Controllers for Discrete Event Systems via Morphisms

P. Madhusudan[1] and P. S. Thiagarajan[2] *

[1] Institute of Mathematical Sciences, C.I.T. Campus, Taramani, Chennai, India
madhu@imsc.ernet.in
[2] SPIC Mathematical Institute, 92 G.N.Chetty Road, T.Nagar, Chennai, India
pst@smi.ernet.in

Abstract. We study the problem of synthesising controllers for discrete event systems. Traditionally this problem is tackled in a linear time setting. Moreover, the desired subset of the computations of the uncontrolled system (often called a plant) is specified by automata theoretic means. Here we formulate the problem in a branching time framework. We use a class of labelled transition systems to model both the plant and the specification. We deploy behaviour preserving morphisms to capture the role of a controller; the controlled behaviour of the plant should be related via a behaviour preserving morphism to the specification at the level of unfoldings. One must go over to unfoldings in order to let the controller use memory of the past to carry out its function.
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 the size of the finite controller, if one exists can be bounded by a polynomial in the sizes of the plant and the specification. Such a controller can also be effectively constructed. We then prove that in a natural concurrent setting, the problem of checking for the existence of a (finite) controller is undecidable.

1 Introduction

We study the problem of synthesizing controllers for discrete event systems. Often, this problem is tackled in a linear time setting. One starts with an open system P, called a *plant*, whose behaviour consists of a collection of sequences L_P over a finite alphabet of actions. The alphabet will consist of two types of actions: those carried out by the system and those carried out by the environment. L_P will usually be a prefix-closed regular set of finite sequences or an ω-regular language of infinite sequences. One then specifies the desired behaviour as a collection of sequences L_S. The synthesis problem is to find a controller C such

* This research has been supported by the IFCPAR project 1502-1.

that $L_{\widehat{P}} \subseteq L_S$ where $L_{\widehat{P}}$ is the constrained language generated by the plant-controller combination. A characteristic restriction on the controller is that it can control only the actions of the system.

This line of work goes back to a realization problem formulated by Church [Ch], later solved elegantly by Büchi and Landweber[BL]. There has been a revival of this area both from control theoretic and computer science perspectives. For a sample of the literature with a computer science flavour, the reader is referred to [PR], [RW], [T] and [AMP] as well as the references therein. From the control-theoretic perspective, the modelling of discrete-event systems (DES's) as automata is often set in the Ramadge-Wonham framework [RW]. Many problems relevant to control theory such as partial-observability, controllability, decentralized and hierarchical control of DES's have been studied ([RW], [KGM], [WW], [KS], [KG]).

The present work departs from these linear-time based studies in a number of ways. Firstly, we study controllers in a branching time setting. Secondly, we do not use automata theoretic means for cutting out the undesired runs of the plant. Instead, we use the same mechanism, namely, a class of transition systems for talking about both plants and specifications. Last but not least, we deploy a type of homomorphisms called c-simulations (control-simulations) to model the separation of good and bad behaviours. Roughly speaking, what can be mapped via a c-simulation into the specification is good and the rest is bad. The role of a controller is then to restrict the system actions in such way that that the controlled behaviour can be related to the specification via a c-simulation.

An important lesson taught by the existing literature is that a rich class of controllers can be obtained by permitting the control to make use of memory of the past to achieve its goal. To bring this in, we will demand that a correct controller should function – possibly using history information – in such a way that that the unfolding of the plant-controller combination can be related to the unfolding of the specification by a c-simulation. Hence the correctness criterion for controllers will be in terms of two potentially infinite objects (even if the plant and the specification are finite objects). As a consequence, the issue of deciding the existence of a controller is non-trivial in the present setting too.

We wish to advocate the use of morphisms in the study of controllers for a number of reasons. One major motivation is that morphisms compose naturally. This gives rise to the hope – as yet unrealized admittedly – that one might be able to apply refinement techniques in the synthesis of controllers. A second motivation is that a good deal of mathematical machinery is available for behavioural simulations of various kinds as evidenced in the systematic account in [LV]. We believe the work initiated in the present paper could be extended to exploit this machinery.

There is a neighbouring body of work in the setting of equation solving in process algebras. A sample of this literature can be found in [LX], [JL]. 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.

There is yet another more recent effort which addresses very similar concerns [O] but which uses the failure semantics model for processes. A pre-order relates the plant to the specification. This relation has a predominantly language-theoretic flavour. However [O] also deals with partial descriptions of the plant by using internal events. We have not incorporated this important feature but we believe our work can be smoothly extended to do so.

Turning now to the main results of the paper, we show that in a sequential setting the question of deciding whether a finite plant-specification pair of transition systems admits a controller is decidable. It turns out that if a controller exists then there exists a finite controller which can be computed in time polynomial in the sizes of the plant and the specification. Moreover, the size of the controller can also be bounded by a similar polynomial. A similar result has been reported for an equation solving problem in a CCS setting [JL]. There it is shown that if the plant is deterministic then the controller can be found in polynomial time. In our framework a plant is modelled as a deterministic transition system over an alphabet of events. But the events carry an additional labelling structure and the morphisms preserve just these labels. The point being, with respect to these labels, the resulting transition system can certainly be non-deterministic.

The second main result of this paper is a negative one. We enrich our transition systems with some concurrency information in a standard way[WN]. We then show that the problem of checking for the existence of a controller in this richer setting is undecidable. We are unaware of a comparable result in the literature concerning the synthesis of controllers for concurrent discrete event systems.

We see this work as a first step towards synthesizing controllers with the help of behaviour preserving morphisms. Our setting is a simple one (in comparison to, say, models based on failure semantics) and yet it captures many of the important features of controllers. A number of extensions need to be carried out before we can evaluate the advantages and the disadvantages of our approach. In particular we need to distinguish between internal and external (observable) actions. We note however that due to the way we model the plant-environment interaction, the distinction between controllable and uncontrollable actions does not apply. Yet another feature that needs to be added is a backward component to our morphisms so that liveness properties can be captured as well. Finally, we must also develop ways of exploiting the compositional aspects of morphisms to contribute to the controller synthesis problem.

