Open Maps, Alternating Simulations and Control Synthesis

Paulo Tabuada

Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556
ptabuada@nd.edu

Abstract. Control synthesis is slowly transcending its traditional application domain within engineering to find interesting and useful applications in computer science. Synthesis of interfaces, distributed network monitors or reactive programs are some examples that benefit from this design paradigm. In this paper we shed new light on the interplay between the fundamental notion of bisimulation and the control synthesis problem. We first revisit the notion of alternating simulation introduced by Alur and co-workers as it naturally captures important ingredients of the control synthesis problem. We then show that existence of controllers enforcing specifications through bisimulation, alternating simulation or simulation can be characterized by the existence of certain alternating simulations and bisimulations between the specification and the system to be controlled. These results highlight and unify the role of simulations and bisimulations in the control synthesis setting for a wide range of concurrency models. This is achieved by developing our study within the framework of open maps. We illustrate our results on transition systems and timed transition systems.

1 Introduction

Computer Science and Control Theory. The control synthesis problem is the central theme of control theory. The traditional setup consists of a system, usually modeled by a differential equation with certain inputs that can be freely assigned, and a specification. The objective is to synthesize a controller, which based on the observation of the current system state, changes the system inputs in order to alter its behavior and to enforce the specification. However, many man made systems are not adequately described by differential equations and in the late 80's Ramadage and Wonham initiated the application of control theoretic ideas to the control of systems described by finite state automata [1]. Even though a different model is used, the same control synthesis problem was shown to be relevant in this context. As introduced by Ramadge and Wonham, the control synthesis problem consists in synthesizing a supervisor finite state automaton C whose parallel composition with the finite state automaton P, modeling the system to be controlled, recognizes a specified regular language S.

P. Gardner and N. Yoshida (Eds.): CONCUR 2004, LNCS 3170, pp. 466–480, 2004.

If one interprets P, S and C as software models, the same problem immediately suggests different applications within computer science such as synthesis of interfaces between software modules [2], distributed monitoring of networks [3], synthesis of reactive embedded controllers [4], etc.

Approximately at the same time that Ramadage and Wonham were obtaining the first results on supervisory control, a similar problem was being investigated in the computer science community: Pnueli and Rosner considered synthesis of reactive software [5, 6]. Synthesis of software from (temporal logic) specifications had already been addressed by the computer science community [7, 8] for closed systems. Independently of the (computer or control) perspective, it is the author's belief that control synthesis problems benefit from the different approaches and contributions originating from computer science and control communities.

Motivation. In this paper we revisit the control synthesis problem in a branching time framework with 3 main objectives:

- Unify control synthesis results across several different concurrency models such as transition systems, asynchronous transition systems, probabilistic transition systems, timed transition systems, Petri nets, etc.
- Highlight the fundamental role played by the notions of bisimulation, alternating simulation and simulation in control synthesis problems.
- Reduce decidability and complexity of control synthesis to decidability and complexity of bisimulation, alternating simulation and simulation.

To accomplish the first objective, we develop our results within the general framework of open maps introduced by Joyal and co-workers [9]. Open maps provide a unified language to discuss and prove results for a large class of apparently different concurrency models. We will use transition systems as a source of motivation and examples throughout the paper and we will also apply our results to timed transition systems which underlie timed automata. However, the general framework of open maps allows to export the presented results to other classes of concurrency models as described in [10, 9, 11, 12].

The second objective motivated us to generalize Alur and co-workers [13] notion of alternating simulation to the open maps framework. Such generalization provides the right language to formulate the control synthesis problem by considering the environment as an opponent trying to violate the specification. The proposed notion coincides with Alur and co-workers notion for transition systems and provides notions of alternating simulation for other classes of concurrency models through the co-reflections introduced in [10]. Such notions and corresponding logic characterizations remain largely unexplored as we focus, in this paper, on the control synthesis problem.

