, \varphi)$ be a transition system. When viewed as the model of a system–environment combination, E will represent the environment actions and Σ the actions of the system. The occurrence of the transition $q \xrightarrow[a]{e} q'$ is to be viewed as the system offering to perform an a-action and the environment choosing the specific (a-labelled) event e as its matching response. There could be more than one a-labelled event enabled at q for the environment to choose from. We note also that it could be the case that $q \xrightarrow[a]{e} q'$ and $q \xrightarrow[b]{e'} q'$. Thus, the environment could choose the same response – in terms of the change produced in the global state – to two different actions a and b of the system. This way of describing the system–environment interaction is taken from [2]. In the present setting this will be easier to work with than the usual one in which the system moves and environment moves explicitly alternate [30].

We shall model both plants and specifications as transition systems. The controlled behaviour of a plant will be related to its specification by a simulation.

Definition 2.2. Let $TS_p = (Q_p, E_p, T_p, q^p_{\text{in}}, \varphi_p)$ and $TS_s = (Q_s, E_s, T_s, q^s_{\text{in}}, \varphi_s)$ be a pair of transition systems. Then a simulation f from TS_p to TS_s – denoted $f : TS_p \to TS_s$ – is a map $f : Q_p \cup T_p \to Q_s \cup T_s$ with $f(Q_p) \subseteq Q_s$ and $f(T_p) \subseteq T_s$ such that the following conditions are satisfied:

(i) $f(q^p_{\text{in}}) = q^s_{\text{in}}$.
(ii) Suppose $t = (q_1, e, q_2) \in T_p$ and $f(t) = (q'_1, e', q'_2)$. Then $f(q_1) = q'_1$ and $f(q_2) = q'_2$ and $\varphi_p(e) = \varphi_s(e')$.

Thus a simulation is just a structure preserving homomorphism. Given two transition systems TS_1 and TS_2 we will say that TS_1 and TS_2 are isomorphic in case there is a simulation $f : TS_1 \to TS_2$ such that $f : Q_1 \cup T_1 \to Q_2 \cup T_2$ is a bijection with $Q_i(T_i)$ being the set of states (transitions) of TS_i for $i = 1, 2$.

The notion of the controlled behaviour of a plant meeting its specification via a simulation will be defined at the level of unfoldings. As we point out later this will permit a larger class of contollers.

Definition 2.3. Let $TS = (Q, E, T, q_{\text{in}}, \varphi)$ be a transition system. Then $\mathscr{U}f(TS)$, the unfolding of TS is the structure $\widehat{TS} = (\hat{Q}, \hat{E}, \hat{T}, \hat{q}_{\text{in}}, \hat{\varphi})$ where $\hat{Q} \subseteq Q \times E^*$, $\hat{E} \subseteq E$ and $\hat{T} \subseteq \hat{Q} \times \hat{E} \times \hat{Q}$ are the least sets satisfying

(i) $(q_{\text{in}}, \varepsilon), \in \hat{Q}$.
(ii) Suppose $(q, \sigma) \in \hat{Q}$ and $(q, e, q') \in T$. Then $(q', \sigma e) \in \hat{Q}$, $e \in \hat{E}$ and $((q, \sigma), e, (q', \sigma e)) \in \hat{T}$.

Further, $\hat{q}_{\text{in}} = (q_{\text{in}}, \varepsilon)$ and $\hat{\varphi}$ is φ restricted to $\hat{E}$.

It is easy to check that $\widehat{TS}$ is a deterministic Σ-labelled transition system. We could have defined $\hat{Q}$ in terms of E^* alone but the present formulation will be easier to work with.

Finally, the controlled behaviour of a plant will be obtained by taking the (synchronized) product of the plant and a controller.

Definition 2.4. Let $TS_i=(Q_i,E_i,T_i,q^i_{\text{in}},\varphi_i)$, $i=1,2$ be a pair of transition systems. Then the product of TS_1 and TS_2 – denoted $TS_1\|TS_2$ – is the structure $TS=(Q,E,T,q_{\text{in}},\varphi)$ where $Q=Q_1\times Q_2$ and $E=E_1\cup E_2$ and T is the least subset of $Q\times E\times Q$ satisfying

- Suppose $(q_1,q_2)\in Q$ and $(q_1,e,q'_1)\in T_1$ with $e\notin E_2$. Then $((q_1,q_2),e,(q'_1,q_2))\in T$.
- Suppose $(q_1,q_2)\in Q$ and $(q_2,e,q'_2)\in T_2$ and $e\notin E_1$. Then $((q_1,q_2),e,(q_1,q'_2))\in T$.
- Suppose $(q_1,q_2)\in Q$ and $(q_1,e,q'_1)\in T_1$ and $(q_2,e,q'_2)\in T_2$ with $\varphi_1(e)=\varphi_2(e)$. Then $((q_1,q_2),e,(q'_1,q'_2))\in T$.

Further, $q_{\text{in}}=(q^1_{\text{in}},q^2_{\text{in}})$ and $\varphi:E_1\cup E_2\to\Sigma$ is given by $\varphi(e)=\varphi_1(e)$ if $e\in E_1$ and $\varphi(e)=\varphi_2(e)$ if $e\in E_2\backslash E_1$.

Again it is easy to check that $TS_1\|TS_2$ is also a deterministic Σ-labelled transition system. We are now ready to define controllers. As it will turn out, the plant–controller interaction will be a much tighter version of the product operation.

In defining the notion of a controller and elsewhere we will make use of the notion of reachable states. In other words, given a transition system $TS=(Q,E,T,q_{\text{in}},\varphi)$ we will say $q\in Q$ is reachable from q_{in} if $q=q_{\text{in}}$ or there exists a non-null sequence of states $q_0q_1\dots q_n$ with $q_0=q_{\text{in}}$ and $q_n=q$ and for $0\leqslant i<n$, $\exists e.\ (q_i,e,q_{i+1})\in T$.

Definition 2.5. Let $TS_x=(Q_x,E_x,T_x,q^x_{\text{in}},\varphi_x)$, $x\in\{p,c,s\}$ be three transition systems. Then TS_c is a controller for the pair (TS_p,TS_s) iff the following conditions are satisfied: Let $TS_p\|TS_c=(Q,E,T,q_{\text{in}},\varphi)$:

(CT1) $E_c=E_p$ and $\varphi_c=\varphi_p$.

(CT2) Suppose (q^1_p,q^1_c) is reachable from the initial state in $TS_p\|TS_c$ and $((q^1_p,q^1_c),e,(q^2_p,q^2_c))\in T$ and $(q^1_p,e',q^3_p)\in T_p$ with $\varphi_p(e)=\varphi_p(e')$. Then there exists $q^3_c\in Q_c$ such that $((q^1_p,q^1_c),e',(q^3_p,q^3_c))\in T$ (and hence $(q^1_c,e',q^3_c)\in T_c$).

(CT3) Suppose (q^1_p,q^1_c) is reachable from the initial state in $TS_p\|TS_c$ and $(q^1_p,e,q^2_p)\in T_p$. Then there exists $e'\in E_p$ and $q^3_p\in Q_p$ and $q^3_c\in Q_c$ such that $((q^1_p,q^1_c),e',(q^3_p,q^3_c))\in T$.

(CT4) There is a simulation from $\mathscr{U}f(TS_p\|TS_c)$ to $\mathscr{U}f(TS_s)$.

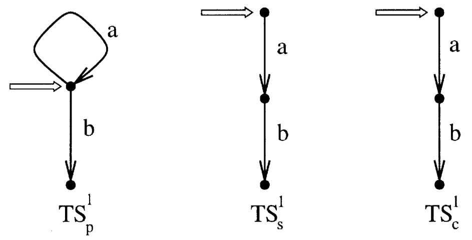
Condition (CT1) demands that the plant and the controller be tightly coupled. There are no "autonomous" transitions either for the plant or for the controller. Condition (CT2) says that TS_c should restrict only the system moves. If at a reachable state it permits one a-move then it should permit all a-moves. Condition (CT3) requires that the controller should be nonblocking. Stated diferently, the controller should not introduce any new deadlocks in the constrained plant behaviour. This condition also ensures that the problem does not degenerate, as otherwise there is always a controller which restricts *all* system moves and satisfies the specification.

The role of (CT4) should be clear. It says that the specification must be able to simulate the controlled plant. This basically means that we can cater for simple safety properties. We could have defined the simulation direction the other way, i.e. demand that the controlled plant must be able to simulate the specification. This will be a natural way to capture liveness properties. However, in the controller synthesis problem for such specifications, the notion becomes very weak because the controller will have no useful role to play. It is easy to see that if the plant does not satisfy the specification then no pruning of its behaviour (by a controller) will satisfy it.

In formulating (CT4), we could have used TS_s instead of $\mathcal{U}f(TS_s)$. The choice of the latter is for the sake of uniformity. We note that due to the deterministic event-based transition relation, the controller can record the history of the plant as a sequence of events. It can then use this record to determine the current state of plant and act accordingly.

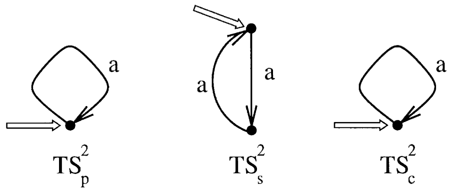
We have defined the goal of the controller to be able to restrict the plant such that there is a simulation function from the unfolding of the plant–controller combination to the unfolding of the specification. We could have instead required that there be *simulation relation* between the plant–controller pair (not its unfolding) and the specification. Though this would have been the more conventional route to take, we have chosen to take the present route because we feel that it is more transparent, especially in bringing out the role of the memory used by the controller. Moeover, our notion extends naturally to the concurrent setting considered in Section 6. In this extended setting the existence of a simulation function between the unfoldings of two transition systems does not imply the existence of a corresponding simulation relation between the two transition systems.

Example 2.6.



It is easy to see that TS^1_c is a controller for the pair (TS^1_p, TS^1_s).

Example 2.7.



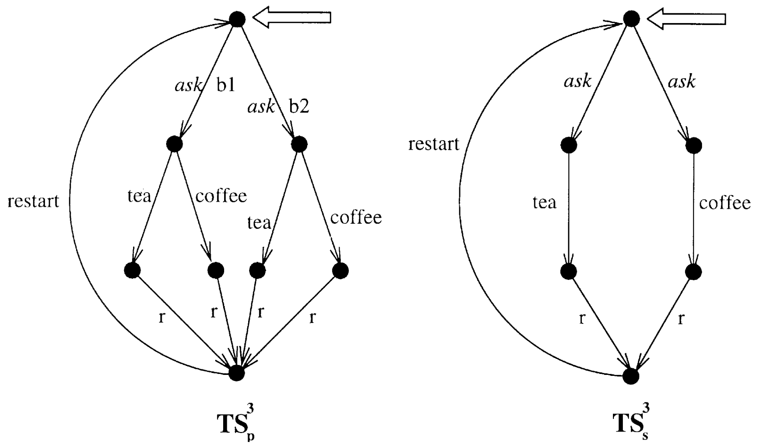
Again it is easy to see that TS_c^2 is a "trivial" controller for the pair (TS_p^2, TS_s^2). The point here is that it will not be a controller, if in Definition 2.5, we had replaced (CT4) by

(CT4′) There is a simulation from $TS_p \| TS_c$ to TS_s.

Thus, demanding a simulation map at the level of unfoldings admits a larger class of controllers in general.

The following example will illustrate the branching nature of the specification. The plant TS_p^3 is a vending machine which first asks the user to press a button $b1$ or $b2$. Then it can serve either tea or coffee, and reset (the event r) and restart. Events $b1$ and $b2$ are thus labelled with *ask* – the other events are pure plant moves and they are labelled using the identity function. In the specification TS_s, we show only the labels on the events – in this setting, the specification demands that the button which is pressed must determine whether coffee or tea is served. It must not be the case that after a button is pressed, there is a possibility of both coffee and tea being served. However, it cannot demand that the user can get tea if he/she wants tea. We can demand such a specification in the bisimulation setting, which we will discuss in Section 5.

Example 2.8.



A valid controller has to just disable either coffee or tea after $b1$ and $b2$. Note that it cannot disable both tea and coffee nor can it leave both enabled. It is easy to see that there is no *minimally restrictive controller*, in the sense of one which allows maximum number of event sequences. This, in fact, also shows that such specifications cannot be stated in the Ramadge–Wonham framework [29], as in their setting minimally restrictive controllers always exist.

We conclude this section by stating one of our main results. In doing so and elsewhere we will say that the transition system TS is finite in case both Q and E are finite sets. In case TS is finite, its size – denoted $|TS|$ – is defined to be $|Q| + |E|$.

Theorem 2.9. *Let (TS_p, TS_s) be a pair of finite transition systems and $m = \max(|TS_p|, |TS_s|)$. Then the question whether there exists a controller for (TS_p, TS_s) can be decided in time polynomial in m. Moreover, if a controller exists, we can construct one whose size is bounded by a polynomial in m. This construction also takes time bounded by a polynomial in m.*

3. A good subgraph characterization

Our goal here is to characterize controllers in terms of objects called good subgraphs. This will easily lead to a proof of Theorem 2.9. Given a pair of finite transition systems (TS_p, TS_s) we shall form an edge-labelled directed graph G_{ps} which will be a restricted product of TS_p and TS_s. We will then show that (TS_p, TS_s) admits a controller iff G_{ps} contains a good subgraph possessing certain closure properties.