In this extended abstract we only present proof sketches. The details appear in the full paper [MT].

2 The Problem Setting

Given our purposes it will be convenient to work with deterministic transition systems with an additional layer of labelling. Through the rest of the paper we fix a finite set Σ of labels and let a,b range over Σ. A Σ-labelled deterministic transition system is a structure $TS = (Q, E, T, q_{in}, \psi)$ where Q is a set of states, E is a set of events and $T \subseteq Q \times E \times Q$ is a deterministic transition relation. In other words, if $(q, e, q'), (q, e, q'') \in T$ then $q' = q''$. Further, $q_{in} \in Q$ is the initial state and $\psi : E \longrightarrow \Sigma$ is a labelling function. We use T to denote the set of transitions because the morphisms we consider will be operating on states as well as transitions. Let $t = (q, e, q') \in T$. Then we set ${}^\bullet t = q$ and $t^\bullet = q'$ and $ev(t) = e$. We will often write $q \xrightarrow{e} q'$ to stand for (q, e, q') and sometimes write $q \overset{e}{\underset{a}{\dashrightarrow}} q'$ to mean that $\psi(e) = a$. Abusing notation, we will also sometimes write $\psi(t) = a$ instead of $\psi(ev(t)) = a$.

Σ is the set of system actions and E is the set of environment actions. The occurrence of the transition $q \overset{e}{\underset{a}{\dashrightarrow}} q'$ is to be viewed as the system at q 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. Note also that it could be the case that $q \overset{e}{\underset{a}{\dashrightarrow}} q'$ and $q \overset{e'}{\underset{b}{\dashrightarrow}} q'$ and consequently 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 "game" is taken from [AMP]. From now on, by a transition system we shall mean a Σ-labelled deterministic transition system. Next we introduce control morphisms (c-morphisms for short) to capture the sense in which a system may be controlled to meet its specification.

Definition 2.1 Let $TS_p = (Q_p, E_p, T_p, q^p_{in}, \psi_p)$ and $TS_s = (Q_s, E_s, T_s, q^s_{in}, \psi_s)$ be a pair of transition systems. A *c-morphism* $f : TS_p \longrightarrow TS_s$ *is a partial function* $f : Q_p \cup T_p \longrightarrow Q_s \cup T_s$ *with* $f(Q_p) \subseteq Q_s$ *and* $f(T_p) \subseteq T_s$ *such that the following conditions are satisfied:*

(C1) $f(q^p_{in}) = q^s_{in}$

(C2) *Suppose* $t_1 \in T_p$ *and* $f(t_1) = t_2$. *Then* $f({}^\bullet t_1) = {}^\bullet t_2$ *and* $f(t_1^\bullet) = t_2^\bullet$. *Moreover,* $\psi_p(t_1) = \psi_s(t_2)$.

(C3) *Suppose* $t_1, t'_1 \in T_p$ *such that* $\psi_p(t_1) = \psi_p(t'_1)$ *and* ${}^\bullet t_1 = {}^\bullet t'_1$. *If* $f(t_1)$ *is defined then* $f(t'_1)$ *is also defined.*

(C4) *Let* $q \in Q_p$ *and* $f(q)$ *be defined. If* $\{t_1 \mid {}^\bullet t_1 = q\} \neq \emptyset$, *then there exists* $t \in T_p$ *such that* ${}^\bullet t = q$ *and* $f(t)$ *is defined.*

The conditions (C1) and (C2) together require f to be a partial homomorphism in a natural way. (C3) is an essential feature of a c-morphism. It demands that TS_s should simulate TS_p without restricting the choices available to the environment in TS_p. The condition (C4) is the so-called *nonblocking* property. If a state q of TS_p has been reached during the simulation and q is not a terminal state, then at least one transition starting from q must be simulated. This ensures that the simulation is not degenerate. Otherwise, one could set $f(q^p_{in}) = q^s_{in}$ and leave everything else undefined.

We note that c-morphisms preserve just labels and w.r.t. the labels the underlying transition systems can be non-deterministic. We have introduced an extra degree of freedom in terms of the events in order facilitate the interaction between the plant and the controller as will become evident soon.

In what follows, we will use the term *c-simulation* to mean a total c-morphism. Note that a c-simulation is just a homomorphism. The notion of a system meeting its specification will be defined at the level of unfoldings. This will permit the system to use memory of the past to interact suitably with its environment.

Let $TS = (Q, E, T, q_{in}, \psi)$ be a transition system. Then $\mathcal{U}f(TS)$, the *unfolding* of TS is the transition system $(\widehat{Q}, \widehat{E}, \widehat{T}, \widehat{q_{in}}, \widehat{\psi})$ where $\widehat{Q} \subseteq Q \times E^*$, $\widehat{E} \subseteq E$ and $\widehat{T} \subseteq \widehat{Q} \times \widehat{E} \times \widehat{Q}$ are the least sets satisfying the following conditions:
• $(q_{in}, \epsilon) \in \widehat{Q}$ (ϵ is the null word).
• Next suppose $(q, \sigma) \in \widehat{Q}$ and $(q, e, q') \in T$. Then $(q', \sigma e) \in \widehat{Q}$, $e \in \widehat{E}$ and $((q, \sigma), e, (q', \sigma e)) \in \widehat{T}$.
We set $\widehat{q_{in}} = (q_{in}, \epsilon)$ and define $\widehat{\psi}$ to be ψ restricted to $\widehat{E}$.

It is easy to check that $\mathcal{U}f(TS)$ is a Σ-labelled transition system. One could have defined $\widehat{Q}$ in terms of E^* alone but the present formulation will be easier to deal with.

Let (TS_p, TS_s) be a pair of transition systems. Then TS_p *meets the specification* TS_s iff there exists a c-simulation from $\mathcal{U}f(TS_p)$ to $\mathcal{U}f(TS_s)$.

In the examples to follow, we will assume that each event is involved in exactly one transition. Hence we will display only the Σ-members involved.

Example 2.1.