The open maps framework was also crucial in highlighting the similarities and differences between the different versions of the control synthesis problem we have considered. We studied three natural requirements to be enforced by control: bisimulation, alternating simulation and simulation. For each different requirement, we show that existence of a controller is characterized by existence of a bisimulation, alternating simulation or simulation between the specification

and the system to be controlled. In addition to unifying existing results and to highlight the role of bisimulation and similar notions, the developed results also allow to reduce decidability and complexity of control synthesis to decidability and complexity of bisimulation and related notions.

Related Work. The control synthesis problem for transition systems in a branching time framework has been shown to be decidable by Madhusudan and Thiagarajan in [14]. The main ingredient was the characterization of controllers in terms of *good subgraphs* and *strong subgraphs* whose existence can be decided. However, it was not clear in [14] how such objects depend on the underlying concurrency model (transition systems) neither how they relate with alternating simulations. Our results show that such graphs correspond in fact to certain simulations and bisimulations between specification and the system to be controlled. Furthermore, by reformulating existence results in terms of such well known notions, the results become applicable to other classes of systems where these notions make sense. The relation between bisimulation and supervisory control problems was also discussed in [15]. However, bisimulation was used as a way to efficiently compute controllers in a linear time framework, rather than as an essential ingredient for branching time. A different approach was discussed in [16] using co-algebraic methods. Even though bisimulation was used in a fundamental way, through co-inductive definitions and proofs, the approach is rather different from the one considered in this paper. In [16], the adversarial effect of disturbances is captured by a new composition operator rather than by the use of alternating simulations. It is therefore not possible to understand how the requirements for the existence of controllers can be weakened by weakening the required relation between specification and controlled system. Supervisory controllers in branching time were also considered in [17], however failure semantics was used instead of bisimulation to specify the desired behavior. Other lines of research in branching time scenarios considered supervisory control problems for CTL or CTL* specifications [18, 19, 20].

2 The Model

The control synthesis problem can naturally be viewed as a game between the *controller* and the *environement*. To provide motivation for the abstract setup used throughout the paper we will consider such games on a certain class of transition systems, which we will call game structures.

Definition 1. *A game structure is a tuple* $(Q, Q_0, A, \longrightarrow)$ *where:*

1. Q *is a finite set of states;*
2. $Q_0 \subseteq Q$ *is a set of initial states;*
3. A *is a finite set of actions partitioned in two components* A_c *and* A_e *satisfying* $A_c \cup A_e = A$ *and* $A_c \cap A_e = \varnothing$*. Intuitively, the set* A_c *represents the set of controller actions while* A_e *represents the set of environment actions;*
4. $\longrightarrow \subseteq Q \times A \times Q$ *is a transition relation.*

A game structure is said to be deterministic if $(q_1, a, q_2) \in \longrightarrow$ *and* $(q_1, a, q_3) \in \longrightarrow$ *implies* $q_2 = q_3$.

We will frequently resort to the more intuitive notation $q_1 \xrightarrow{a} q_2$ to represent $(q_1, a, q_2) \in \longrightarrow$. We will also restrict our attention to deterministic games where the actions of each player uniquely determine the next state. This is a natural assumption when the nondeterminism in the controller (environment) actions is due to environmental (controller) effects. However, the specification and the controller are allowed to be nondeterministic.

Note that the adopted game model does not require explicit alternation between controller and environment moves, neither does preclude it. However, controller and environment do not play simultaneously. This is simply a technical artifact, since we can consider their actions simultaneous if no information about the opponent move can be used at the time of play. Other game formulations consider game structures where simultaneous play is built in the transition relation as is the case in [13]. These game models, from now on called simultaneous, have a similar structure to the introduced game structures, except that $A = A_c + A_e$ is replaced by $A = A_c \times A_e$. Simultaneous game models $X' = (Q', Q'_0, A'_c \times A'_e, \longrightarrow)$ can be embedded in our framework resulting in games $NS(X') = X = (Q, Q_0, A, \longrightarrow)$ defined by:

1. $Q = Q' \cup Q' \times A'_c$;
2. $Q_0 = Q'_0$;
3. $A = A_c + A_e$, $A_c = A'_c$ and $A_e = A'_e$;
4. $q_1 \xrightarrow{a_c} q_2$ in X with $a_c \in A_c$ iff $q_2 = (q_1, a_c)$ and there is a state $q_3 \in Q$ and an action $a_e \in A_e$ such that $q_1 \xrightarrow{a_c, a_e} q_3$ in X';
5. $q_2 \xrightarrow{a_e} q_3$ in X with $a_e \in A_e$ iff $q_2 = (q_1, a_c)$, $a_c \in A_c$ and $q_1 \xrightarrow{a_c, a_e} q_3$ in X';

We shall not elaborate on the properties of such embedding as it will only be used to relate the notions of alternating simulation and bisimulation introduced in [13] with the ones proposed in this paper. Before introducing such notions, we introduce morphisms between games so as to define the category where we shall develop our study of the control synthesis problem.

Definition 2. *A morphism* $f : X \to Y$ *between two game structures* $X = (Q_X, Q_{0X}, A_X, \longrightarrow)$ *and* $Y = (Q_Y, Q_{0Y}, A_Y, \longrightarrow)$ *is given by a pair of maps* $f = (f_Q, f_A)$ *with* $f_Q : Q_X \to Q_Y$ *a totally defined map and* $f_A : A_X \to A_Y$ *a partially defined map satisfying:*

1. $f_Q(Q_{0X}) \subseteq Q_{0Y}$;
2. $f_A(A_{cX}) \subseteq A_{cY}$ *and* $f_A(A_{eX}) \subseteq A_{eY}$;
3. $q_1 \xrightarrow{a} q_2$ *in* X *implies* $f_Q(q_1) \xrightarrow{f_A(a)} f_Q(q_2)$ *in* Y *if* $f_A(a)$ *is defined and* $f_Q(q_1) = f_Q(q_2)$ *if* $f_A(a)$ *is not defined.*

It is not difficult to see that game structures with the above defined morphisms constitute a category. We shall denote such category by **G**. Furthermore,

since our game models are in particular transition systems, the category **G** is, in many respects, similar to the category of transition systems introduced in [10] thus sharing many of its properties.

3 Bisimulation and Open Maps

In this section we quickly review the open maps framework introduced by Joyal and co-workers [9]. We consider a category **M** of machines with morphisms $X \xrightarrow{f} Y$ describing how machine Y simulates machine X. In this framework, the notion of bisimulation is introduced by resorting to the notion of computation path. We thus consider a subcategory **P** of **M** of path objects whose morphisms describe how paths objects can be extended.

To illustrate this approach we take **G** as the category of machines and for **P** we consider the full subcategory of **G** consisting of objects of the form:

$$q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} q_3 \xrightarrow{a_3} \dots \xrightarrow{a_{n-1}} q_n \tag{1}$$

with q_1 as initial state and $q_i \neq q_j$ for $i \neq j$. We also define the control length of an object M of **P**, denoted by $l_c(M)$, as the number of (not necessarily distinct) controller actions in (1). Similarly, the environment length of M, denoted by $l_e(M)$, is given by the number of environment actions in (1). Given two path objects M and N, a morphism $M \xrightarrow{o} N$ sends the initial state q_1 of M into the initial state q_1' of N, the immediate successor of q_1 into the immediate successor of q_1' and so on. We thus see that o only exists when $l_c(N)+l_e(N) \geq l_c(M)+l_e(M)$ in which case N can be seen as an extension of M. A game path in a game X is now defined as a morphism from a path object M into X, that is $M \xrightarrow{m} X$. Intuitively, morphism $M \xrightarrow{m} X$ describes a possible evolution of the game modeled by X. A morphism $X \xrightarrow{f} Y$ between games can now be seen as describing how Y simulates the game evolution or path $M \xrightarrow{m} X$ by the game evolution path $M \xrightarrow{f \circ m} Y$.