Through the rest of the section we fix a pair of finite transition systems (TS_p, TS_s) with $TS_x = (Q_x, E_x, T_x, q_{\text{in}}^x, \varphi_x)$, $x \in \{p, s\}$. Then the edge-labelled directed graph $G_{ps} = (X, \rightarrow)$ is given by

- $X = Q_p \times Q_s$,
- $\rightarrow \subseteq X \times (E_p \times E_s) \times X$ is defined as $(q_p, q_s) \xrightarrow{(e,e')} (q_p', q_s')$ iff $q_p \xrightarrow{e} q_p'$ in TS_p and $q_s \xrightarrow{e'} q_s'$ in TS_s and $\varphi_p(e) = \varphi_s(e')$.

We shall say that $G = (Y, \Rightarrow)$ is a subgraph of G_{ps} iff $Y \subseteq X$ and $\Rightarrow \subseteq \rightarrow \cap (Y \times (E_p \times E_s) \times Y)$.

Definition 3.1. Let $G = (Y, \Rightarrow)$ be a subgraph of G_{ps}. Then G is said to be *good* iff it satisfies the following conditions.

(G1) $(q_{\text{in}}^p, q_{\text{in}}^s) \in Y$.

(G2) Suppose $(q_p, q_s) \overset{(e,e')}{\Longrightarrow} (q_p', q_s')$ and $q_p \xrightarrow{e_1} q_p^1$ with $\varphi_p(e) = \varphi_p(e_1)$, then $\exists e_1'$. $(q_p, q_s) \overset{(e_1,e_1')}{\Longrightarrow} (q_p^1, q_s^1)$ in G

(G3) Suppose $(q_p, q_s) \in Y$ and there exists $q_p \xrightarrow{e} \widehat{q}_p$ in TS_p. Then there exists $q_p' \in Q_p$, $q_s' \in Q_s$, $e_1 \in E_p$, $e_1' \in E_s$ such that $(q_p, q_s) \overset{(e_1,e_1')}{\Longrightarrow} (q_p', q_s')$ in G.

The next sequence of results will assume these notions.

Lemma 3.2. *Suppose TS_c is a controller for (TS_p, TS_s). Then G_{ps} contains a good subgraph.*

Proof. Let $TS_c = (Q_c, E_c, T_c, q_{\text{in}}^c, \varphi_c)$ and $TS = \mathcal{U}f(TS_p \| TS_c) = (Q, E, T, q_{\text{in}}, \varphi)$. Let f be a simulation from TS to $\mathcal{U}f(TS_s)$. We now define the subgraph $(Y, \Longrightarrow)$ of G_{ps} induced by f as follows:

- $(q_p, q_s) \in Y$ iff there exists $((q_p, q_c), \sigma) \in Q$ such that $f((q_p, q_c), \sigma) = (q_s, \sigma')$ for some σ' in E_s^*

- $(q_p, q_s) \overset{(e,e')}{\Longrightarrow} (q'_p, q'_s)$ iff there exists $t = ((q_p, q_c), \sigma), e, ((q'_p, q'_c), \sigma e)) \in T$ such that $f(t) = ((q_s, \sigma'), e', (q'_s, \sigma' e'))$ for some σ' in E_s^*.

We claim that $(Y, \Longrightarrow)$ is a good subgraph of G_{ps}. Property (G1) follows from $f((q_p^{\text{in}}, q_c^{\text{in}}), \varepsilon) = (q_s^{\text{in}}, \varepsilon)$ which in turn implies $(q_p^{\text{in}}, q_s^{\text{in}}) \in Y$. To verify (G2), assume that $(q_p, q_s) \overset{(e,e')}{\Longrightarrow} (q'_p, q'_s)$ and that $q_p \xrightarrow{e_1} q_p^1$ in TS_p with $\varphi_p(e) = \varphi_p(e_1)$. From the definition of $\Longrightarrow$, it follows that there exists $t = (((q_p, q_c), \sigma), e, ((q'_p, q'_c), \sigma e)) \in T$ with $f(t) = ((q_s, \sigma'), e', (q'_s, \sigma' e'))$. Clearly (q_p, q_c) is a reachable state in $TS_p \| TS_c$ since $((q_p, q_c), \sigma)$ is a state of $\mathcal{U}f(TS_p \| TS_c)$. From the existence of t it follows that $((q_p, q_c), e, (q'_p, q'_c))$ is a transition in $TS_p \| TS_c$. But then $q_p \xrightarrow{e_1} q_p^1$ is a transition in TS_p with $\varphi_p(e) = \varphi_p(e_1)$. Hence, the fact that TS_c is a controller ensures that $((q_p, q_c), e_1, (q_p^1, q_c^1))$ is a transition in $TS_p \| TS_c$ for some q_c^1 in Q_c. This in turn implies that $t_1 = (((q_p, q_c), \sigma), e_1, ((q_p^1, q_c^1), \sigma e_1)) \in T$. By the definition of a simulation we have $f((q_p, q_c), \sigma) = (q_s, \sigma')$. Let $f(t_1) = ((q_s, \sigma'), e'_1, (q_s^1, \sigma' e'_1))$. Then by the definition of $\Longrightarrow$ we are assured that $((q_p, q_s), (e_1, e'_1), (q_p^1, q_s^1)) \in \Longrightarrow$. This establishes (G2).

In a similar fashion, we can use property (CT3) in the definition of a controller to establish (G3). □

As a first step towards proving the converse of Lemma 3.2 we first show that if G_{ps} contains a good subgraph then in fact it contains a good subgraph of a restricted kind.

Lemma 3.3. *Suppose G_{ps} contains a good subgraph. Then it contains a good subgraph $(Y, \Longrightarrow)$ which satisfies the following condition*:

- *Suppose $(q_p, q_s) \overset{(e,e')}{\Longrightarrow} (q'_p, q'_s)$ and $(q_p, q_s) \overset{(e,e'')}{\Longrightarrow} (q'_p, q''_s)$. Then $e' = e''$ and hence $q'_s = q''_s$.*

Proof. Let $(Y_1, \Longrightarrow_1)$ be a good subgraph of G_{ps}. Then we set $Y = Y_1$ and fix a linear order $<$ over E_s. Define now $\Longrightarrow$ to be the least subset of $\Longrightarrow_1$ which satisfies

- Suppose $((q_p, q_s), (e, e'), (q'_p, q'_s)) \in \Longrightarrow_1$ and there does not exist $((q_p, q_s), (e, e''), (q''_p, q''_s)) \in \Longrightarrow_1$ with $e'' < e'$.
  Then $((q_p, q_s)), (e, e'), (q'_p, q'_s)) \in \Longrightarrow$.

It is now easy to check that $(Y, \Longrightarrow)$ is a good subgraph of G_{ps} having the desired property. □

We will say that a good subgraph of G_{ps} is s-deterministic ("simulation-deterministic") in case it satisfies the condition specified in the statement of Lemma 3.3.

Let $G = (Y, \Longrightarrow)$ be a s-deterministic good subgraph of G_{ps}. We next define the set of computation pairs $CP_G \subseteq E_p^* \times E_s^*$ and the map δ_G: $CP_G \longrightarrow Y$ inductively as follows. For convenience we will write $CP(\delta)$ instead of CP_G (δ_G):

- $(\varepsilon, \varepsilon) \in CP$ and $\delta((\varepsilon, \varepsilon)) = (q_{\text{in}}^p, q_{\text{in}}^s)$.

- Suppose $(\sigma,\sigma') \in CP$ and $\delta((\sigma,\sigma')) = (q_p, q_s)$ and $(q_p,q_s) \overset{(e,e')}{\Longrightarrow} (q'_p, q'_s)$. Then $(\sigma e, \sigma' e') \in CP$ and $\delta((\sigma e, \sigma' e')) = (q'_p, q'_s)$. Since $(Y, \Longrightarrow)$ is s-deterministic it is clear that if (σ,σ'), $(\sigma,\sigma'') \in CP$ then $\sigma' = \sigma''$.

Let $G = (Y, \Longrightarrow)$ be a s-deterministic good subgraph of G_{ps}. We now define the structure $TS_c = (Q_c, E_c, T_c, q_{\text{in}}, \varphi_c)$ induced by G as follows. It will turn out that TS_c is a controller for (TS_p, TS_s):

- $Q_c = Y$ and $E_c = E_p$ and $\varphi_c = \varphi_p$.
- T_c is the least subset of $Q_c \times E_c \times Q_c$ satisfying
  - Suppose $(q_p,q_s) \overset{(e,e')}{\Longrightarrow} (q'_p, q'_s)$. Then $((q_p,q_s), e, (q'_p,q'_s)) \in T_c$.
- $q^c_{\text{in}} = (q^{\text{in}}_p, q^{\text{in}}_s)$.

The next sequence of lemmas will assume the notations introduced above.

Lemma 3.4. *TS_c is a deterministic Σ-labelled transition system.*

Proof. Suppose $((q_p,q_s), e, (q'_p,q'_s)) \in T_c$ and $((q_p,q_s), e, (q''_p,q''_s)) \in T_c$. Then there exist $(q_p,q_s) \overset{(e,e')}{\Longrightarrow} (q'_p,q'_s)$ and $(q_p,q_s) \overset{(e,e'')}{\Longrightarrow} (q''_p,q''_s)$ in $G = (Y, \Longrightarrow)$. Clearly $q'_p = q''_p$ because TS_p is a deterministic Σ-labelled transition system. On the other hand, $e' = e''$ because G is s-deterministic and hence $q'_s = q''_s$ since TS_s is a deterministic Σ-labelled transition system. □

For a transition system $TS = (Q, E, T, q_{\text{in}}, \varphi)$, we define the rooted extended transition relation $\xrightarrow[*]{}$ to be the least subset of $\{q_{\text{in}}\} \times E^* \times Q$ satisfying:

- $q_{\text{in}} \xrightarrow[*]{\varepsilon} q_{\text{in}}$.
- Suppose $q_{\text{in}} \xrightarrow[*]{\sigma} q$ and $(q,e,q') \in T$. Then $q_{\text{in}} \xrightarrow[*]{\sigma e} q'$.

The next technical result will provide the basis for showing that TS_c is a controller for (TS_p, TS_c). To this end, let $TS = TS_p \| TS_c = (Q, E, T, .q_{\text{in}}, \varphi)$.

Lemma 3.5. *Let $(q^p_{\text{in}}, (q^p_{\text{in}}, q^s_{\text{in}})) \xrightarrow[*]{\sigma} (q_p, (q'_p, q_s))$ in TS. Then there exists a unique $\sigma' \in E^*_s$ such that $(\sigma,\sigma') \in CP$. Furthermore $q_p = q'_p$ and $\delta(\sigma,\sigma') = (q_p, q_s)$.*

Proof. This lemma can be proved easily by induction on $|\sigma|$, the length of σ. □

From Lemma 3.5, it follows that every reachable state of $TS = TS_p \| TS_c$ is of the form $(q_p, (q_p, q_s))$ with $q_p \in Q_p$ and $q_s \in Q_s$. The next two lemmas will show that $TS_p \| TS_c$ satisfies conditions (CT2) and (CT3), respectively.

Lemma 3.6. *Suppose $(q_p, (q_p, q_s))$ is a reachable state of $TS = TS_p \| TS_c$ and $(q_p, (q_p, q_s)) \xrightarrow{e} (q'_p, (q'_p, q'_s))$ in TS. Suppose further that $q_p \xrightarrow{e_1} q^1_p$ in TS_p with $\varphi_p(e) = \varphi(e_1)$. Then there exists $(q_p, (q_p, q_s)) \xrightarrow{e_1} (q^1_p, (q^1_p, q^1_s))$ in TS.*

Proof. Since $(q_p,(q_p,q_s))$ is a reachable state, there exists $\sigma \in E_p^* = E_c^*$ such that $(q_p^{\text{in}},(q_p^{\text{in}},q_s^{\text{in}})) \xrightarrow[*]{\sigma} (q_p,(q_p,q_s))$ in TS. By Lemma 3.5 there exists $\sigma' \in E_s^*$ such that $(\sigma,\sigma') \in CP$ and $\delta((\sigma,\sigma')) = (q_p,q_s)$. But then $(q_p,q_s) \in Y$ and $G = (Y, \Longrightarrow)$ is a good subgraph. Hence from property (G2), the required conclusion follows. □

Lemma 3.7. *Suppose* $(q_p,(q_p,q_s))$ *is a reachable state in* $TS = TS_p \| TS_c$ *and* $q_p \xrightarrow{e} q_p'$ *in* TS_p. *Then there exists* $(q_p,(q_p,q_s)) \xrightarrow{e_1} (q_p^1,(q_p^1,q_s^1))$ *in* TS.

Proof. By an argument similar to the one used for proving the previous lemma we can establish this lemma. The only difference is that we use the fact that a good subgraph of G_{ps} has the property (G3). □

Lemma 3.8. TS_c *is a controller for* (TS_p, TS_s). *Hence, if* G_{ps} *contains a good subgraph then there is a controller for* (TS_p, TS_s).