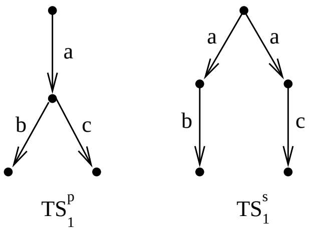

Fig. 1

There are two c-morphisms from TS_1^p to TS_1^s but there is no c-simulation. This illustrates the branching nature of the specification.

Example 2.2.

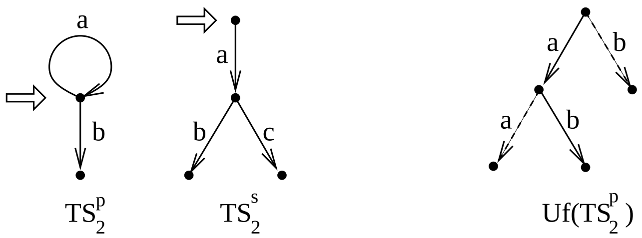

Fig. 2

There is no c-morphism from TS_2^p to TS_2^s but there will be a (unique) c-morphism from $\mathcal{U}f(TS_2^p)$ to $\mathcal{U}f(TS_2^s)$. This is shown above on the picture of (a portion of) the unfolding of TS_2^p with the dotted parts denoting undefinedness.

Note that there is no c-simulation from $\mathcal{U}f(TS_2^p)$ to $\mathcal{U}f(TS_2^s)$. Thus a system may not meet its specification but it might be controlled so that the resulting behaviour meets the specification. In our example, the correct control will permit a initially but not b. After the first a, it will permit a b but no more a's. It is in this sense that memory can be used to achieve the desired control. This example also illustrates that the specification could contain "unused" parts since there is no backward component in our notion of a simulation. Loosely speaking, in the present framework we cater only for universal branching-time properties.

To define controllers, we need the notion of a product of two transition systems. Let $TS_i = (Q_i, E_i, T_i, q_{in}^i, \psi_i)$, $i = 1, 2$ be a pair of transition systems. Then $TS_1 || TS_2$ is the (synchronized) product of TS_1 and TS_2 and it is defined to be the structure (Q, E, T, q_{in}, ψ) where $Q = Q_1 \times Q_2$, $E = E_1 \cup E_2$ and T is the least subset of $Q \times E \times Q$ satisfying the following conditions:

- Suppose $(q_1, q_2) \in Q$, $(q_1, e, q_1') \in T_1$ and $e \in E_1$ but $e \notin E_2$. Then $((q_1, q_2), e, (q_1', q_2)) \in T$
- Suppose $(q_1, q_2) \in Q$, $(q_2, e, q_2') \in T_2$ and $e \in E_2$ but $e \notin E_1$. Then $((q_1, q_2), e, (q_1, q_2')) \in T$
- Next suppose $(q_1, q_2) \in Q$, $(q_1, e, q_1') \in T_1$, $(q_2, e, q_2') \in T_2$ and $\psi_1(e) = \psi_2(e)$. Then $((q_1, q_2), e, (q_1', q_2')) \in T$.

We set $q_{in} = (q_{in}^1, q_{in}^2)$ and define ψ in the obvious way.
Again it is easy to check that $TS_1 || TS_2$ is a Σ-labelled deterministic transition system.
We can now define controllers.

Definition 2.2 Let TS_p, TS_s, TS_c be three transition systems with $TS_p = (Q_p, E_p, T_p, q_{in}^p, \psi_p)$ and $TS_c = (Q_c, E_c, T_c, q_{in}^c, \psi_c)$. Then the transition system TS_c is a controller for the pair (TS_p, TS_s) iff the following conditions are satisfied:
Let $TS_p || TS_c = (\widetilde{Q}, \widetilde{E}, \widetilde{T}, \widetilde{q_{in}}, \widetilde{\psi})$

(CT1) $E_c = E_p$ and $\psi_p = \psi_c$.
(CT2) Suppose (q_1^p, q_1^c) is reachable from the initial state in $TS_p || TS_c$ and $((q_1^p, q_1^c), e, (q_2^p, q_2^c)) \in \widetilde{T}$ and $(q_1^p, e', q_3^p) \in T_p$ with $\psi_p(e) = \psi_p(e')$. Then there exists $q_3^c \in Q_c$ such that $((q_1^p, q_1^c), e', (q_3^p, q_3^c)) \in \widetilde{T}$ (and hence $(q_1^c, e', q_3^c) \in T_c$).
(CT3) Let (q_1^p, q_1^c) be reachable from the initial state in $TS_p || TS_c$ and $(q_1^p, e, q_2^p) \in T_p$. Then there must exist $e' \in E_p$ such that there is a transition of the form $((q_1^p, q_1^c), e', (q_2^p, q_2^c))$ in $\widetilde{T}$.
(CT4) $TS_p || TS_c$ meets the specification TS_s

The condition (CT1) demands that the system and the controller be tightly coupled to each other; there are no "autonomous" transitions. The condition

(CT2) demands that TS_c should only restrict the choices of the system. If at a state, it permits one a-move of the system, then it should permit all the a-moves of the system at this state. (CT3) demands that the controller should not block the system completely. The role of (CT4) should be clear.

The use of the deterministic event-based transition relation can be now brought out. It lets the controller record the history of the plant as a sequence of events and from this record determine the current state of the plant.

Our notion of a controller is rather strict. One could weaken (CT1) by demanding that the set of events of the controller is included in (and not necessarily equal to) the set of events of the plant. With minor modifications all our results will go through. Ideally one should simply drop (CT1). But in this case one should model the moves that involve the controller alone as internal moves. We however do not wish to deal with this complication at this stage.

3 c-Morphisms and controllers

As a step towards proving our main decidability result we will show that one can go back and forth between controllers and c-morphisms at the level of unfoldings.