Bisimulation is described through a special path lifting property:

Definition 3. *A morphism $X \xrightarrow{f} Y$ is said to be* **P***-open if given the left commutative diagram in (2), where M and N are path objects, there exists a diagonal morphism $N \xrightarrow{r} X$ making the right diagram in (2) commutative, that is, $m = r \circ o$ and $n = f \circ r$.*

$$\begin{array}{ccc} M & \xrightarrow{m} & X \\ {\scriptstyle o}\downarrow & & \downarrow{\scriptstyle f} \\ N & \xrightarrow{n} & Y \end{array} \qquad \begin{array}{ccc} M & \xrightarrow{m} & X \\ {\scriptstyle o}\downarrow & \nearrow{\scriptstyle r} & \downarrow{\scriptstyle f} \\ N & \xrightarrow{n} & Y \end{array} \tag{2}$$

In the category $\mathbf{G}$ with the above defined path category, the notion of $\mathbf{P}$-open morphism admits the following characterization:

Proposition 1 (Adapted from [9]). *A morphism $X \xrightarrow{f} Y$ is $\mathbf{P}$-open iff for all reachable states q_1 of X:*

if $f_Q(q_1) \xrightarrow{a'} q_2'$ in Y, then $q_1 \xrightarrow{a} q_2$ in X, $f_A(a') = a$ and $f_Q(q_2') = q_2$.

We now consider the fiber subcategories $\mathbf{G}_A$ and $\mathbf{P}_A$ defined by the objects of $\mathbf{G}$ and $\mathbf{P}$ having the same action set A and morphisms f satisfying $f_A = 1_A$. In these subcategories we recover Park [21] and Milner's [22] notion of strong bisimulation through a span of $\mathbf{P}_A$-open maps:

Theorem 1 ([9]). *Let X and Y be objects in $\mathbf{G}_A$. X is bisimilar to Y iff there exists a span $X \xleftarrow{f} Z \xrightarrow{g} Y$ with f a $\mathbf{P}$-open morphism and g a $\mathbf{P}$-open morphism.*

In this setting, a deterministic game model X in $\mathbf{G}_A$ can be characterized by the existence of at most one morphism from a path object in $\mathbf{P}_A$ to X.

4 Alternating Simulation and Open Maps

To introduce alternating simulations we follow a similar route as the one outlined in the previous section by considering two path categories, one for each player:

Definition 4. *The controller (environment) path category $\mathbf{P_c}$ ($\mathbf{P_e}$)consists of the objects of $\mathbf{P}$ and morphisms $M \xrightarrow{o} N$ satisfying $l_c(N) \geq l_c(M)$ and $l_e(N) = l_e(M)$ ($l_c(N) = l_c(M)$ and $l_e(N) \geq l_e(M)$).*

Note that when $l_c(N) \geq l_c(M)$ and $l_e(N) = l_e(M)$, path N extends path M only by controller moves and when $l_e(N) \geq l_e(M)$ and $l_c(N) = l_c(M)$, path N extends path M only by environment moves. Similarly to our discussion in Section 3 we have the following characterization of $\mathbf{P_e}$-open and $\mathbf{P_c}$-open morphisms which is a straightforward generalization of Proposition 1:

Proposition 2. *Let $X \xrightarrow{f} Y$ be a morphism in $\mathbf{G}$. Then, f is $\mathbf{P_c}$-open ($\mathbf{P_e}$-open) iff for any reachable state q_1 in X, $f_Q(q_1) \xrightarrow{a'} q_2'$ in Y implies $q_1 \xrightarrow{a} q_2$ in X, $f_A(a) = a'$ and $f_Q(q_2) = q_2'$ with $a \in A_{cX}$ ($a \in A_{eX}$).*

The above result immediately suggests the following definition of controller and environment simulations:

Definition 5. *Let X and Y be objects in $\mathbf{G}$. Game X c-simulates (e-simulates) game Y if there exists a span $X \xleftarrow{g} Z \xrightarrow{h} Y$ with g a $\mathbf{P_e}$-open ($\mathbf{P_c}$-open) morphism and h a $\mathbf{P_c}$-open ($\mathbf{P_e}$-open) morphism.*

The previous definition captures Alur and co-workers notion of alternating simulation [13] when two player simultaneous games are considered. For later use we recall such notion in this context:

Definition 6 (Adapted from [13]). *Let* $X = (Q_X, Q_{0X}, A_X, \longrightarrow)$ *and* $Y = (Q_Y, Q_{0Y}, A_Y, \longrightarrow)$ *be simultaneous games. A relation* $H \subseteq Q_X \times Q_Y$ *is a c-simulation from* X *to* Y *if for all states* $(q_{1x}, q_{1y}) \in H$ *we have:*

for every controller action $a_{cX} \in A_{cX}$ *available at* q_{1x} *there exists a controller action* $a_{cY} \in A_{cY}$ *available at* q_{1y} *such that for every environment action* $a_{eY} \in A_{eY}$ *available at* q_{1y} *there is an environment action* $a_{eX} \in A_{eX}$ *available at* q_{1x} *satisfying* $q_{1x} \xrightarrow{a_{cX}, a_{eX}} q_{2x}$ *in* X, $q_{1y} \xrightarrow{a_{cY}, a_{eY}} q_{2y}$ *in* Y *and* $(q_{2x}, q_{2y}) \in H$.

Environment simulations or *e*-simulations are obtained from controller simulations or *c*-simulations by reversing the role of the controller and environment. The precise equivalence between Definition 5 and Definition 6 is characterized in the following result:

Theorem 2. *Let* X *and* Y *be two simultaneous game models and* $NS(X)$ *and* $NS(Y)$ *the corresponding objects in* $\mathbf{G}$. *Then,* $NS(X)$ *c-simulates (e-simulates)* $NS(Y)$, *in the sense of Definition 5, iff* X *c-simulates (e-simulates)* Y *in the sense of Definition 6.*

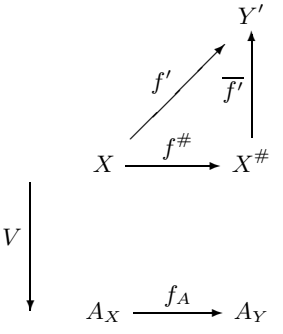
It is now clear that the notion of alternating simulation can be naturally described within the open maps framework. An interesting question not addressed in this paper is the study of alternating simulation notions induced by Definition 5 in other classes of concurrency models as well as the corresponding logic characterizations. Alternating simulation will play a fundamental role in the control synthesis problem described in the next section.

5 Control Synthesis

Co-fibrations and Parallel Composition. The control synthesis problem requires, in addition to bisimulations and alternating simulations, a notion of parallel composition. As detailed in [10], the usual notions of parallel composition can not be described by a single categorical construct. Instead, they are obtained by a sequence of product, restriction and relabeling operations. In this paper, we consider only the usual composition by synchronization on common events, although through a simpler alternative description resorting to co-fibrations. To motivate the notion of co-fibration, we revisit our game category $\mathbf{G}$.

Every game model X contains a set of actions and every morphism f contains a map f_A transforming actions into actions. This suggests a "projection" functor V from $\mathbf{G}$ to the category of sets and partial maps between sets. Such functor has the obvious definition $V(X) = V(Q, Q_0, A, \longrightarrow) = A$ and $V(f) = V(f_Q, f_A) = f_A$. For a given set A, we denote by $\mathbf{G}_A$ the fiber category consisting of the objects X of $\mathbf{G}$ satisfying $V(X) = A$ and morphisms f satisfying $V(f) = 1_A$. Consider now a morphism $X \xrightarrow{f} Y$ in $\mathbf{G}$ and let $V(X) = A_X$ and $V(Y) = A_Y$. We can construct an object $X^{\#}$, $V(X^{\#}) = V(Y)$, from X and f_A by replacing every $q_1 \xrightarrow{a} q_2$ in X with $q_1 \xrightarrow{f_A(a)} q_2$. This new object allows to factor f as $X \xrightarrow{f^{\#}} X^{\#} \xrightarrow{\overline{f}} Y$, where $f^{\#} = (1_Q, f_A)$ and $\overline{f} = (f_Q, 1_{A_Y})$. Furthermore,