Proof. It suffices to construct a simulation from $\mathscr{U}f(TS_p \| TS_c)$ to $\mathscr{U}f(TS_s)$. Let $\widehat{TS} = \mathscr{U}f(TS_p \| TS_c) = (\hat{Q}, \hat{E}, \hat{T}, \hat{q}_{\text{in}}, \hat{\varphi})$ and $\mathscr{U}f(TS_s) = \widehat{TS}_s = (\hat{Q}_s, \hat{E}_s, \hat{T}_s, \hat{q}_{\text{in}}^s, \hat{\varphi}_s)$. Define the map $f : \hat{Q} \cup \hat{T} \longrightarrow \widehat{Q}_s \cup \widehat{T}_s$ as follows:

- Let $\hat{q} = ((q_p,(q_p,q_s)),\sigma) \in \hat{Q}$. Then $f(\hat{q}) = (q_s,\sigma')$ where σ' is the unique of member of E_s^* such that $(\sigma,\sigma') \in CP$.
- Let $\hat{t} = (((q_p,(q_p,q_s)),\sigma), e, ((q_p',(q_p',q_s'),\sigma e))$ be in $\hat{T}$.
  Then $f(\hat{t}) = ((q_s,\sigma'), e', (q_s',\sigma' e'))$ where $\sigma' \in E_s^*$ and $e' \in E_s$ such that $(\sigma,\sigma') \in CP$ and $(\sigma e, \sigma' e') \in CP$ and $\delta((\sigma e, \sigma' e')) = (q_p', q_s')$.

Again using Lemma 3.5 and the definitions, it is routine to verify that f is well defined and is in fact a simulation. □

4. The synthesis procedure

We develop here a proof of Theorem 2.9. We know from Lemmas 3.2 and 3.8 that deciding whether the pair (TS_p, TS_s) admits a controller boils down to deciding whether or not the graph G_{ps} contains a good subgraph. We shall establish in two steps that good subgraphs can be efficiently found.

Theorem 4.1. *There is a uniform decision procedure which takes as its input a pair of finite transition systems* (TS_p, TS_s) *and decides whether or not the edge-labelled directed graph* G_{ps} (*as defined in the previous section*) *contains a good subgraph.*

Proof. We set $G_0 = G_{ps}$ and construct a sequence of graphs $G_0, G_1, \ldots, G_n$ up to a stage where $G_n = G_{n+1}$. For every $i \in \{0,\ldots,n\}$, G_{i+1} will be a subgraph of G_i. This pruning procedure will remove edges or vertices which evidence violations of properties (G1) or (G2). Then testing G_n for a simple property (whether $(q^p_{\text{in}}, q^s_{\text{in}}) \in G_n$), we will decide whether or not G_{ps} contains a good subgraph.

Assume that $G_0, \ldots, G_i$, $i \geqslant 0$ have been constructed.

Let $TS_x = (Q_x, E_x, T_x, q^x_{\text{in}}, \varphi_x)$, $x \in \{p, s\}$. Now, G_{i+1} is obtained from G_i by applying one of the following pruning steps to G_i. If neither of these two steps can be applied to G_i then we set $G_{i+1} = G_i$ and stop:

(i) Let $G_i = (X_i, \to_i)$. Suppose $(q_1, q_2) \in X_i$, (q_1, e_1, q'_1) is in T_p but there is no $(e'_1, e'_2) \in E_p \times E_s$ such that $(q_1, q_2) \xrightarrow{(e'_1, e'_2)} (q'_1, q'_2)$ in G_i. Then remove (q_1, q_2) from X_i and all edges coming into (q_1, q_2). Let the resulting graph be G_{i+1}.

(ii) Suppose $(q_1, q_2) \xrightarrow{(e_1, e_2)} (q'_1, q'_2)$ is an edge of G_i and (q_1, e'_1, q''_1) is in T_p such that $\varphi_p(e_1) = \varphi_p(e'_1)$. Further, suppose that there is no edge of the form $(q_1, q_2) \xrightarrow{(e'_1, e'_2)} (q''_1, q''_2)$ in G_i. Then remove the edge $((q_1, q_2), (e_1, e_2), (q'_1, q'_2))$ from G_i and let the resulting graph be G_{i+1}.

Clearly $G_{i+1} = G_i$ (in which case we stop) or G_{i+1} is strictly smaller than G_i. Since G_0 is finite this pruning procedure must stop after a finite number of steps. Let n be the least integer such that $G_n = G_{n+1}$ and let $G_n = (X_n, \to_n)$.

Claim. *G_{ps} contains a good subgraph iff $(q^p_{\text{in}}, q^s_{\text{in}}) \in X_n$.*

To see that the claim holds, suppose G_{ps} contains a good subgraph G. Then, by induction on n, is easy to prove that G_n must also contain G as its subgraph. Thus $(q^p_{\text{in}}, q^s_{\text{in}}) \in X_n$.

Next suppose that $(q^p_{\text{in}}, q^s_{\text{in}}) \in X_n$. From the fact that $G_n = G_{n+1}$ (i.e. no pruning rule is applicable on G_n), it follows at once that G_n is a good subgraph of G_{ps}. This establishes the claim. □

Corollary 4.2. *Let $TS_p = (Q_p, E_p, T_p, q^p_{\text{in}}, \varphi_p)$ and $TS_s = (Q_s, E_s, T_s, q^s_{\text{in}}, \varphi_s)$ be a pair of finite transition systems. Let $|Q_p| = n_1$, $|Q_s| = n_2$, $|E_p| = k_1$ and $|E_s| = k_2$. Let $m = \max\{n_1, n_2, k_1, k_2\}$. Then in time polynomial in m, one can decide whether or not (TS_p, TS_s) has a controller.*

Proof. Due to Lemmas 3.2 and 3.8 it suffices to prove that in time polynomial in m one can check whether or not G_{ps} contains a good subgraph. Now consider the decision procedure developed in the proof of Theorem 4.1 for achieving this.

$G_0 = G_{ps}$ has at most $n_1 \cdot n_2$ vertices and $n_1^2 \cdot n_2^2 \cdot k_1 \cdot k_2$ edges. One can compute G_{i+1} from G_i in time which is linear in the size of G_i. Each G_{i+1} is smaller than G_i. Hence, the decision procedure will terminate in at most $n_1^2 \cdot n_2^2 \cdot k_1 \cdot k_2$ steps. □

Corollary 4.3. *Let $TS_p = (Q_p, E_p, T_p, q^p_{\text{in}}, \varphi_p)$ and $TS_s = (Q_s, E_s, T_s, q^s_{\text{in}}, \varphi_s)$ be a pair of finite transition systems. Let m be defined as in the previous corollary*:

(i) *If (TS_p, TS_s) has a controller, then it has a finite controller of size at most $n_1^2 \cdot n_2^2 \cdot k_1 \cdot k_2$.*

(ii) *Such a controller, if it exists, can be computed in time which is polynomial in m.*

Proof. Again referring to the proof of Theorem 4.1, let n be the least integer such that $G_n = G_{n+1}$. Assume that $G_n = (X_n, \rightarrow_n)$ and that $(q^p_{\text{in}}, q^s_{\text{in}}) \in X_n$. We know from the previous corollary that G_n is of size at most $n_1^2 \cdot n_2^2 \cdot k_1 \cdot k_2$ and that G_n can be computed in time which is polynomial in m.

Now suppose $G_n = (X_n, \rightarrow_n)$ has the property $(q^p_{\text{in}}, q^s_{\text{in}}) \in X_n$. Then following the proof of Lemma 3.8 one can extract a controller TS_c for (TS_p, TS_s) in time which is linear in the size of G_n. $\square$

5. The bisimulation setting

We shall show in this section that Theorem 2.9 goes through even if we replace simulations by the stronger notion of bisimulations. Though the notion of bisimulations is well established [22], we shall mention them first in our context.

Let $TS_i = (Q_i, E_i, T_i, q^i_{\text{in}}, \varphi_i)$, $i = 1, 2$, be a pair of (deterministic Σ-labelled) transition systems. A bisimulation between TS_1 and TS_2 is a relation $R \subseteq Q_1 \times Q_2$ which satisfies

- $(q^1_{\text{in}}, q^2_{\text{in}}) \in R$.
- Suppose $(q_1, q_2) \in R$ and $q_1 \xrightarrow[a]{e_1} q_1'$ is in TS_1. Then there exists a transition $q_2 \xrightarrow[a]{e_2} q_2'$ in TS_2 such that $(q_1', q_2') \in R$.
- Suppose $(q_1, q_2) \in R$ and $q_2 \xrightarrow[a]{e_2} q_2'$ is in TS_2. Then there exists a transition $q_1 \xrightarrow[a]{e_1} q_1'$ in TS_1 such that $(q_1', q_2') \in R$.

We shall say that TS_1 and TS_2 are bisimilar in case there is a bisimulation between them. Clearly, bisimilarity is an equivalence relation. It is also clear that every transition system is bisimilar to its unfolding. Hence, we can work with bisimulations between transition systems rather than between their unfoldings.

Definition 5.1. Let TS_x, $x \in \{p, s, c\}$ be three transition systems. Then TS_c is a strong controller for the pair (TS_p, TS_s) iff TS_c satisfies conditions (CT1), (CT2) of being a controller (Definition 2.5) and $TS_p \| TS_c$ is bisimilar to TS_s.

Note that we have dropped the nonblocking property (CT3). In the setting of simulations, we were capturing safety properties only and thus required that the controller should not introduce deadlock. However, in the bisimulation setting, we handle liveness specifications as well – hence we must allow the specification to demand that the plant halts at some points.

Consider Example 2.8. In the bisimulation setting, the specification demands that after a button is pressed, the possibility of serving both tea and coffee does not exist, as in the simulation setting. Further it demands that there must be a way the user can get tea and a way in which he can get coffee. A controller which serves only coffee on pressing either button will satisfy the specification in the simulation setting but not in the bisimulation setting. A controller in this setting must enable coffee (only) on one input and tea (only) in the other. It easy also to see that a minimally restricting controller does not exist in this setting too.

The synthesis problem now is the following: given a pair of finite transition systems (TS_p, TS_s), is there a strong controller for this pair?

It will be convenient to solve this problem while assuming that TS_s is reduced with respect to bisimilarity.

Let $TS = (Q, E, T, q_{\text{in}}, \varphi)$ be a transition system. Then TS is said to be *reduced* (w.r.t. bisimilarity) iff the following conditions are satisfied:

(i) $\{R \mid R \subseteq Q \times Q \text{ is a bisimulation}\} = \{id_Q\}$ where $id_Q = \{(q,q) \mid q \in Q\}$
(ii) Suppose $q \xrightarrow[a]{e_1} q'$ and $q \xrightarrow[a]{e_2} q'$. Then $e_1 = e_2$.

The next observation says why it is convenient to work with reduced transition systems:

Proposition 5.2. *Let $TS_i = (Q_i, E_i, T_i, q^i_{\text{in}}, \varphi_i)$, $i = 1, 2$, be a pair of transition systems such that TS_2 is reduced. If $\approx$ is a bisimulation between TS_1 and TS_2, then it must satisfy the following properties*:

(i) *If $q_1 \approx q_2$ and $q_1 \approx q'_2$, then $q_2 = q'_2$.*
(ii) *If $q_1 \approx q_2$ and $q_1 \xrightarrow[a]{e_1} q'_1$, then there exists a* unique *$e_2 \in E_2 : q_2 \xrightarrow[a]{e_2} q'_2$ and $q'_1 \approx q'_2$.*

Proof. Follows easily from the definitions. □

Through the rest of this section, we fix a pair of finite transition systems (TS_p, TS_s) with $TS_x = (Q_x, E_x, T_x, q^x_{\text{in}}, \varphi_x)$, $x \in \{p, s\}$. We recall the definition of the edge-labelled directed graph G_{ps} and the associated terminology developed in Section 3. Let $G_{ps} = (X, \rightarrow)$ and $G = (Y, \Rightarrow)$ be a subgraph of G_{ps}. Then G is a strong subgraph of G_{ps} iff the following conditions are satisfied:

(BS0) $(q^p_{\text{in}}, q^s_{\text{in}}) \in Y$.
(BS1) Suppose $(q, q') \overset{(e_1, e'_1)}{\Longrightarrow} (q_1, q'_1)$ is in G and $q \xrightarrow{e_2} q_2$ is in TS_p with $\varphi_p(e_1) = \varphi_p(e_2)$. Then there exists $(q, q') \overset{(e_2, e'_2)}{\Longrightarrow} (q_2, q'_2)$ in G.
(BS2) Suppose $(q, q') \in Y$ and $q' \xrightarrow{e'_1} q'_1$ is in TS_s. Then there exists $(q, q') \overset{(e_1, e'_1)}{\Longrightarrow} (q_1, q'_1)$ in G.
(BS3) Let $(q, q') \in Y$ and $E_{q,q'} = \{(e_1, e'_1) \mid \exists (q_2, q'_2) : (q, q') \overset{(e_1, e'_1)}{\Longrightarrow} (q_2, q'_2) \text{ is in } G\}$. Then there exists $\Gamma \subseteq E_{q,q'}$ satisfying:
  (i) If $(e_1, e'_1) \in E_{q,q'}$, then there exists $e_2 \in E_p$ such that $(e_2, e'_1) \in \Gamma$.
  (ii) If $(e_1, e'_1) \in E_{q,q'}$, then there exists $e'_2 \in E_s$ such that $(e_1, e'_2) \in \Gamma$.
  (iii) If $(e_1, e'_1), (e_1, e'_2) \in \Gamma$, then $e'_1 = e'_2$.

Our aim now is to show that (TS_p, TS_s) admits a strong controller iff G_{ps} contains a strong subgraph.