Lemma 3.1 *Let $TS_p = (Q_p, E_p, T_p, q_{in}^p, \psi_p)$ and $TS_s = (Q_s, E_s, T_s, q_{in}^s, \psi_s)$ be a pair of transition systems. Let $f : \mathcal{U}f(TS_p) \to \mathcal{U}f(TS_s)$ be a c-morphism. Then there exists a controller for the pair (TS_p, TS_s).*

Proof

Assume that $\mathcal{U}f(TS_p) = (\widehat{Q_p}, \widehat{E_p}, \widehat{T_p}, \widehat{q_{in}^p}, \widehat{\psi_p})$. Define $TS_c = (Q_c, E_c, T_c, q_{in}^c, \psi_c)$ to be the transition system induced by the the domain of f. In other words, with $dom(f)$ as the domain of the partial function f, we require: $Q_c = dom(f) \cap \widehat{Q_p}$, $T_c = dom(f) \cap \widehat{T_p}$, $E_c = E_p$, Further, $q_{in}^c = \widehat{q_{in}^p}$, and $\psi_c = \psi_p$.

It is routine to verify that TS_c is a Σ-labelled transition system and that it satisfies the first three conditions $(CT1, CT2, CT3)$ for being a controller. To show that there exists a c-simulation from $\mathcal{U}f(TS_p \| TS_c)$ to $\mathcal{U}f(TS_s)$, it suffices to prove that $\mathcal{U}f(TS_p \| TS_c)$ is isomorphic to TS_c. This is a tedious but straightforward exercise. ∎

Lemma 3.2 *Let $TS_x = (Q_x, E_x, T_x, q_{in}^x, \psi_x)$, $x \in \{p, s, c\}$ be three transition systems such that TS_c is a controller for (TS_p, TS_s). Then there exists a c-morphism from $\mathcal{U}f(TS_p)$ into $\mathcal{U}f(TS_s)$.*

Proof Since TS_c is a controller for (TS_p, TS_s), there exists a c-simulation f from $\mathcal{U}f(TS_p \| TS_c)$ into $\mathcal{U}f(TS_s)$. Let $\mathcal{U}f(TS_x) = (\widehat{Q}_x, \widehat{E}_x, \widehat{T}_x, \widehat{q_{in}^x}, \widehat{\psi}_x)$, $x \in \{p, s\}$. Further, let $\mathcal{U}f(TS_p \| TS_c) = (\widehat{Q}, \widehat{E}, \widehat{T}, \widehat{q_{in}}, \widehat{\psi})$. Define the partial function $g : \widehat{Q_p} \cup \widehat{T_p} \to \widehat{Q_s} \cup \widehat{T_s}$ as follows:

- Let $(q_1, \sigma) \in \widehat{Q_p}$. Then $g((q_1, \sigma)) = (q_2, \tau)$ if there exists $q_3 \in Q_c$ such that $((q_1, q_3), \sigma) \in \widehat{Q}$ and $f((q_1, q_3), \sigma) = (q_2, \tau)$. Otherwise $g((q_1, \sigma))$ is undefined.

– Let $\widehat{t_1} = ((q_1, \sigma), e, (q_2, \sigma e)) \in \widehat{T_p}$. Then $g(\widehat{t_1}) = \widehat{t_2} \in \widehat{T_s}$ if there exists $q_3, q_4 \in Q_c$ such that $\widehat{t} = (((q_1, q_3), \sigma), e, ((q_2, q_4), \sigma e)) \in \widehat{T}$ and $f(\widehat{t}) = \widehat{t_2}$. Otherwise $g(\widehat{t_1})$ is undefined.

Again it is straightforward to check that g is a c-morphism from $\mathcal{U}f(TS_p)$ into $\mathcal{U}f(TS_s)$. ■

In general, the controller yielded by the proof of Lemma 3.1 may not be realizable as a finite controller, even if both TS_p and TS_s are finite. This is because the c-morphism $f : \mathcal{U}f(TS_p) \rightarrow \mathcal{U}f(TS_s)$ one starts with may be irregular. This would prevent one from folding down the domain of the c-morphism into a finite transition system. Our goal now is to show that one can always extract a regular c-morphism in such a situation. This will lead to the synthesis of a finite controller.

4 Regular c-morphisms and finite controllers

We start with the notion of regular c-morphisms. Let $TS_i = (Q_i, E_i, T_i, q^i_{in}, \psi_i)$, $i = 1, 2$ be a pair of transition systems and let $f : TS_1 \rightarrow TS_2$ be a c-morphism. An *f-congruence* is an equivalence relation $\approx \subseteq Q_1 \times Q_1$ which satisfies the following two conditions:
(CG1) Suppose $q_1 \approx q'_1$ and $f(q_1)$ is defined. Then $f(q'_1)$ is also defined.
(CG2) Suppose $q_1 \approx q'_1$ and $t_1 = q_1 \xrightarrow[a]{e} q_2$ is in T_1 and $f(t_1)$ is defined. Then there exists t'_1, q'_2 such that $t'_1 = q'_1 \xrightarrow[a]{e} q'_2$ is in T_1 and $q_2 \approx q'_2$ and $f(t'_1)$ is also defined.

The c-morphism f is said to be *regular* iff there exists an f-congruence of finite index (i.e. $\approx$ partitions Q into a finite number of equivalence classes). The next result is the key to solving the synthesis problem.

Lemma 4.1 *Let $TS_p = (Q_p, E_p, T_p, q^p_{in}, \psi_p)$ and $TS_s = (Q_s, E_s, T_s, q^s_{in}, \psi_s)$ be a pair of* finite *transition systems. Let $f : \mathcal{U}f(TS_p) \rightarrow \mathcal{U}f(TS_s)$ be a c-morphism. Then there exists a regular c-morphism $g : \mathcal{U}f(TS_p) \rightarrow \mathcal{U}f(TS_s)$.*

Proof First we associate the finite edge-labelled directed graph $G_{ps} = (X, \rightarrow)$ with (TS_p, TS_s):
– $X = Q_p \times Q_s$
– $\rightarrow$ is the least subset of $X \times (E_p \times E_s) \times X$ satisfying:
If $q_1 \xrightarrow{e_1} q'_1$ is in T_p and $q_2 \xrightarrow{e_2} q'_2$ is in T_s, and $\psi_p(e_1) = \psi_s(e_2)$, then $((q_1, q_2), (e_1, e_2), (q'_1, q'_2))$ is in $\rightarrow$.