for any other morphism $X \xrightarrow{g} Z$ with $V(g) = V(f)$ there exists a unique morphism $X^{\#} \xrightarrow{\overline{g}} Z$ such that $\overline{g} \circ f^{\#} = f'$ as is pictorially represented in (3).



(3)

Such unique factorization properties are abstracted into the notion of co-fibration that we now introduce following [23].

Definition 7. *Let $F : \mathbf{D} \to \mathbf{E}$ be a functor and $\alpha : J \to I$ a morphism of $\mathbf{E}$. A morphism $f^{\#} : X \to Y$ of $\mathbf{D}$ is pre-cocartesian over α if:*

1. *$F(f^{\#}) = \alpha$;*
2. *if $g : X \to Z$ is a morphism of $\mathbf{E}$ such that $F(g) = \alpha$, there exists a unique morphism in the fiber $\mathbf{D}_I$ $h : Y \to Z$ such that $g = h \circ f^{\#}$*

Pre-cocartesian morphisms are used to define co-fibrations as follows:

Definition 8. *A functor $F : \mathbf{D} \to \mathbf{E}$ is said to be a co-fibration if:*

1. *for every morphism $\alpha : J \to I$ of $\mathbf{E}$ and every object X in the fiber over J, there exists in $\mathbf{D}$ a pre-cocartesian morphism $f^{\#} : X \to Y$ over α;*
2. *the composition of two pre-cocartesian morphisms is again pre-cocartesian.*

At this point the reader may find useful to return to diagram (3) and the discussion preceding it. Once again looking at $\mathbf{G}$, we see that every pre-cocartesian morphism $f^{\#}$ is $\mathbf{P}$-open, since every $1_Q(q_1) \xrightarrow{a'} q_2$ in $X^{\#}$ was obtained from a transition $q_1 \xrightarrow{a} q_2$ in X with $a' = f_A(a)$ which implies $\mathbf{P}$-openness of f by Proposition 2. Based on this observation, we will make the following assumption which will hold throughout the paper:

A.I *The game category $\mathbf{G}$ is equipped with a functor $V : \mathbf{G} \to \mathbf{L}$ which is a co-fibration. Furthermore, the co-fibration respects open maps in the sense that every pre-cocartesian morphism in $\mathbf{G}$ is $\mathbf{P}$-open.*

We now turn to another important ingredient, parallel composition. We shall abstract the usual notion of parallel composition by synchronization on common events to our framework through the following assumption:

A.II *The parallel composition operator restricts to a fiber product, that is, for objects X and Y in the fiber $\mathbf{G}_A$, $X \parallel Y = X \times_A Y$. Furthermore, $X \parallel Y$ comes equipped with morphisms $X \xleftarrow{x} X \parallel Y \xrightarrow{y} Y$.*

We now recall the definition of composition by synchronization on common events with the purpose of illustrating the above assumption.

Definition 9. *Let X and Y be objects in $\mathbf{G}$. The parallel composition of X and Y by synchronization on common events is the object $X \parallel Y = (Q_X \times Q_Y, Q_{0X} \times Q_{0Y}, (A_{cX} \cup A_{cY}) + (A_{eX} \cup A_{eY}), \longrightarrow)$ defined by $(q_{1x}, q_{1y}) \xrightarrow{a} (q_{2x}, q_{2y})$ in $X \parallel Y$ if:*