Lemma 5.3. *If there is a strong controller for the pair of finite transition systems (TS_p, TS_s), where TS_s is reduced, then G_{ps} has a strong subgraph.*

Proof. Let $TS_c = (Q_c, E_c, T_c, q^c_{\text{in}}, \varphi_c)$ be a strong controller for (TS_p, TS_s). Let $TS_p \| TS_c = TS = (Q, E, T, q_{\text{in}}, \varphi)$. Let $\approx \subseteq Q \times Q_s$ be a bisimulation. Now, define $G = (Y, \Rightarrow)$, a subgraph of G_{ps}, as follows:

- $(q_p, q_s) \in Y$ iff $\exists q_c \in Q_c : (q_p, q_c) \approx q_s$
- $(q_p, q_s) \overset{(e_1, e_1')}{\Longrightarrow} (q_p', q_s')$ iff $\exists q_c, q_c' \in Q_c : (q_p, q_c) \approx q_s$, $(q_p, q_c) \xrightarrow{e_1} (q_p', q_c') \in T$, $\varphi_p(e_1) = \varphi_s(e_1')$, $q_s \xrightarrow{e_1'} q_s'$, and $(q_p', q_c') \approx q_s'$.

Claim. G is a strong subgraph of G_{ps}.
Since $(q^p_{\text{in}}, q^c_{\text{in}}) \approx q^s_{\text{in}}$, $(q^p_{\text{in}}, q^s_{\text{in}}) \in Y$.

Now suppose that $(q_p, q_s) \overset{(e_1, e_1')}{\Longrightarrow} (q_p', q_s')$ is in G and $q_p \xrightarrow{e_2} q_p''$ is in TS_p, with $\varphi_p(e_1) = \varphi_p(e_2)$. Then $\exists q_c, q_c' \in Q_c : (q_p, q_c) \approx q_s$, $(q_p, q_c) \xrightarrow{e_1} (q_p', q_c')$ is in TS, $(q_p', q_c') \approx q_s'$ and $q_s \xrightarrow{e_1'} q_s'$ is in TS_s. Since TS_c is a controller, $(q_p, q_c) \xrightarrow{e_2} (q_p'', q_c'')$ is in TS. Since $\approx$ is a bisimulation, $\exists e_2' : q_s \xrightarrow{e_2'} q_s''$ and $(q_p'', q_c'') \approx q_s''$. Hence $(q_p, q_s) \overset{(e_2, e_2')}{\Longrightarrow} (q_p'', q_s'')$ is in G. This establishes (BS1).

Now suppose $(q_p, q_s) \in Y$ and $q_s \xrightarrow{e_1'} q_s'$ is in TS_s. Then $\exists q_c \in Q_c : (q_p, q_c) \approx q_s$. Hence $\exists e_1 : (q_p, q_c) \xrightarrow{e_1} (q_p', q_c')$, $(q_p', q_c') \approx q_s'$ and $\varphi_p(e_1) = \varphi_s(e_1')$. So $(q_p, q_s) \overset{(e_1, e_1')}{\Longrightarrow} (q_p', q_s')$ is in G.

To show (BS3), let $(q_p, q_s) \in Y$ and let E_{q_p, q_s} be as defined in the condition. Fix some $q_c \in Q_c$ with $(q_p, q_c) \approx q_s$. Then define $\Gamma = \{(e_1, e_1') \mid \varphi_p(e_1) = \varphi_s(e_1'), (q_p, q_c) \xrightarrow{e_1} (q_p', q_c')$ in TS, $q_s \xrightarrow{e_1'} q_s'$ in TS_s and $(q_p', q_c') \approx q_s'\}$. Clearly $\Gamma \subseteq E_{q_p, q_s}$.

If $(e_1, e_1') \in E_{q_p, q_s}$, then $q_s \xrightarrow{e_1'} q_s'$ is in TS_s.

Since $(q_p, q_c) \approx q_s$, $\exists e_2 : (q_p, q_c) \xrightarrow{e_2} (q_p', q_c')$, $\varphi_p(e_1) = \varphi_s(e_2)$, $(q_p', q_c') \approx q_s'$. Hence $(e_2, e_1') \in \Gamma$. This shows BS3(i).

Let $(e_1, e_1') \in E_{q_p, q_s}$. Then $\exists \tilde{q}_c, \tilde{q}_c' : (q_p, \tilde{q}_c) \approx q_s$, $(q_p, \tilde{q}_c) \overset{(e_1, e_1')}{\Longrightarrow} (q_p', \tilde{q}_c')$, $q_s \xrightarrow{e_1'} q_s'$ and $(q_p', \tilde{q}_c') \approx q_s'$. Since $(q_p, q_c) \approx q_s$, $\exists e_2 : (q_p, q_c) \xrightarrow{e_2} (q_p'', q_c'')$ in TS such that $\varphi_p(e_2) = \varphi_s(e_1') = \varphi_p(e_1)$ and $(q_p'', q_c'') \approx q_s'$. Since TS_c is a controller and $\varphi_p(e_1) = \varphi_p(e_2)$, $\exists q_c' :$ $(q_p, q_c) \xrightarrow{e_1} (q_p', q_c')$. Then $\exists e_2' : q_s \xrightarrow{e_2'} q_s''$, $\varphi_p(e_1) = \varphi_s(e_2')$, $(q_p', q_c') \approx q_s''$. Then $(e_1, e_2') \in \Gamma$. This proves BS3(ii).

Now let $(e_1, e_1'), (e_1, e_2') \in \Gamma$. Then $(q_p, q_c) \xrightarrow{e_1} (q_p', q_c')$, $q_s \xrightarrow{e_1'} q_s'$, $q_s \xrightarrow{e_2'} q_s''$ and $(q_p', q_c') \approx q_s'$, $(q_p', q_c') \approx q_s''$. By Proposition 5.2, we know that $e_1' = e_2'$. □

Lemma 5.4. *Suppose (TS_p, TS_s) is such that TS_s is reduced and G_{ps} has a strong subgraph. Then there exists a strong controller for (TS_p, TS_s).*

Proof. Let $G=(Y,\Rightarrow)$ be a strong subgraph of G_{ps}. For each $(q,q')\in Y$, let us fix a $\Gamma_{q,q'}\subseteq E_{q,q'}$ satisfying condition (BS3). Consider the following transition system: $TS_c=(Q_c,E_c,T_c,q^c_{\text{in}},\varphi_c)$ given by

- $Q_c=Y$.
- $E_c=E_p$; $\varphi_c=\varphi_p$.
- $((q_p,q_s),e,(q'_p,q'_s))\in T_c$ iff $\exists(e,e')\in\Gamma_{q_p,q_s}:(q_p,q_s)\overset{(e,e')}{\Longrightarrow}(q'_p,q'_s)$ in G.
- $q^c_{\text{in}}=(q^p_{\text{in}},q^s_{\text{in}})$.

It is now a tedious but straightforward exercise to verify that TS_c is a strong controller for the pair (TS_p,TS_s). We can, in fact, show that every state of $TS_p\|TS_c$ is of the form $(q_p,(q_p,q_s))$ and that if we define $\approx$ as $(q_p,(q_p,q_s))\approx q'_s$ iff $q_s=q'_s$, then $\approx$ is a bisimulation between $TS_p\|TS_c$ and TS_s. $\square$

We now wish to show that the existence of a strong controller can be decided in polynomial time. As a first step we will observe that assuming the specification transition system is reduced involves no loss of generality.

Lemma 5.5. *Let $TS=(Q,E,T,q_{\text{in}},\varphi)$ be a finite transition system. Then in time polynomial in $|TS|$ one can construct a reduced transition system TS' which is bisimilar to TS.*

Proof. This observation follows easily from the polynomial time algorithm for checking bisimilarity of two finite transition systems due to [12].

To be specific, we set $R_0=Q\times Q$ and construct a sequence of relations $R_0,R_1,\ldots,R_n$ till $R_n=R_{n+1}$ and then stop. Assume inductively that $R_0,R_1,\ldots,R_i$ have been constructed. We define R_{i+1} to be the relation obtained by applying one of the following pruning steps to R_i. If neither of the two steps can be applied to R_i, then we set $R_{i+1}=R_i$ and stop:

- Suppose $(q,q')\in R_i$ and $q\xrightarrow{e_1}q_1$ is in T but there is no $q'\xrightarrow{e'_1}q'_1$ in T such that $\varphi(e_1)=\varphi(e'_1)$ and $(q_1,q'_1)\in R_i$. Then $R_{i+1}=R_i\setminus\{(q,q')\}$.
- Suppose $(q,q')\in R_i$ and $q'\xrightarrow{e'_1}q'_1$ is in T but there is no $q\xrightarrow{e_1}q_1$ in T such that $\varphi(e_1)=\varphi(e'_1)$ and $(q_1,q'_1)\in R_i$. Then $R_{i+1}=R_i\setminus\{(q,q')\}$.

Since R_0 is a finite set and $R_{i+1}=R_i$ (in which case we stop) or $R_{i+1}\subset R_i$, this procedure will terminate after at most $|Q\times Q|$ steps. Let n be the least integer such that $R_n=R_{n+1}$. It is easy to check that R_n is an equivalence relation. For $q\in Q$, let $[q]$ be the R_n-equivalence class containing q.

Next, we fix a strict linear order $<$ on E.

We now define $TS'=(Q',E',T',q'_{\text{in}},\varphi')$ via:

- $Q'=Q/R_n=\{[q]\mid q\in Q\}$.
- $E'=E$; $\varphi'=\varphi$.

- $([q], e, [q']) \in T'$ iff there exists $(q_1, e, q_1') \in T$. such that $q_1 \in [q]$ and $q_1' \in [q']$ and furthermore, if $(p, e', p') \in T$ with $p \in [q]$ and $p' \in [q']$ and $\varphi(e) = \varphi(e')$, then $e = e'$ or $e < e'$.
- $q'_{\text{in}} = [q_{\text{in}}]$.

It is easy to verify that TS' is reduced and that TS and TS' are bisimilar with $\{(q, [q]) \mid q \in Q\}$ being a bisimulation. It is also easy to verify that $|TS'| \leqslant |TS|$ and that TS' can be computed in time polynomial in $|TS|$. □

Theorem 5.6. *There is a uniform procedure which takes as input a pair of finite transition systems* (TS_p, TS_s) *and decides whether or not* (TS_p, TS_s) *admits a strong controller.*

Proof. Due to the previous lemma, it involves no loss of generality to assume that TS_s is reduced. It now follows from Lemmas 5.3 and 5.4 that it suffices to decide whether or not G_{ps} contains a strong subgraph. This can be achieved by constructing a sequence of graphs $G_0, G_1, \ldots, G_{n+1}$ such that each G_i is a subgraph of G_{ps} and each G_{i+1} a subgraph of G_i with $G_0 = G_{ps}$ and $G_n = G_{n+1}$. Assume inductively that $G_0, \ldots, G_i$ have been constructed. We now obtain G_{i+1} by applying one of the following pruning steps to G_i. If none of the pruning steps can be applied we set $G_{i+1} = G_i$ and stop.

Let $G_i = (X_i, \rightarrow_i)$:

(PR1) Suppose $t = ((q, q'), (e_1, e_1'), (q_1, q_1')) \in \rightarrow_i$ and there exists $(q, e_2, q_2) \in TS_p$ with $\varphi_p(e_1) = \varphi_p(e_2)$. Further suppose that there exists no edge in $\rightarrow_i$ of the form $((q, q'), (e_2, e_2'), (q_2, q_2'))$. Then remove the edge t from $\rightarrow_i$ and set G_{i+1} to be the resulting graph.

(PR2) Suppose $(q, q') \in X_i$ and $q' \xrightarrow{e_1'} q_1'$ is in T_s but there is no edge of the form $((q, q'), (e_1, e_1'), (q_1, q_1'))$ in $\rightarrow_i$. Then remove (q, q') and all its incoming and outgoing edges from G_i and define G_{i+1} to be the resulting graph.

(PR3) Let $(q, q') \in X_i$.
Let $E^i_{q,q'} = \{(e_1, e_1') \mid \exists (q_2, q_2') : ((q, q'), (e_1, e_1'), (q_2, q_2'))$ is in $\rightarrow_i\}$ Suppose every $\Gamma \subseteq E^i_{q,q'}$ fails to satisfy at least one of the conditions BS3 (i), (ii) and (iii). Then remove (q, q') and all its incoming and outgoing edges and define G_{i+1} to be the resulting graph.

Since G_0 is finite, this procedure will terminate after a finite number of steps. Let n be the least integer such that $G_n = G_{n+1}$. Let $G_n = (X_n, \rightarrow_n)$. Now it is easy to show that G_{ps} contains a strong subgraph iff $(q^p_{\text{in}}, q^s_{\text{in}}) \in X_n$.

First, if $(q^p_{\text{in}}, q^s_{\text{in}}) \in X_n$, it is clear that G_n is a strong subgraph of G_{ps}.

To prove the converse, let us assume that G_{ps} has a strong subgraph G. We can inductively prove (by induction on n) that G is a subgraph of G_n. It would then follow that since $(q^p_{\text{in}}, q^s_{\text{in}})$ is in G, it would be in G_n also. The induction goes as follows.

Clearly, G is a subgraph of $G_0 = G_{ps}$. Now assume inductively that G is a sugraph of G_i. If pruning step (PR1) or (PR2) is applied, it is easy to see that G will be a subgraph of G_{i+1}. If (PR3) is used, let the pruned node be (q,q'). To prove G is a subset of G_{i+1} it suffices to prove that $(q,q') \notin G$. Assume the contrary. Then since G is a strong subgraph, it satisfies (BS3) for the node (q,q') – let $\Gamma \subseteq E_{q,q'}$ be a set which satisfies (BS3)(i)–(iii) with $E_{q,q'} = \{(e,e') \mid \exists (q_2,q_2') : ((q,q'),(e,e'),(q_2,q_2'))$ is in $G\} \subseteq E^i_{q,q'}$.