$(Y, \Rightarrow)$ is said to be a subgraph of G_{ps} in case $Y \subseteq X$ and $\Rightarrow \subseteq (Y \times (E_p \times E_s) \times Y) \cap \rightarrow$.
The subgraph $(Y, \Rightarrow)$ is said to be *good* in case it satisfies the following conditions:
(G1) $(q^p_{in}, q^s_{in}) \in Y$
(G2) Suppose $(q_1, q_2) \overset{(e_1, e_2)}{\Longrightarrow} (q'_1, q'_2)$ and $q_1 \xrightarrow{e'_1} q''_1$ is in T_p with $\psi_p(e_1) = \psi_p(e'_1)$. Then there exists $e'_2 \in E_s$ and $q''_2 \in Q_s$ such that $(q_1, q_2) \overset{(e'_1, e'_2)}{\Longrightarrow} (q''_1, q''_2)$.

(G3) Suppose $(q_1, q_2) \in Y$ and q_1 is not a terminal state in TS_p. In other words, there exists $q_1 \xrightarrow{e} q$ in T_p. Then there exist $q_1' \in Q_p$ and $q_2' \in Q_s$ and $(e_1, e_2) \in E_p \times E_s$ such that $(q_1, q_2) \overset{(e_1,e_2)}{\Longrightarrow} (q_1', q_2')$

Now, consider the c-morphism $f : \mathcal{U}f(TS_p) \to \mathcal{U}f(TS_s)$. Define the subgraph $(Y, \Rightarrow)$ of G_{ps} induced by f as:

- $(q_1, q_2) \in Y$ iff $\exists \sigma_1 \in E_p^*, \exists \sigma_2 \in E_s^*$ such that $f((q_1, \sigma_1)) = (q_2, \sigma_2)$
- $(q_1, q_2) \overset{(e_1,e_2)}{\Longrightarrow} (q_1', q_2')$ iff $\exists \sigma_1 \in E_p^*, \exists \sigma_2 \in E_s^*$ such that $\widehat{t_1} = ((q_1, \sigma_1), e_1, (q_1', \sigma_1 e_1))$ is in the set of transitions of $\mathcal{U}f(TS_p)$ and moreover, $f(\widehat{t_1}) = ((q_2, \sigma_2), e_2, (q_2', \sigma_2 e_2))$.

It is not difficult to verify that $(Y, \Rightarrow)$ is a good subgraph of G_{ps}. Next, we extract a subgraph of $(Y, \Rightarrow)$, say $(Z, \rightsquigarrow)$ which is deterministic in the first component in the sense that if $(q_1, q_2) \overset{(e_1,e_2)}{\rightsquigarrow} (q_1', q_2')$ and $(q_1, q_2) \overset{(e_1,e_2')}{\rightsquigarrow} (q_1', q_2'')$, then $e_2 = e_2'$ and hence $q_2' = q_2''$. We can extract such a $(Z, \rightsquigarrow)$ from $(Y, \Rightarrow)$ by first fixing a linear order over E_s. Then we set $Z = Y$ and define $\rightsquigarrow$ to be the least subset of $\Rightarrow$ satisfying:

Suppose $(q_1, q_2) \overset{(e_1,e_2)}{\Longrightarrow} (q_1', q_2')$ and there is no edge of the form $(q_1, q_2) \overset{(e_1,e_2')}{\Longrightarrow} (q_1', q_2'')$ such that e_2' is strictly less than e_2 in the linear order we have fixed for E_s. Then $(q_1, q_2) \overset{(e_1,e_2)}{\rightsquigarrow} (q_1', q_2')$. Again it is easy to check that $(Z, \rightsquigarrow)$ is a good subgraph if G_{12}.

We now use $(Z, \rightsquigarrow)$ to induce a regular c-morphism from $\mathcal{U}f(TS_p)$ into $\mathcal{U}f(TS_s)$. To this end, define the set of *computation pairs* $CP \subseteq E_p^* \times E_s^*$ and an accompanying function $\delta : CP \to Z$ as follows:

- $(\epsilon, \epsilon) \in CP$ and $\delta((\epsilon, \epsilon)) = (q_{in}^p, q_{in}^s)$.
- Suppose $(\sigma_1, \sigma_2) \in CP$, $\delta((\sigma_1, \sigma_2)) = (q_1, q_2)$ and $(q_1, q_2) \overset{(e_1,e_2)}{\rightsquigarrow} (q_1', q_2')$. Then $(\sigma_1 e_1, \sigma_2 e_2) \in CP$ and $\delta((\sigma_1 e_1, \sigma_2 e_2)) = (q_1', q_2')$.

Note that by the definition of $(Z, \rightsquigarrow)$, if $(\sigma_1, \sigma_2) \in CP$ and $(\sigma_1, \sigma_2') \in CP$ then $\sigma_2 = \sigma_2'$. We can now define the required regular c-morphism $g : \mathcal{U}f(TS_p) \to \mathcal{U}f(TS_s)$. Let $\mathcal{U}f(TS_p) = (\widehat{Q_p}, \widehat{E_p}, \widehat{T_p}, \widehat{q_{in}^p}, \widehat{\psi_p})$ and $\mathcal{U}f(TS_s) = (\widehat{Q_s}, \widehat{E_s}, \widehat{T_s}, \widehat{q_{in}^s}, \widehat{\psi_s})$. Suppose $(q_1, \sigma_1) \in \widehat{Q_p}$. Then $g((q_1, \sigma_1))$ is defined only if there exists $\sigma_2 \in E_s^*$ such that $(\sigma_1, \sigma_2) \in CP$. In this case, $g((q_1, \sigma_1)) = (q_2, \sigma_2)$ where $\delta((\sigma_1, \sigma_2)) = (q_1, q_2)$.