Now it is clear that $\Gamma \subseteq E^i_{q,q'}$. We will show that Γ in fact satisfies (BS3)(i)–(iii) for the node (q,q') in G_i.

Let $(e,e') \in E^i_{q,q'}$. Then there must be a transition $q' \xrightarrow{e'} q_1'$ in T_s. Since G is a strong subgraph, by (BS2), $\exists e_1 : (q,q') \xrightarrow{e_1,e'} (q_1,q_1')$ is in G. Hence $(e_1,e') \in E_{q,q'}$. By (BS3)(i) for (q,q') in G, $\exists e_2 : (e_2,e') \in \Gamma$. This shows (BS3)(i) for (q,q') in G_i.

Let $(e,e') \in E^i_{q,q'}$. Then, as argued above, $\exists e_1 : (q,q') \xrightarrow{e_1,e'} (q_1,q_1')$ is in G. Since $\varphi_p(e) = \varphi_p(e_1)$, by (BS1), $\exists e_2' : (q,q') \xrightarrow{e,e_2'} (q_2,q_2')$ is in G. So, $(e,e_2') \in E_{q,q'}$. Since G is a strong subgraph, by BS3(ii), $\exists e_3' : (e,e_3') \in \Gamma$, which shows BS3(ii) holds for G_i.

Also, since Γ satisfies BS3(iii) for G, it also satisfies it for G_i.

This shows that the conditions for using (PR3) is not met, which contradicts our assumption. Hence $(q,q') \notin G$ and hence G is a subgraph of G_{i+1}. □

Corollary 5.7. *Let (TS_p, TS_s) be a pair of finite transition systems with $|Q_p| = n_p$, $|Q_s| = n_s$, $|E_p| = k_p$ and $|E_s| = k_s$. Let $m = \max\{n_p, n_s, k_p, k_s\}$. Then in time polynomial in m, one can decide whether or not (TS_p, TS_s) admits a strong controller.*

Proof. By Lemma 5.5, we can construct in time polynomial in m, a reduced transition system TS_s' such that TS_s and TS_s' are bisimilar. We can now supply (TS_p, TS_s') as input to the decision procedure presented in the proof of Theorem 5.5. This procedure will take time only polynomial in m. To show this, the only nontrivial part is to show how rule (PR3) can be implemented in time polynomial in m. We will do this by showing a reduction to the *maximal matching problem*.

Let $G_i = (X_i, \rightarrow_i)$ and $(q,q') \in X_i$. Let $E_{q,q'}$ be defined as before. Consider the bipartite (undirected) graph (S_1, S_2, A) where

- $S_1 = \{e_1 \mid \exists e_2 : (e_1,e_2) \in E_{q,q'}\}$,
- $S_2 = \{e_2 \mid \exists e_1 : (e_1,e_2) \in E_{q,q'}\}$,
- $A = E_{q,q'}$.

It is easy to see that there exists a Γ satisfying the conditions BS3(i)–(iii) iff the maximal matching of (S_1,S_2,A) is of size $|S_2|$. Since maximal matching can be done in polynomial time, we can implement (PR3) in polynomial time. In fact, we can solve the maximal matching problem by reducing it to the maxflow problem and use the Ford–Fulkerson method (see [7]) to get a maximal matching in polynomial time, from which we can get a witness Γ. These witnesses will be useful in constructing the controller. □

Corollary 5.8. *Let (TS_p, TS_s) be a pair of finite transition systems with m defined as above. Then (TS_p, TS_s) admits a strong controller iff it admits a strong controller* *of size at most a polynomial in m. Moreover, such a controller can be constructed in time polynomial in m.*

Proof. Using the decision procedure presented in the proof of Theorem 5.6, once can compute a strong subgraph of G_{ps}, if one exists, in time polynomial in m. We can synthesize a strong controller from the strong subgraph as shown in the proof of Lemma 5.4. Clearly, the size of this controller will be at most polynomial in m. □

6. A negative result in a concurrent setting

Transition systems can be augmented with some concurrency information to model distributed systems. Here we consider one well-established variant called asynchronous transition systems [4, 31]. The system and the specification will be modeled as asynchronous transition systems and the notion of a simulation from one asynchronous transition system to another will be defined so that it preserves the independence of events. We will show that the problem of deciding whether there exists a (finite) controller in this setting is undecidable. In fact, it turns out that even the problem of deciding whether there is a simulation from one asynchronous transition system to another is undecidable. We will also show that this negative result holds even in very restricted classes of asynchronous transition systems.

A Σ-labelled deterministic asynchronous transition system is a structure $TS=(Q,E,T,q_{\text{in}},\varphi,I)$ where $(Q,E,T,q_{\text{in}},\varphi)$ is a transition system and $I\subseteq E\times E$ is an irreflexive and symmetric independence relation such that the following conditions are satisfied:

(TR1) Suppose $q\xrightarrow{e_1}q_1$ and $q\xrightarrow{e_2}q_2$ and $e_1\,I\,e_2$. Then there exists q' such that $q_1\xrightarrow{e_2}q'$ and $q_2\xrightarrow{e_1}q'$.

(TR2) Suppose $q\xrightarrow{e_1}q_1\xrightarrow{e_2}q'$ and $e_1\,I\,e_2$. Then there exists q_2 such that $q\xrightarrow{e_2}q_2\xrightarrow{e_1}q'$.

From now on, we will refer to Σ-labelled deterministic asynchronous transition systems as just asynchronous transition systems.

Simulations will now be required to preserve the independence of events. Let $TS_1=(Q_1,E_1,T_1,q_{\text{in}}^1,\varphi_1,I_1)$ and $TS_2=(Q_2,E_2,T_2,q_{\text{in}}^2,\varphi_2,I_2)$ be a pair of asynchronous transition systems. Then an asynchronous simulation $f:TS_1\to TS_2$ is a simulation from $(Q_1,E_1,T_1,q_{\text{in}}^1,\varphi_1)$ to $(Q_2,E_2,T_2,q_{\text{in}}^2,\varphi_2)$ which in addition satisfies

- Suppose in TS_1, we have $e_1\,I_1\,e_2$, $t_1=(q,e_1,q_1)$, $t_2=(q_1,e_2,q')$, $t_3=(q,e_2,q_2)$ and $t_4=(q_2,e_1,q')$.
  - If $f(t_1)=(p,e_1',p_1)$ and $f(t_2)=(p_1,e_2',p')$ then $e_1'\,I_2\,e_2'$ and there exists p_2 such that $f(t_3)=(p,e_2',p_2)$ and $f(t_4)=(p_2,e_1',p')$.
  - If $f(t_1)=(p,e_1',p_1)$ and $f(t_3)=(p,e_2',p_2)$ then $e_1'\,I_2\,e_2'$ and there exists p' such that $f(t_2)=(p_1,e_2',p')$ and $f(t_4)=(p_2,e_1',p')$.

From now on we will often drop the adjective "asynchronous" in referring to asynchronous simulations. As before controllers will be defined in terms of unfoldings. The new feature is that the independence of events will induce a partial order over the runs of the system. A standard technique taken from Mazurkiewicz trace theory [8] will be used to group together different interleavings of the same partially ordered stretch of behaviour.

Let $TS = (Q, E, T, q_{\text{in}}, \varphi, I)$ be an asynchronous transition system. Then $\sim_{TS}$ is the least equivalence relation (which will turn out to be a congruence) contained in $E^* \times E^*$ which satisfies: $\tau e_1 e_2 \tau' \sim_{TS} \tau e_2 e_1 \tau'$ whenever $e_1 \, I \, e_2$ and $\tau, \tau' \in E^*$. We let $[\tau]$ denote the $\sim_{TS}$-equivalence class containing τ. We now define $\mathcal{U}f(TS) = (\hat{Q}, \hat{E}, \hat{T}, \hat{q}_{\text{in}}, \hat{\varphi}, \hat{I})$ via:

- $(q_{\text{in}}, [\varepsilon]) \in \hat{Q}$.
- If $(q, [\tau]) \in \hat{Q}$ and $(q, e, q') \in T$ then $(q', [\tau e]) \in \hat{Q}$, $e \in \hat{E}$ and $((q, [\tau]), e, (q', [\tau e])) \in \hat{T}$.

The rest of the definition is routine. Trace theory will ensure that $\mathcal{U}f(TS)$ is also an asynchronous transition system. Next, we consider products of asynchronous transition systems. The new feature is that the concerned independence relations should agree on the common events. Let TS_1 and TS_2 be two asynchronous transition systems with E_i as the set of events and φ_i as the labelling function of TS_i, $i \in \{1,2\}$. Then $TS_1 \| TS_2$ is defined iff $\forall e, e' \in E_1 \cap E_2.\ e \, I_1 \, e'$ iff $e \, I_2 \, e'$. If this condition is satisfied then $TS_1 \| TS_2$ is defined as done in Section 2 with the new independence relation defined as $I_1 \cup I_2$. Again, it should be clear that $TS_1 \| TS_2$ is an asynchronous transition system.

Let TS_p, TS_s and TS_c be three asynchronous transition systems. Then TS_c is an asynchronous controller for (TS_p, TS_s) iff TS_c satisfies the usual properties (CT1)–(CT3) of Definition 2.5 for being a controller and if there exists an asynchronous simulation from $\mathcal{U}f(TS_p \| TS_c)$ into $\mathcal{U}f(TS_s)$.

Let us consider the example given below in Fig. 1. The plant consists of two agents which do the following: they wait for the user to press a button (ask_i) after which they enter a critical section (cs_i). When they finish and exit the critical section, they send a signal (fin_i) which can be observed by the other agent. The two agents are shown in Fig. 1. The combined system is the normal synchronized product of the two systems. The unfolding of the plant is also shown in Fig 1. The induced independence relation is the symmetric closure of $\{ask_1, cs_1\} \times \{ask_2, cs_2\}$. Let us fix the labelling function as $\varphi(ask_i) = ask$, $\varphi(cs_i) = cs$ and $\varphi(fin_i) = fin$, $i \in \{1,2\}$.

The specification TS_s^4 (which is equivalent to its unfolding) is shown above with only the labels of events on the transitions – the independence of events should be clear. On the labels, it is identical to the plant, except that it has no moves enabled when both agents are in the critical section. This therefore demands that the plant should not reach a state where both agents are in their critical sections (if it reaches such a state, then the controller will not be able to satisfy the nonblocking condition at this state).

An asynchronous controller is required to respect the independence relation. Hence, it cannot enable an agent entering a critical section depending upon an independent event occurring in the other agent. In this example, the controller is forced to sequentialize the agents in a predetermined manner – an example of a valid controller is TS_c shown above

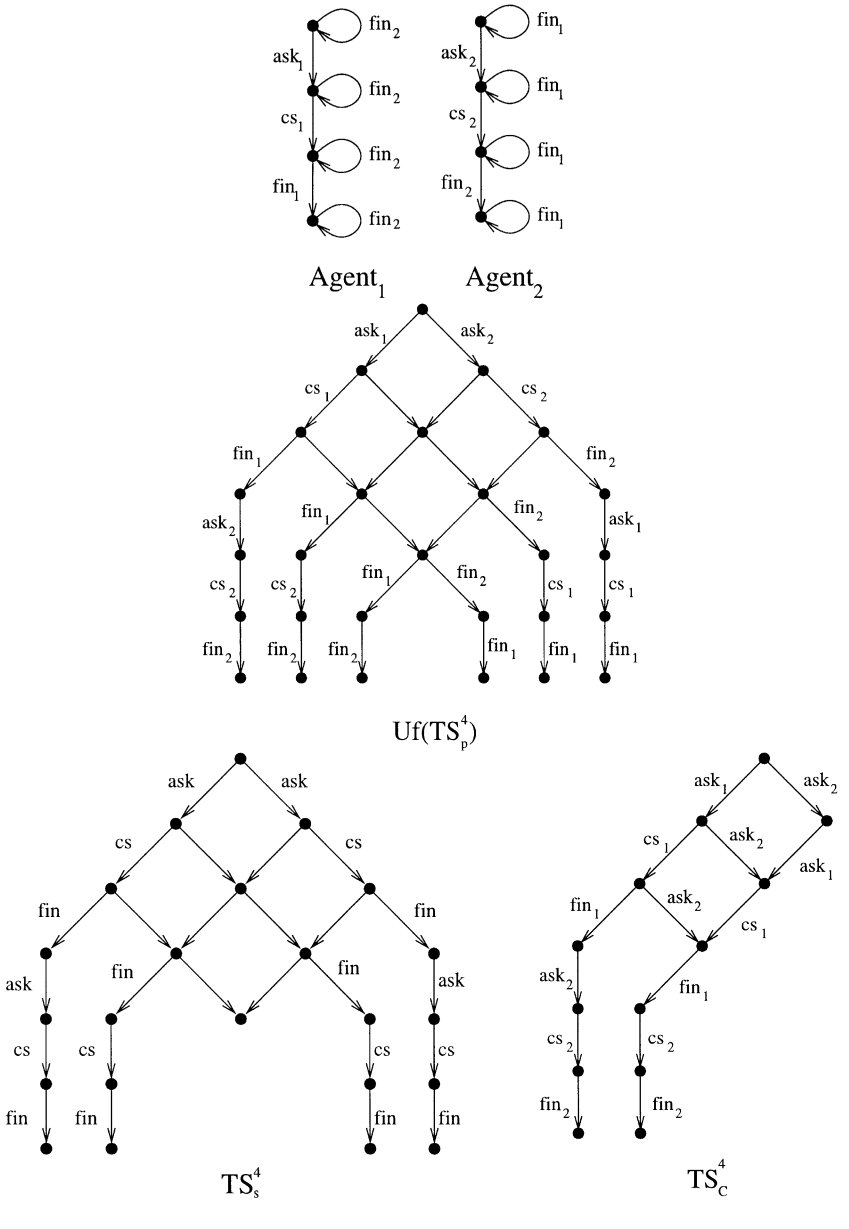


Fig. 1.

which allows the first agent to enter its critical section before the second, regardless of the sequence of buttons pressed.

We now wish to show that the problem of deciding if a pair of *finite* asynchronous transition systems admits an asynchronous controller – finite or otherwise – is undecidable. The reduction is from the tiling problem [19] which is known to be undecidable.

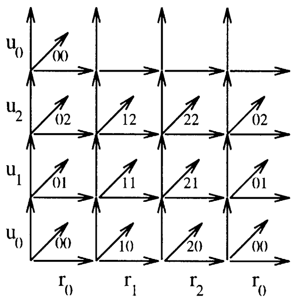


Fig. 2. The main grid.

In what follows, it will be convenient to talk about the tiling problem as a colouring problem. An instance of the colouring problem is a quadruple $\mathscr{C}P = (C, c_{\text{in}}, R, U)$ where C is a finite set of colours, $c_{\text{in}} \in C$ is a distinguished initial colour and $R : C \longrightarrow 2^C$ and $U : C \longrightarrow 2^C$ are two functions. A solution to $\mathscr{C}P$ is a map $col : \omega \times \omega \longrightarrow C$ (ω is the set of natural numbers) which satisfies

- $col(0,0) = c_{\text{in}}$,
- $\forall (m,n) \in \omega \times \omega.\ col(m+1,n) \in R(col(m,n))$ and $col(m,n+1) \in U(col(m,n))$.

For each instance $\mathscr{C}P$ of a colouring problem we will construct a pair of finite asynchronous transition systems (TS_p, TS_s) such that $\mathscr{C}P$ has a solution iff there exists an asynchronous *simulation* from $\mathscr{U}f(TS_p)$ into $\mathscr{U}f(TS_s)$.

We will then show that for each pair of finite asynchronous transition system (TS_p, TS_s) over an alphabet Σ, we can effectively construct a pair of finite asynchronous transition systems (TS'_p, TS'_s) such that there exists an asynchronous simulation from $\mathscr{U}f(TS_p)$ into $\mathscr{U}f(TS_s)$ iff there exists an asynchronous *controller* for the pair of systems (TS'_p, TS'_s). This will lead to the desired result.

Through the rest of the section fix an instance of the colouring problem $\mathscr{C}P = (C, c_{\text{in}}, R, U)$ and let c, c' range over C. The associated pair of finite asynchronous transition systems will be denoted as TS_p and TS_s. It will be convenient to explain how the construction works by displaying the unfoldings of the systems rather than the systems themselves. We will construct the systems later.

The main part of $\mathscr{U}f(TS_p)$ will look like a two-dimensional grid generated by the two sets of events $E_R = \{r_0, r_1, r_2\}$ and $E_U = \{u_0, u_1, u_2\}$ with $E_R \times E_U \subseteq I_p$ where I_p is the independence relation of $\mathscr{U}f(TS_p)$. This is shown in Fig. 2. We display only the events concerned and not their labels. We will deal with the labels later.

In addition, there will be nine events $\{0,1,2\}^2$. At each grid point at most four such events will be sticking out. For convenience we will often write ij instead of (i,j) for $i,j \in \{0,1,2\}$. At a grid point, the event ij will be enabled if r_i and u_j are enabled at this point. This event will commute with events r_i and u_j enabled at this grid point. It will also commute with the events $i(j+1)$ and $(i+1)j$ enabled at the neighbouring grid points. Here and in what follows addition is taken to be addition modulo 3. Thus

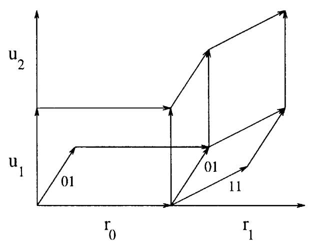


Fig. 3.

the independence relation I_p will demand:

- $ij\ I_p\ r_{i'}$ iff $i = i'$
- $ij\ I_p\ u_{j'}$ iff $j = j'$
- $ij\ I_p\ i'j'$ iff $[(i' = i + 1$ and $j = j')$ or $(i = i'$ and $j' = j + 1)]$

TS_p is such that along any run, an event ij can occur at most once. Thus a typical neighbourhood in $\mathcal{U}f(TS_p)$ will look as in Fig. 3.

Note that once an event of type ij is performed, one can never get back to the main grid; at most three more events can be performed before reaching a terminal state. These events which stick out of the grid will be used – via a simulation – to check whether the colours assigned to neighbouring grid points are consistent.

The assignment of colours to the grid points will be done in $\mathcal{U}f(TS_s)$. This transition system will look exactly like $\mathcal{U}f(TS_p)$ except that we will use events taken from the set $C \times \{0,1,2\}^2$ instead of $\{0,1,2\}^2$. At a grid point, the event (c, ij) will be enabled if r_i and u_j are enabled at this point. As an exception, at the origin only the event $(c_{\text{in}}, 00)$ will be enabled apart from the events r_0 and u_0. In addition the event (c, ij) can wander forward a bit through the independence relation as described below. The crucial point is, the independence relation I_s of $\mathcal{U}f(TS_s)$ will be used to check for the consistency of the colouring scheme. We define I_s to be the least irreflexive and symmetric subset of $E_s \times E_s$ with $E_s = \{r_0, r_1, r_2, u_0, u_1, u_2\} \cup (C \times \{0,1,2\}^2)$ satisfying

- $\{r_0, r_1, r_2\} \times \{u_0, u_1, u_2\} \subseteq I_s$.
- $r_i\ I_s\ (c, i'j')$ if $i = i'$
- $u_j\ I_s\ (c, i'j')$ if $j = j'$
- $(c, ij)\ I_s\ (c', i'j')$ if $[(i' = i + 1,\ j' = j$ and $c' \in R(c))$ or $(i' = i,\ j' = j + 1$ and $c' \in U(c))]$.

We force $\mathcal{U}f(TS_p)$ and $\mathcal{U}f(TS_s)$ to march together by a suitable choice of labels. Fix $\Sigma = \{r_0, r_1, r_2, u_0, u_1, u_2\} \cup \{0,1,2\}^2$. In both the systems the event $x \in \{r_0, r_1, r_2, u_0, u_1, u_2\}$ will get the label x. Each event ij in TS_p will get the label ij and each event (c, ij) in TS_s will get the label ij.

More formally, TS_p and TS_s are Σ-labelled transition systems, (where $\Sigma = \{r_0, r_1, r_2, u_0, u_1, u_2\} \cup \{0,1,2\}^2$) defined as follows:

TS_p is defined as $TS_p = (Q_p, E_p, T_p, q_{\text{in}}^p, \varphi_p, I_p)$ where

- $E_p = \{r_i, u_i \mid i \in \{0,1,2\}\} \cup \{ij \mid i,j \in \{0,1,2\}\}$.
- $\varphi_p(x) = x$ for all $x \in E_p$.
- I_p is the least irreflexive and symmetric subset of $E_p \times E_p$ satisfying:
  $\{r_0, r_1, r_2\} \times \{u_0, u_1, u_2\} \subseteq I_p$
  and for all $i, j, i', j' \in \{0,1,2\}$,
  $ij \; I_p \; r_i$, $ij \; I_p \; u_j$ and
  $ij \; I_p \; i'j'$ if ($i' = i+1$ and $j = j'$) or ($i' = i$ and $j' = j+1$).

We will denote by D_p the dependence relation: $D_p = (E_p \times E_p) \backslash I_p$.

- $Q_p = \{r_0, r_1, r_2\} \times \{u_0, u_1, u_2\} \times 2^{\{0,1,2\}} \times 2^{\{0,1,2\}} \times 2^{\{0,1,2\}^2}$.
- $q_{\text{in}}^p = (r_0, u_0, \{0\}, \{0\}, \emptyset)$.
- Let a typical member of Q_p be denoted as a tuple (R, U, L_R, L_U, X).
  T_p is defined as follows:
  - $(R, U, L_R, L_U, X) \xrightarrow{r_i} (R', U', L_R', L_U', X')$ if $R = r_i$, $(\nexists i'j' \in X : i \neq i')$, $R' = r_{i+1}$, $U' = U$, $L_U' = L_U$, $L_R' = L_R \backslash \{i-1\} \cup \{i+1\}$, $X' = X$.
  - $(R, U, L_R, L_U, X) \xrightarrow{u_j} (R', U', L_R', L_U', X')$ if $U = u_j$, $(\nexists i'j' \in X : j \neq j')$, $U' = u_{j+1}$, $R' = R$, $L_R' = L_R$, $L_U' = L_U \backslash \{j-1\} \cup \{j+1\}$, $X' = X$.
  - $(R, U, L_R, L_U, X) \xrightarrow{ij} (R', U', L_R', L_U', X')$ if $i \in L_R$, $j \in L_U$, $(\nexists i'j' \in X : i'j' D_p ij)$, $R' = R$, $U' = U$, $X' = X \cup \{ij\}$, $L_R' = L_R$, $L_U' = L_U$.

TS_s is defined as $TS_s = (Q_s, E_s, T_s, q_{\text{in}}^s, \varphi_s, I_s)$ where

- $E_s = \{r_i, u_i \mid i \in \{0,1,2\}\} \cup \{(c, ij) \mid c \in C, \; i,j \in \{0,1,2\}\}$.
- $\varphi_s(x) = x$ for all $x \in \{r_i, u_i \mid i \in \{0,1,2\}\}$ and $\varphi_s((c, ij)) = ij$ for all $c \in C$, $i,j \in \{0,1,2\}$.
- I_s is the least irreflexive and symmetric subset of $E_p \times E_p$ satisfying:
  $\{r_0, r_1, r_2\} \times \{u_0, u_1, u_2\} \subseteq I_s$ and for all $i, j, i', j' \in \{0,1,2\}$ and $c, c' \in C$, $(c, ij) \; I_s \; r_i$, $(c, ij) \; I_s \; u_j$ and $(c, ij) \; I_p \; (c', i'j')$ if ($i' = i+1$, $j' = j$ and $c' \in R(c)$) or ($i' = i$, $j' = j+1$ and $c' \in U(c)$).

We will denote by D_s the dependence relation: $D_s = (E_s \times E_s) \backslash I_s$:

- $Q_s = \{r_0, r_1, r_2\} \times \{u_0, u_1, u_2\} \times 2^{\{0,1,2\}} \times 2^{\{0,1,2\}} \times 2^{C \times \{0,1,2\}^2} \times \{init, *\}$.
- $q_{\text{in}}^s = (r_0, u_0, \{0\}, \{0\}, \emptyset, init)$.
- Let a typical member of Q_s be denoted as a tuple (R, U, L_R, L_U, X, S) T_s is defined as follows:
  - $(R, U, L_R, L_U, X, S) \xrightarrow{r_i} (R', U', L_R', L_U', X', S')$ if $R = r_i$, $(\nexists i'j' \in X : i \neq i')$, $R' = r_{i+1}$, $U' = U$, $L_U' = L_U$, $L_R' = L_R \backslash \{i-1\} \cup \{i+1\}$, $X' = X$, $S' = *$.
  - $(R, U, L_R, L_U, X, S) \xrightarrow{u_j} (R', U', L_R', L_U', X', S')$ if $U = u_j$, $(\nexists i'j' \in X : j \neq j')$, $U' = u_{j+1}$, $R' = R$, $L_R' = L_R$, $L_U' = L_U \backslash \{j-1\} \cup \{j+1\}$, $X' = X$, $S' = *$.
  - $(R, U, L_R, L_U, X, S) \xrightarrow{(c,ij)} (R', U', L_R', L_U', X', S')$ if $(S = init \Rightarrow c = c_{\text{in}})$, $i \in L_R$, $j \in L_U$, $(\nexists (c', i'j') \in X : (c', i'j') D (c, ij))$, $R' = R$, $U' = U$, $X' = X \cup \{(c, ij)\}$, $L_R' = L_R$, $L_U' = L_U$, $S' = S$.

The states of the plant contain the following information:

- R encodes which r_i event is enabled and U encodes which u_j event is enabled.
- L_R encodes the set of all i such that events ij' may be permitted and L_U encodes the set of all j such that events $i'j$ may be permitted. Together they encode exactly which ij events are permitted at a grid-point: an event ij is permitted iff $i \in L_R$ and $j \in L_j$.
- X encodes the set of all ij events that have occurred so far.

It should be clear now how the definitions of the plant transitions work. Note that $X = \emptyset$ at any grid point.

The specification is constructed in almost the same way, except that the L_R and L_U components encode the (c, ij) events enabled and the independence relation of the (c, ij) events are constrained by the given colouring problem. We also keep track in a new component S whether the last grid-point seen was $(0, 0)$ or not. If it is, then we only allow the c_{in} event to occur.

It is easy to see that the unfolding of the above transition systems is as we have described.