Next, consider some transition $\widehat{t_1} = (q_1, \sigma_1) \xrightarrow{e_1} (q_1', \sigma_1 e_1)$ in $\widehat{T_p}$. Then $g(\widehat{t_1})$ is defined provided there exists $\sigma_2 e_2 \in E_s^*$ such that $(\sigma_1 e_1, \sigma_2 e_2) \in CP$. In this case, $g(\widehat{t_1}) = (q_2, \sigma_2) \xrightarrow{e_2} (q_2', \sigma_2 e_2)$ where $\delta(\sigma_1, \sigma_2) = (q_1, q_2)$ and $\delta(\sigma_1 e_1, \sigma_2 e_2) = (q_1', q_2')$. It is easy to verify that g is well-defined and that it is a c-morphism. To check that g is regular, we define $\approx$ to be the subset of $\widehat{Q_p} \times \widehat{Q_p}$ such that $(q_1, \sigma_1) \approx (q_1', \sigma_1')$ if *one* of the following conditions are satisfied:

(i) Neither there exists $\sigma_2 \in E_s^*$ such that $(\sigma_1, \sigma_2) \in CP$ nor does there exist $\sigma_2' \in E_s^*$ such that $(\sigma_1', \sigma_2') \in CP$

OR

(ii) There exist $\sigma_2, \sigma'_2 \in E_s^*$ such that (σ_1, σ_2), $(\sigma'_1, \sigma'_2) \in CP$. Moreover, $\delta((\sigma_1, \sigma_2)) = \delta((\sigma'_1, \sigma'_2))$.

Clearly $\approx$ is an equivalence relation. Further, the number of equivalence classes induced by $\approx$ is at most $| Q_p | \cdot | Q_s | + 1$. A detailed examination of the various definitions will lead to the fact that $\approx$ is a g-congruence. ■

It turns out that regular c-morphisms and finite controllers are strongly related to each other.

Lemma 4.2 *Let $f : \mathcal{U}f(TS_p) \rightarrow \mathcal{U}f(TS_s)$ be a regular c-morphism where TS_p and TS_s are finite transition systems. Then there exists a finite controller for (TS_p, TS_s).*

Proof Let $\mathcal{U}f(TS_p) = (\widehat{Q_p}, \widehat{E_p}, \widehat{T_p}, \widehat{q}^p_{in}, \widehat{\psi_p})$, $\mathcal{U}f(TS_s) = (\widehat{Q_s}, \widehat{E_s}, \widehat{T_s}, \widehat{q}^s_{in}, \widehat{\psi_s})$ and $\approx \subseteq \widehat{Q_p} \times \widehat{Q_p}$ be an f-congruence of finite index. In what follows, $[q, \sigma]$ will denote the $\approx$ *$-$equivalence* class containing (q, σ) where $(q, \sigma) \in \widehat{Q_p}$. Define the transition system $TS_c = (Q_c, E_c, T_c, q^c_{in}, \psi_c)$ via:

- $Q_c = \{[q, \sigma] \mid (q, \sigma) \in \widehat{Q_p}\}$
- $E_c = E_p$ where E_p is the set of events of TS_p.
- Let $([q, \sigma], [q', \sigma']) \in Q_c$ and $e \in E_c$. Then $([q, \sigma], e, [q', \sigma']) \in T_c$ if there exist $(q_1, \sigma_1) \in [q, \sigma]$ and $(q'_1, \sigma'_1) \in [q', \sigma']$ such that $\widehat{t_1} = ((q_1, \sigma_1), e, (q'_1, \sigma'_1))$ is in $\widehat{T_p}$ and $f(\widehat{t_1})$ is defined.
- $q^c_{in} = (q^p_{in}, \epsilon)$
- $\psi_c = \psi_p$ where ψ_p is the labelling function of TS_p.

It is tedious but routine to prove that TS_c is a controller for (TS_p, TS_c). Clearly, both Q_c and E_c are finite sets. ■

The results of this section can be summarized as follows:

Theorem 4.3 *Let TS_p and TS_s be a pair of finite transition systems. Then the following statements are equivalent:*

(i) *There is a c-morphism from $\mathcal{U}f(TS_p)$ into $\mathcal{U}f(TS_s)$*
(ii) *There is a regular c-morphism from $\mathcal{U}f(TS_p)$ into $\mathcal{U}f(TS_s)$.*
(iii) *There is a controller for (TS_p, TS_s)*
(iv) *There is a finite controller for (TS_p, TS_s).*

Proof Follows at once from Lemmas 3.1, 3.2, 4.1 and 4.2.

5 The synthesis procedure

The proof of Lemma 4.1 yields a procedure for solving the controller synthesis problem for finite transition systems. First we solve a decision problem.

Theorem 5.1 *There is a decision procedure which takes as input two finite transition systems $TS_p = (Q_p, E_p, T_p, q^p_{in}, \psi_p)$, $TS_s = (Q_s, E_s, T_s, q^s_{in}, \psi_s)$ and decides whether or not there exists a controller for (TS_p, TS_s).*

Proof Recall the finite edge-labelled directed graph G_{ps} defined in the proof of Lemma 4.1. It is clear from the proof of Lemma 4.1 and Theorem 4.3 that (TS_p, TS_s) has a controller iff G_{ps} has a good subgraph. Our decision procedure will therefore check for the existence of a good subgraph of G_{ps}. We set $G_0 = G_{ps}$ and construct a sequence of graphs $G_0, G_1, \ldots G_n$ up to the stage at which $G_n = G_{n+1}$.

Assume that G_i, $i \geq 0$ has been computed. G_{i+1} is obtained from G_i by applying one of the following pruning steps to G_i. If neither step can be applied then we set $G_{i+1} = G_i$ and stop.