We can now show the following:

Claim. *$\mathscr{C}P$ has a solution iff there is a simulation from $\mathscr{U}f(TS_p)$ into $\mathscr{U}f(TS_s)$.*

Proof. ($\Rightarrow$) Let $col : \omega \times \omega \to C$ be a solution for $\mathscr{C}P$. Now, there is a simulation which works as follows. Map the grid-points (those points reached by using only r_i and u_j events) of $\mathscr{U}f(TS_p)$ to the grid-points of $\mathscr{U}f(TS_s)$. This is easily achieved by mapping the initial state of $\mathscr{U}f(TS_p)$ to the initial state of $\mathscr{U}f(TS_s)$ and mapping the r_i and u_j events of $\mathscr{U}f(TS_p)$ to the r_i and u_j events (respectively) in $\mathscr{U}f(TS_s)$. If at a grid-point, r_i and u_j events are enabled, then map the outgoing edge ij from this grid-point to the (c, ij) event in the corresponding grid-point of the system, where c is the colour assigned by col to that grid-point. Extend the function to map other occurrences of the same event to appropriate transitions. It is now easy to see that this defines a simulation.

($\Leftarrow$) Let $f : \mathscr{U}f(TS_p) \to \mathscr{U}f(TS_s)$ be a simulation. First, it is easy to argue that the grid-points of $\mathscr{U}f(TS_p)$ must get mapped to the grid-points of $\mathscr{U}f(TS_s)$. This follows from the fact that f must preserve the label of events that are mapped. Now, we can assign colours to the grid-points as follows: at any gridpoint, if r_i and u_j are enabled, then the colour for that grid-point is c where f maps the outgoing edge ij event to (c, ij). It follows easily from the construction and the fact that f preserves the independence of events, that the colouring defined is a solution to $\mathscr{C}P$.

Hence we have:

Theorem 6.1. *The problem of uniformly determining the existence of a simulation from the unfolding of a finite asynchronous transition system to another is undecidable.*

Next, we show that the problem of checking for an asynchronous simulation reduces to that of checking for the existence of an asynchronous controller. Given TS_p and TS_s, we construct TS'_p and TS'_s such that there exists a simulation from $\mathcal{U}f(TS_p)$ into $\mathcal{U}f(TS_s)$ iff there is a controller for (TS'_p, TS'_s). TS'_p and TS'_s will have more behaviours than TS_p and TS_s. Hence $\mathcal{U}f(TS_p)$ and $\mathcal{U}f(TS_s)$ can be embedded into $\mathcal{U}f(TS'_p)$ and $\mathcal{U}f(TS'_s)$. Further, it will turn out that if (TS'_p, TS'_s) has a controller, say TS'_c, then it would have to be the trivial controller which allows *all* system moves. Hence $\mathcal{U}f(TS'_p \| TS'_c)$ will be isomorphic to $\mathcal{U}f(TS'_p)$. Hence, if (TS'_p, TS'_s) has a controller, it would imply that there is a simulation from $\mathcal{U}f(TS'_p)$ to $\mathcal{U}f(TS'_s)$, from which we will show how to extract a simulation from $\mathcal{U}f(TS_p)$ to $\mathcal{U}f(TS_s)$. To prove the converse, we will show how any simulation from $\mathcal{U}f(TS_p)$ to $\mathcal{U}f(TS_s)$ easily extends to a simulation from $\mathcal{U}f(TS'_p)$ to $\mathcal{U}f(TS'_s)$. This will show that the completely non-restrictive controller is a valid controller for (TS'_p, TS'_s).

Let $TS_p = (Q_p, E_p, T_p, q^p_{\text{in}}, \varphi_p, I_p)$ and $TS_s = (Q_s, E_s, T_s, q^s_{\text{in}}, \varphi_s, I_s)$. Assume, without loss of generality, that Σ is disjoint from Q_p, Q_s, E_p and E_s. Then $TS'_p = (Q'_p, E'_p, T'_p, q^{p'}_{\text{in}}, \varphi'_p, I'_p)$ is defined as follows:

- $Q'_p = Q_p \cup \{X \mid X \text{ is a non-empty subset of } \Sigma\}$.
- $E'_p = E_p \cup \Sigma$.
- $\varphi'_p(e_1) = \varphi_p(e_1)$, if $e_1 \in E_p$;
  $\varphi'_p(a) = a$, if $a \in \Sigma$.
- $q^{p'}_{\text{in}} = q^p_{\text{in}}$.
- $T'_p = T_p \cup \{(q_1, a, \{a\}) \mid q_1 \in Q_p \text{ and } a \in \Sigma\}$
  $\cup \{(X, a, Y) \mid X,\ Y \text{ are non-empty subsets of } \Sigma \text{ and } a \notin X \text{ and}$
  $Y = X \cup \{a\}\}$.
- $I'_p = I_p \cup \{(a, b) \mid a \neq b \text{ and } a, b \in \Sigma\}$.

TS'_s is defined in a similar way.

Now we can prove the following:

Lemma 6.2. *There is an simulation from $\mathcal{U}f(TS_p)$ to $\mathcal{U}f(TS_s)$ iff there is a controller for (TS'_p, TS'_s).*

Proof. ($\Rightarrow$) Let $f : \mathcal{U}f(TS_p) \to \mathcal{U}f(TS_s)$ be a simulation. Consider $TS'_c = TS'_p$. Then $\mathcal{U}f(TS'_p)$ and $\mathcal{U}f(TS'_p \| TS'_c)$ are clearly isomorphic. Now it is easy to see that f can be extended to a simulation from $\mathcal{U}f(TS'_p \| TS'_c)$ to $\mathcal{U}f(TS'_s)$ by mapping all the Σ-events of $TS'_p \| TS'_c$ to the corresponding Σ-events of TS'_s.

($\Leftarrow$) Let TS'_c be a controller for (TS'_p, TS'_s). Let g be a simulation from $\mathcal{U}f(TS'_p \| TS'_c)$ to $\mathcal{U}f(TS'_s)$. First, we can show that TS'_c cannot restrict any system move of TS_p.

Claim. *If (q_p, q_c) is reachable in $TS'_p \| TS'_c$ and $q_p \xrightarrow{e} q'_p$, then $\exists q'_c : q_c \xrightarrow{e} q'_c$ in TS'_c.*

The claim can be checked as follows. If q_p is in Q_p, then we know that some event from (q_p, q_c), say e', must be enabled in $TS'_p \| TS'_c$. Now, the corresponding event $\varphi(e')$

is also enabled. We can then argue that if any one Σ-event is enabled, then all of them must be enabled (using the nonblocking property of the controller and the fact that it preserves independence of events). Using the properties of a controller, it follows that all events from (q_p, q_c) must be enabled in the controlled plant.

It therefore follows that $\mathcal{U}f(TS'_p \| TS'_c)$ is isomorphic to $\mathcal{U}f(TS'_p)$. We can also show that g maps the $\mathcal{U}f(TS_p)$ fragment of $\mathcal{U}f(TS'_p \| TS'_c)$ to the $\mathcal{U}f(TS_s)$ fragment of $\mathcal{U}f(TS'_s)$. Hence a restriction of g will give a simulation from $\mathcal{U}f(TS_p)$ to $\mathcal{U}f(TS_s)$. $\square$

This leads to the main result of this section.

Theorem 6.3. *The problem of uniformly determining if a pair of finite asynchronous transition systems admits an asynchronous controller is undecidable.*

From our constructions above it is easy to deduce that the problem of uniformly determining if a pair of finite transition systems (TS_p, TS_s) admits a *finite* controller is undecidable. This holds since in the reduction from the undecidable simulation problem to the controller problem, our plant–specification pair is such that it admits a controller iff it admits a finite controller.

Our undecidability result goes through even for the restricted class of asynchronous transition systems that correspond to product transition systems. The main details of the construction of product transition systems whose unfoldings will be the same as we require, are given in Appendix A. Consequently, the undecidability extends to other models, for example, when the plant and specification are presented as labelled 1-safe Petri nets.

Yet another restriction one can consider is the class of asynchronous transition systems where there is an underlying independence over the labels Σ which respects the independence of events, i.e.:

$TS = (Q, E, T, q_{\text{in}}, \varphi, I, \hat{I})$ where $TS = (Q, E, T, q_{\text{in}}, \varphi, I)$ is an asynchronous transition system and $\hat{I} \subseteq \Sigma \times \Sigma$ is irreflexive and symmetric and

$\forall e, e' \in E,\ e\ I\ e' \Rightarrow \varphi(e)\ \hat{I}\ \varphi(e')$.

It is easy to see that the class of systems and specifications used in the undecidability result for simulation fall within this class. Hence, checking existence of simulation for this class is also undecidable. We can also show that checking for the existence of a controller for this class is undecidable. The reduction is from the simulation problem for this class and the details are given in Appendix A.

7. Concluding remarks

In this paper we have studied the controller synthesis problem in a branching time setting. We started with a simple notion of branching time specifications, namely simulations, which can capture simple safety properties. We then considered bisimulation specifications, which can express liveness properties as well, and are a natural extension to simulations. In both instances we have established polynomial time decision procedures as well as polynomial time synthesis procedures which produce polynomial sized controllers whenever controllers exist. We have also shown the undecidability of the problem of checking for the existence of a controller in a simple and natural distributed setting.

Our positive results can be extended in a number of ways. To mention just a few, one could consider plants with internal events as also controllers with internal events. In the case of controllers with internal events one will have to deal with refinement maps instead of simulations and one will have to deal with weak bisimulations instead of (strong) bisimulations. This extension of our work is yet to be done.

A natural extension of this work is to consider the problem where we can handle specifications written in branching-time logics such as CTL, $\forall$-CTL, CTL*, etc. It is hard to pin down a nice logic (say as a sublogic of CTL) which will capture the notion of simulation/bisumulation we have considered.

A challenging extension is suggested by the environment model considered by Kupferman and Vardi in their work on module checking [16, 17]. The idea is that in a branching time setting what one should require is: the controller should prune the system moves in such a way that for *every* pruning of its moves by the environment, the resulting computation tree should meet the specification. We note however that in the presence of simulations and bisimulations, this refined modelling of the environment is immaterial. It is however very relevant when we start considering branching time temporal logics, such as CTL, as specification mechanisms. A variety of interesting and computationally hard problems arise in this new setting and it is the subject of current research.

Turning now to the concurrent setting, there is a natural notion of bisimulation between asynchronous transition systems called the *hereditary history-preserving bisimulation* (see [31, 10]). It has been a long-standing open question whether the problem of checking if there is a hereditary history-preserving bisimulation between a pair of finite asynchronous transition systems is decidable. Jurdziński and Nielsen [11] have recently shown that this problem is undecidable. Their proof makes essential use of the technique we develop in Section 6 to encode grids into unfoldings of asynchronous transition systems. We conjecture that their result can be extended to show that the controller problem for hereditary history-preserving bisimulation is also undecidable.

Appendix A.

Here we will show how to realize the plants and specifications given in Section 6 as a restricted class of asynchronous transition systems – those which can be described as synchronized products of ordinary transition systems.

A *Σ-labeled deterministic synchronized product system* is a structure $(\{P_i\}_{i=1}^{n}, \varphi)$ which consists a set of deterministic transition systems (processes) $P_i = (Q_i, E_i, T_i, q_{\text{in}}^i)$.

The P_i's are supposed to represent concurrent processes which synchronize on common events. φ is a labeling function $\varphi : \bigcup E_i \to \Sigma$. The asynchronous transition system which captures the behaviours of such a system is defined as the following "global" system $TS = (Q, E, T, q_{\text{in}}, \varphi, I)$ where

- $Q = Q_1 \times Q_2 \times \cdots \times Q_n$,
- $E = \bigcup E_i$,
- $q_{\text{in}} = (q^1_{\text{in}}, \ldots, q^n_{\text{in}})$
- $(q_1, \ldots, q_n) \xrightarrow{e} (q'_1, \ldots q'_n)$ iff
  $\forall i : e \in E_i \Rightarrow (q_i \xrightarrow{e} q'_i)$ is in P_i and
  $\forall i : e \notin E_i \Rightarrow q_i = q'_i$,
- $e_1 \, I \, e_2$ iff $\{i \mid e_1 \in E_i\} \cap \{j \mid e_2 \in E_j\} = \emptyset$.

It is easy to see that the system defined above is indeed an asynchronous transition system.

The construction of the plant TS_p: TS_p can be realised as a product of the following processes:

- A process $R = (\{R_0, R_1, R_2\}, \{r_0, r_1, r_2\}, T_R, R_0)$ where T_r has the transitions $R_0 \xrightarrow{r_0} R_1 \xrightarrow{r_1} R_2 \xrightarrow{r_2} R_0$.
- A process $U = (\{U_0, U_1, U_2\}, \{u_0, u_1, u_2\}, U_R, U_0)$ where U_r has the transitions $U_0 \xrightarrow{u_0} U_1 \xrightarrow{u_1} U_2 \xrightarrow{u_2} U_0$.
- For every $i, j \in \{0, 1, 2\}$ we have a process
  $R_{ij} = (\{q_1, q_2, q_3\}, \{ij, r_{i+1}, r_{i-1}\}, T, q_{\text{in}})$ where T has the transitions:
  $q_1 \xrightarrow{r_{i+1}} q_2 \xrightarrow{r_{i-1}} q_1 \;\; q_1 \xrightarrow{r_{i-1}} q_1 \;\; q_2 \xrightarrow{r_{i+1}} q_2 \;\; q_1 \xrightarrow{ij} q_3$
  $q_{\text{in}} = q_1$ if $i = 0$ and $q_{\text{in}} = q_2$ if $i \neq 0$.
- For every $i, j \in \{0, 1, 2\}$ we have a process
  $U_{ij} = (\{q_1, q_2, q_3\}, \{ij, u_{j+1}, u_{j-1}\}, T, q_{\text{in}})$ where T has the transitions:
  $q_1 \xrightarrow{u_{j+1}} q_2 \xrightarrow{u_{j-1}} q_1 \;\; q_1 \xrightarrow{u_{j-1}} q_1 \;\; q_2 \xrightarrow{u_{j+1}} q_2 \;\; q_1 \xrightarrow{ij} q_3$
  $q_{\text{in}} = q_1$ if $j = 0$ and $q_{\text{in}} = q_2$ if $j \neq 0$.
- For every $i, j, i', j' \in \{0, 1, 2\}$ such that ij and $i'j'$ are distinct events and it is not the case that $ij \; I \; i'j'$ (as defined in the construction), we have a process $(\{q_1, q_2, q_3\}, \{ij, i'j'\}, T, q_1)$ where T has the transitions: $q_1 \xrightarrow{ij} q_2$ and $q_1 \xrightarrow{i'j'} q_3$.

The construction of the specification TS_s: TS_s can be realised as a product of the following processes:

- The same processes R and U as in the definition of TS_p
- For every (c, ij)-event in TS_s, we have a process
  $R_{(c,ij)} = (\{q_1, q_2, q_3\}, \{(c, ij), r_{i+1}, r_{i-1}\}, T, q_{\text{in}})$ where T has the transitions:
  $q_1 \xrightarrow{r_{i+1}} q_2 \xrightarrow{r_{i-1}} q_1 \;\; q_1 \xrightarrow{r_{i-1}} q_1 \;\; q_2 \xrightarrow{r_{i+1}} q_2 \;\; q_1 \xrightarrow{(c,ij)} q_3$

$$q_{\text{in}} = \begin{cases} q_1 & \text{if } (i = 0 \text{ and } c = c_{\text{in}}), \\ q_2 & \text{if } (i \neq 0 \text{ or } c \neq c_{\text{in}}). \end{cases}$$

For every (c,ij)-event in TS_s, we have a process
$U_{(c,ij)} = (\{q_1,q_2,q_3\},\{(c,ij),u_{j+1},u_{j-1}\},T,q_{\text{in}})$ where T has the transitions:
$q_1 \xrightarrow{u_{j+1}} q_2 \xrightarrow{u_{j-1}} q_1 \;\; q_1 \xrightarrow{u_{j-1}} q_1 \;\; q_2 \xrightarrow{u_{j+1}} q_2 \;\; q_1 \xrightarrow{(c,ij)} q_3$

$$q_{\text{in}} = \begin{cases} q_1 \text{ if } (j=0 \text{ and } c=c_{\text{in}}), \\ q_2 \text{ if } (j\neq 0 \text{ or } c\neq c_{\text{in}}). \end{cases}$$

- For every pair of distinct events (c,ij) and $(c',i'j')$ in TS_s such that it is not the case that $(c,ij)\ I\ (c',i'j')$ (as defined in the construction), we have a process $(\{q_1,q_2,q_3\},\{(c,ij),(c',i'j')\},T,q_1)$ where T has the transitions:
$q_1 \xrightarrow{(c,ij)} q_2$ and $q_1 \xrightarrow{(c',i'j')} q_3$

It is tedious but routine to verify that the product systems given above do generate the asynchronous transition system we need.

A.1. Undecidability of controller synthesis for a restricted class

Here we consider asynchronous transition systems of the form $TS=(Q,E,T,q_{\text{in}},\varphi,I,\hat{I})$ where $TS=(Q,E,T,q_{\text{in}},\varphi)$ is an asynchronous transition system and $\hat{I}\subseteq\Sigma\times\Sigma$ is an irreflexive symmetric independence relation over Σ which satisfies the following property:
$\forall e_1,e_2\in E,\ e_1\,I\,e_2 \Rightarrow \varphi(e_1)\hat{I}\varphi(e_2)$.

We have already observed that the problem of deciding the existence of a simulation between the unfoldings of two such finite asynchronous transition systems is undecidable. Here, we will show that the controller synthesis problem is also undecidable for this class by reducing the simulation-checking problem to this problem.

Let $TS_p=(Q_p,E_p,T_p,q^p_{\text{in}},\varphi_p,I_p,\hat{I}_p)$ and $TS_s=(Q_s,E_s,T_s,q^s_{\text{in}},\varphi_s,I_s,\hat{I}_s)$ be two such systems. We will construct TS'_p and TS'_s such that there is a simulation from $\mathcal{U}f(TS_p)$ to $\mathcal{U}f(TS_s)$ iff there is a controller for (TS'_p,TS'_s).

We will first expand our alphabet. TS'_p and TS'_s will be Σ'-labelled transition systems where $\Sigma'=\Sigma\cup\Sigma_1$, where $\Sigma_1=\{a' \mid a\in\Sigma\}$. Thus, for every action a in Σ we have introduced a new action a'.

Assume, without loss of generality, that Σ' as well as $2^{\Sigma'}$ are disjoint from Q_p, Q_s, E_p and E_s. Then define $TS'_p=(Q'_p,E'_p,T'_p,q^{p'}_{\text{in}},\varphi'_p,I'_p,\hat{I}'_p)$ as follows:

- $Q'_p=Q_p\cup\{q_a,q_{a'},q_{a,a'} \mid a\in\Sigma\}\cup\{X \mid X \text{ is a nonempty subset of } \Sigma_1\}$.
- $E'_p=E_p\cup\Sigma'\cup\{\tilde{a}' \mid a\in\Sigma\}$.
- $\varphi'_p(e)=\begin{cases}\varphi_p(e) & \text{if } e\in E_p,\\ a & \text{if } e=a\in\Sigma',\\ a' & \text{if } e=\tilde{a}'.\end{cases}$
- $q^{p'}_{\text{in}}=q^p_{\text{in}}$.
- $T'_p=T_p$
  $\cup\{(q_1,a,q_a),(q_1,\tilde{a}',q_{a'}),(q_a,\tilde{a}',q_{a,a'}),(q_{a'},a,q_{a,a'}) \mid q_1\in Q_p, a\in\Sigma\}$
  $\cup\{(q_1,a',\{a'\}) \mid q_1\in Q_p \text{ and } a\in\Sigma\}$

$\cup\{(X,a',Y)\,|\,X,Y$ are non-empty subsets of Σ_1 and $a'\notin X$ and
$Y=X\cup\{a'\}\}$

- $I'_p=I_p\cup\{(a,\tilde{a}')\,|\,a\in\Sigma\}$
  $\{(a',b')\,|\,a\neq b$ and $a,b\in\Sigma\}$.
- $\hat{I}'_p=\hat{I}_p\cup\{(a,a')\,|\,a\in\Sigma\}\cup\{(a',b')\,|\,a\neq b$ and $a,b\in\Sigma\}$.

TS'_s is defined in a similar way. Note that the construction preserves the property required to stay within this class.

Again, using the basic properties of asynchronous controllers, we can prove that any controller for (TS'_p, TS'_s) must be the trivial one which allows all system moves at all times. We can use arguments similar to those in the proof of Theorem 6.3 to show that there is a simulation from $\mathcal{U}f(TS_p)$ to $\mathcal{U}f(TS_s)$ iff there is a controller for (TS'_p, TS'_s).

References

[1] M. Antoniotti, B. Mishra, The supervisor synthesis problem for unrestricted CTL is NP-complete, Tech. Report TR1995-707, New York University, NY, USA, November, 1995.

[2] E. Asarin, O. Maler, A. Pnueli, Symbolic controller synthesis for discrete and timed systems, in: P. Antsaklis, W. Kohn, A. Nerode, S. Sastry (Eds.), Hybrid Systems II, Lecture Notes in Computer Science, vol. 999, Springer, Berlin, 1995, pp. 1–20.

[3] G. Barrett, S. Lafortune, Using bisimulation to solve discrete event control problems, Proc. 1997 Amer. Control Conf., Albuquerque, NM, June 1997, pp. 2337–2341.

[4] M.A. Bednarczyk, Categories of asynchronous transition systems, Ph.D. Thesis, University of Sussex, Technical Report No. 1/88, 1988.

[5] J.R. Büchi, L.H. Landweber, Solving sequential conditions by finite-state strategies, Trans. Amer. Math. Soc. 138 (1969) 295–311.

[6] A. Church, Logic, arithmetic and automata, Proc. Internat. Congr. Math. 1960, Almquist and Wiksells, Uppsala, 1963.

[7] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, The MIT Press, Cambridge, MA, 1992.

[8] V. Diekert, G. Rozenberg (Eds.), The Book of Traces, World Scientific, Singapore, 1995.

[9] B. Jonsson, K.G. Larsen, On the complexity of equation solving in process algebra, Lecture Notes in Computer Science, vol. 493, Springer, Berlin, 1991, pp. 381–396.

[10] A. Joyal, M. Nielsen, G. Winskel, Bisimulation and open maps, Inform. and Comput. 127(2) (1996) 164 –185, A preliminary version appeared in Proc. 8th Annu. IEEE Symp. on Logic in Computer Science, Montreal, Canada, June 1993. IEEE Computer Society Press, Silver Spring, MD, pp. 418–427.

[11] M. Jurdziński, M. Nielsen, Hereditary history preserving bisimilarity is undecidable, BRICS Tech. Report BRICS-RS-99-19, June 1999, Proc. 17th Internat. Symp. on Theoretical Aspects of Computer Science, 2000, to appear.

[12] P.C. Kanellakis, S.A. Smolka, CCS expressions, finite state processes, and three problems of equivalence, Proc. 2nd ACM Symp. on the Principles of Distributed Computing, Montreal, Canada, 1983.

[13] R. Kumar, V.K. Garg, Modeling and Control of Logical Discrete Event Systems, Kluwer Academic Publishers, Dordrecht, 1995.

[14] R. Kumar, V. Garg, S.I. Marcus, On controllability and normality of discrete event dynamical systems, Systems Control Lett. 17 (3) (1991) 157–168.

[15] R. Kumar, M.A. Shayman, Centralized and decentralized supervisory control of nondeterministic systems under partial observation, Proc. 1994 IEEE Conf. on Decision and Control, Orlando, FL, December 1994, pp. 3649–3654.

[16] O. Kupferman, M.Y. Vardi, Module checking, in: Computer Aided Verification, Proc. 8th Internat. Conf., Lecture Notes in Computer Science, vol. 1102, Springer, Berlin, 1996, pp. 75–86.

[17] O. Kupferman, M.Y. Vardi, Module checking revisited, in: Computer Aided Verification, Proc. 9th Internat. Conf., Lecture Notes in Computer Science, vol. 1254, Springer, Berlin, 1997, pp. 36–47.

[18] K.G. Larsen, L. Xinxin, Equation solving using modal transition systems, LICS'90, Philadelphia, PA, USA, 1990, pp. 108–117.

[19] H. Lewis, C.H. Papadimitriou, Elements of the Theory of Computation, Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.

[20] N. Lynch, F. Vaandrager, Forward and backward simulations, Inform. and Comput. 121 (2) (1995) 214–233.

[21] P. Madhusudan, P.S. Thiagarajan, Controllers for discrete event systems via morphisms, Tech. Report TCS-98-02, SPIC Mathematical Institute, Chennai, India, 1998. Available at `http://www.smi.ernet.in`.

[22] R. Milner, in: A Calculus for Communicating Systems, Lecture Notes in Computer Science, vol. 92, Springer, Berlin, 1980.

[23] R. Milner, Communication and Concurrency, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[24] R. Milner, Operational and algebraic semantics of concurrent processes, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Chapter 19, Elsevier, North-Holland, Amsterdam, 1990, pp. 1201–1242.

[25] A. Overkamp, Supervisory control for nondeterministic systems, in: Proc. 11th Internat. Conf. Analysis Optimization Systems, Discrete Event Systems, Sophia-Antipolis, 1994, Lecture Notes in Control and Information Sciences, vol. 199, Springer, New York, 1994.

[26] A. Overkamp, Supervisory control using failure semantics and partial specifications, IEEE Trans. Automat. Control 42 (4) (1997) 498–510.

[27] A. Pnueli, R. Rosner, On the synthesis of a reactive module, Proc. 16th ACM Symp. Principles of Programming Language, 1989, pp. 179–190.

[28] A. Pnueli, R. Rosner, Distributed reactive systems are hard to synthesize, in: 31st Annu. Symp. on Foundations of Computer Science, vol. II, St. Louis, Missouri, 22–24 October, IEEE Press, New York, 1990, pp. 746–757.

[29] P.J.G. Ramadge, W.M. Wonham, The control of discrete event systems, Proc. IEEE 77 (1989) 81–98.

[30] W. Thomas, Finite strategies in regular infinite games, Lecture Notes in Computer Science, vol. 880, Springer, Berlin, 1994, pp. 149–158.

[31] G. Winskel, M. Nielsen, Models for concurrency, in: S. Abramsky, D. Gabby (Eds.), Handbook of Logic in Computer Science, Vol. 3, Oxford University Press, Oxford, UK, 1994.

[32] K.C. Wong, W.M. Wonham, Modular control and coordination of discrete-event systems, Discrete Event Dynamic Systems 6 (3) (1996) 241–273.