(i) Let $G_i = (X_i, \longrightarrow_i)$. 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 $\psi_p(e_1) = \psi_p(e_1')$. 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 this procedure must terminate after a finite number of steps. It is now routine to verify that G_{ps} has a good subgraph iff $(q_{in}^p, q_{in}^s) \in X_n$ where $G_n = (X_n, \longrightarrow_n)$. Thus $(q_{in}^p, q_{in}^s) \in X_n$ iff (TS_p, TS_s) has a controller. ■

Corollary 5.2 *Let $TS_p = (Q_p, E_p, T_p, q_{in}^p, \psi_p)$ and $TS_s = (Q_s, E_s, T_s, q_{in}^s, \psi_s)$ be a pair of finite transition systems. Let $\mid Q_p \mid = n_1$, $\mid Q_s \mid = n_2$, $\mid E_p \mid = k_1$ and $\mid E_s \mid = k_2$. 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 $G_0 = G_{ps}$ 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 that 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 5.3 *Let $TS_p = (Q_p, E_p, T_p, q_{in}^p, \psi_p)$ and $TS_s = (Q_s, E_s, T_s, q_{in}^s, \psi_s)$ 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 Lemma 4.1, let n be the least integer such that $G_n = G_{n+1}$. Assume that $G_n = (X_n, \longrightarrow_n)$ and that $(q_{in}^p, q_{in}^s) \in X_n$. 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, starting from G_n, one can extract a good subgraph $(Y, \Rightarrow)$ of G_n which is deterministic in the first component as done in the proof of Lemma 4.1. This can be done in time which is linear in the size of G_n. Define now the transition system $TS_c = (Q_c, E_c, T_c, q_{in}^c, \psi_c)$ via:

- $Q_c = Y$; $E_c = E_p$
- $((q_1, q_2), e_1, (q_1', q_2')) \in T_c$ iff and there exists $e_2 \in E_s$ such that there is an edge $(q_1, q_2) \overset{(e_1,e_2)}{\Longrightarrow} (q_1', q_2')$ in $(Y, \Rightarrow)$.
- $q_{in}^c = (q_{in}^p, q_{in}^s)$
- $\psi_c = \psi_p$.

It is once again a tedious but straightforward exercise to verify that TS_c is a controller for (TS_p, TS_s). ■

6 A Negative Result in a Concurrent Setting

Transition systems can be augmented with some concurrency information to model distributed computations. Here we consider one well-established variant called asynchronous transition systems[WN]. We will show that the problem of deciding whether there exists a (finite) controller for a pair of asynchronous transition systems is undecidable.

A Σ-labelled deterministic asynchronous transition system is a structure $TS = (Q, E, T, q_{in}, \psi, I)$ where (Q, E, T, q_{in}, ψ) 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. c-morphisms will now be required to preserve the independence of events. Let $TS_1 = (Q_1, E_1, T_1, q_{in}^1, \psi_1, I_1)$ and $TS_2 = (Q_2, E_2, T_2, q_{in}^2, \psi_2, I_2)$ be a pair of asynchronous transition systems. Then an asynchronous c-morphism $f : TS_1 \to TS_2$ is a c-morphism from $(Q_1, E_1, T_1, q_{in}^1, \psi_1)$ to $(Q_2, E_2, T_2, q_{in}^2, \psi_2)$ 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 $f(t_4) = (p_2, e_1', p')$.

An asynchronous c-simulation is just a total asynchronous c-morphism. From now on we will often drop the adjective "asynchronous" in referring to asynchronous c-morphisms and c-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 will be used to group together different interleavings of the same partially ordered stretch of behaviour.

Let $TS = (Q, E, T, q_{in}, \psi, I)$ 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 $\mathcal{U}f(TS) = (\widehat{Q}, \widehat{E}, \widehat{T}, \widehat{q}_{in}, \widehat{\psi}, \widehat{I})$ via:

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

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 ψ_i as the labelling function of TS_i. Then $TS_1 \parallel TS_2$ 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 $TS_1 \parallel TS_2$ is defined as done in Section 2. Again, it should be clear that $TS_1 \parallel TS_2$ 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 there exists a c-simulation from $\mathcal{U}f(TS_p \parallel TS_c)$ into $\mathcal{U}f(TS_s)$. It is easy to show that (TS_p, TS_s) admits an asynchronous controller iff there exists an asynchronous c-morphism from $\mathcal{U}f(TS_p)$ into $\mathcal{U}f(TS_s)$.

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 [LP] which is known to be undecidable. 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 $\mathcal{CP} = (C, c_{in}, R, U)$ where C is a finite set of colours, $c_{in} \in C$ is a distinguished initial colour and $R : C \longrightarrow 2^C$ and $U : C \longrightarrow 2^C$ are two functions. A solution to $\mathcal{CP}$ is a map $col : \omega \times \omega \longrightarrow C$ (ω is the set of natural numbers) which satisfies:

$col(0, 0) = c_{in}$

$\forall \; (m, n) \in \omega \times \omega. \; col(m + 1, n) \in R(col(m, n))$ and $col(m, n + 1) \in U(col(m, n))$.

For each instance $\mathcal{CP}$ of a colouring problem we will construct a pair of finite asynchronous transition systems (TS_p, TS_s) such that $\mathcal{CP}$ has a solution iff there exists an asynchronous *c-simulation* from $\mathcal{U}f(TS_p)$ into $\mathcal{U}f(TS_s)$. 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 c-simulation from $\mathcal{U}f(TS_p)$ into $\mathcal{U}f(TS_s)$ iff there exists an asynchronous *c-morphism* from $\mathcal{U}f(TS'_p)$ into $\mathcal{U}f(TS'_s)$. This will lead to the desired result.

Through the rest of the section fix an instance of the colouring problem $\mathcal{CP} = (C, c_{in}, R, U)$ 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 display $\mathcal{U}f(TS_p)$ and $\mathcal{U}f(TS_s)$ instead of TS_p and TS_s. Both $\mathcal{U}f(TS_p)$ and $\mathcal{U}f(TS_s)$ will have only a finite number of isomorphism classes. Hence it will be easy to conclude that they could arise as unfoldings of two finite asynchronous transition systems. The definitions of TS_p and TS_s can be found in the full paper [MT].

The main part of $\mathcal{U}f(TS_p)$ 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 $\mathcal{U}f(TS_p)$. This is shown below. We display only the events concerned and not their labels. We will deal with the labels later.

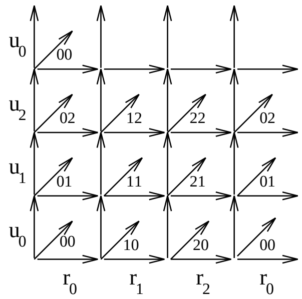

Fig. 4: The main grid

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 the independence relation I_1 will demand:
$ij \ I_p \ r_{i'}$ iff $i = i'$ and $ij \ I_p \ u_{j'}$ 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.5.

Note that once an event of type ij is performed then 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 c-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 $(c_{in}, 00)$ 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 $i = i'$ and $u_j \ I_s \ (c, i'j')$ 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 $c' \in U(c)$).

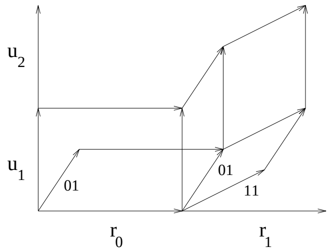

Fig. 5

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. Using the properties of a c-simulation, one can now prove that there exists a c-simulation from $\mathcal{U}f(TS_p)$ into $\mathcal{U}f(TS_s)$ iff $\mathcal{CP}$ has a solution.

Next we show that the problem of checking for a c-simulation reduces to that of checking for a c-morphism. Given TS_p and TS_s, we construct TS'_p and TS'_s such that there exists a c-simulation from $\mathcal{U}f(TS_p)$ into $\mathcal{U}f(TS_s)$ iff there exists a c-morphism from $\mathcal{U}f(TS'_p)$ into $\mathcal{U}f(TS'_s)$. Let $TS_p = (Q_p, E_p, T_p, q^p_{in}, \psi_p, I_p)$ and $TS_s = (Q_s, E_s, T_s, q^s_{in}, \psi_s, I_s)$. Assume without loss of generality that Σ is disjoint from Q_p, Q_s, E_p and E_s. Then $TS'_p = (Q'_p, E'_p, T'_p, q^{p'}_{in}, \psi'_p, I'_p)$ is defined as follows.
$Q'_p = Q_p \cup \{X \mid X$ is a non-empty subset of $\Sigma\}$
$E'_p = E_p \cup \Sigma$.
$\psi'_p(e_1) = \psi_p(e_1)$, if $e_1 \in E_p$; $\quad \psi'_p(a) = a$, if $a \in \Sigma$
$q^{p'}_{in} = q^p_{in}$.
$T'_p = T_p \cup \ \{(q_1, a, \{a\}) \mid q_1 \in Q_p$ and $a \in \Sigma\} \ \cup$
$\{(X, a, Y) \mid X, Y$ are non-empty subsets of Σ and $a \notin X$ and
$Y = X \cup \{a\}\}$
$I'_p = I_p \cup \{(a, b) \mid a \neq b$ and $a, b \in \Sigma\}$.

TS'_s is defined in a similar way. Again using the basic properties of c-morphisms, it is not difficult to prove that there exists a c-morphism from $\mathcal{U}f(TS'_p)$ into $\mathcal{U}f(TS'_s)$ iff there exists c-simulation from $\mathcal{U}f(TS_p)$ into $\mathcal{U}f(TS_s)$. This leads to the main result of this section.

Theorem 6.1 *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. Indeed one may even assume that (TS_p, TS_s) arise from labelled 1-safe Petri nets or synchronized products of ordinary transition systems and the undecidability result will still hold.

References

[AMP] E. Asarin, O. Maler, A. Pnueli: Symbolic Controller Synthesis for Discrete and Timed Systems, in P. Antsaklis, W. Kohn, A. Nerode and S. Sastry (Eds.), *Hybrid Systems II*, 1–20, LNCS 999, Springer, 1995

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

[Ch] A. Church: Logic, arithmetic and automata, *Proc. Intern. Congr. Math. 1960*, Almquist and Wiksells, Uppsala (1963).

[JL] B. Jonsson and K. G. Larsen: On the complexity of equation solving in process algebra, *LNCS 493*, Springer-Verlag (1991) 381–396

[KG] Ratnesh Kumar, Vijay K. Garg: Modeling and Control of Logical Discrete Event Systems, Kluwer Academic Publishers, 1995

[KGM] Ratnesh Kumar, Vijay Garg, Steven I. Marcus: On Controllability and Normality of Discrete Event Dynamical Systems. *Systems and Control Letters*, 17(3):157–168, 1991.

[KS] Ratnesh Kumar, Mark A. Shayman: Centralized and Decentralized Supervisory Control of Nondeterministic Systems Under Partial Observation. In *Proceedings of 1994 IEEE Conference on Decision and Control*, Orlando, FL, December 1994. 3649–3654

[LX] K. G. Larsen and L. Xinxin: Equation solving using modal transition systems, *LICS'90*, Philadelphia, PA, U.S.A., (1990) 108–117.

[LP] H. Lewis and C. H. Papadimitriou: Elements of the theory of computation, *Prentice Hall*, New Jersey, U.S.A. (1981).

[LV] N. Lynch and F. Vaandrager: Forward and backward simulations, *Information and Computation, 121 (2)* (1995) 214–233,

[MT] P. Madhusudan and P. S. Thiagarajan: Controllers for Discrete Event Systems via Morphisms, Technical Report TCS-98-02, SPIC Mathematical Institute, Chennai, India (1998). Available at `http://www.smi.ernet.in`.

[O] A. Overkamp: Supervisory control using failure semantics and partial specifications, *IEEE Trans. on Automatic Control, Vol. 42, No. 4* (1997) 498–510

[PR] A. Pnueli and R. Rosner: On the synthesis of a reactive module, *Proc. 16th ACM Symp. Princ. of Prog. Lang.* (1989) 179–190.

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

[T] W. Thomas: Finite strategies in regular infinite games, *LNCS 880*, Springer-Verlag (1994) 149–158.

[WN] G. Winskel and M. Nielsen: Models for concurrency, In: S. Abramsky and D. Gabby (Eds.), *Handbook of Logic in computer Science, Vol. 3*, Oxford University Press, Oxford, U.K. (1994).

[WW] K. C. Wong, W. M. Wonham: Modular Control and Coordination of Discrete-Event Systems, *Discrete Event Dynamic Systems*, 6(3):241–273, July 1996