1. *$q_{1x} \xrightarrow{a} q_{2x}$ in X, $q_{1y} \xrightarrow{a} q_{2y}$ in Y and $a \in A_{cX} \cap A_{cY}$ or $a \in A_{eX} \cap A_{eY}$.*
2. *$q_{1x} \xrightarrow{a} q_{2x}$ in X, $q_{1y} = q_{2y}$ and $a \in A_{cX}$, $a \notin A_{cY}$ or $a \in A_{eX}$, $a \notin A_{eY}$.*
3. *$q_{1y} \xrightarrow{a} q_{2y}$ in Y, $q_{1x} = q_{2x}$ and $a \in A_{cY}$, $a \notin A_{cX}$ or $a \in A_{eY}$, $a \notin A_{eX}$.*

This notion of parallel composition comes equipped with projection morphisms $X \xleftarrow{x} X \parallel Y \xrightarrow{y} Y$ defined by $x_Q(q_x, q_y) = q_x$, $x_A(a) = a$ if $a \in A_X$ and $x_A(a)$ undefined in $a \notin A_X$. Morphism y is similarly defined. Furthermore, when $A_X = A = A_Y$, $X \parallel Y$ coincides with the categorical product $X \times_A Y$ on the fiber category $\mathbf{G}_A$. Recall that the categorical product $X \times_A Y$ is the object of $\mathbf{G}_A$ equipped with morphisms $X \xleftarrow{\pi_X} X \times_A Y \xrightarrow{\pi_Y} Y$ and satisfying the following property: for every $X \xleftarrow{f} Z \xrightarrow{g} Y$ in $\mathbf{G}_A$, there is one and only one morphism $h : Z \to X \times_A Y$ such that $\pi_X \circ h = f$ and $\pi_Y \circ h = g$.

Assumptions **A.I** and **A.II** provide a general setup allowing to study the control synthesis problem across several different categories of game or computation models. In addition to the working example of transition systems, in Section 6 we will apply the developed results to timed transition systems.

Existence and Synthesis of Controllers (Bisimulation). We now consider the control synthesis problem for bisimulation equivalence, that is, given a plant P and a specification S we seek to determine if a controller C rendering $C \parallel P$ bisimilar to S exists. More specifically we have:

Definition 10. *Let P, S and C be objects in $\mathbf{G}$. Object C is a bisimulation controller for plant P, enforcing specification S, if the following holds:*

1. *Morphism $p : C \parallel P \to P$ is $\mathbf{P_e}$-open;*
2. *There exists a span $S \xleftarrow{s} Z \xrightarrow{cp} C \parallel P$ with s a $\mathbf{P}$-open morphism and cp a $\mathbf{P}$-open morphism, that is, S bisimulates $C \parallel P$.*

The first condition requires controller C not to restrict environment moves as these cannot be influenced by the controller. The second condition asks for bisimulation equivalence between the controlled game $C \parallel P$ and the specification, a natural requirement in a branching time framework. Necessary and sufficient conditions for the existence of such controller can be formulated in terms of certain $\mathbf{P}$-open and $\mathbf{P_e}$-open morphisms:

Theorem 3. *Let P be a deterministic object in $\mathbf{G}$ and S an arbitrary object in $\mathbf{G}$. There exists a bisimulation controller C for plant P enforcing specification S iff there is a span $S \xleftarrow{\gamma} Z \xrightarrow{\delta} P$ with γ a $\mathbf{P}$-open morphism and δ a $\mathbf{P_e}$-open morphism. Furthermore, when a bisimulation controller C exists, we can take $C = Z^{\#}$ which has the same set of actions as P.*

The previous result shows that existence of a bisimulation controller is equivalent to the requirement that P must simulate a bisimilar version Z of S while ensuring that every environment move in P is also possible in Z. This is a natural requirement as the controller C will restrict P to the image under δ of Z.

Existence and Synthesis of Controllers (e-Simulation). We now restrict attention to safety environment properties and liveness control specifications. These requirements are modeled by requiring the specification to e-simulate the controlled game. A controller enforcing the specification through an e-simulation restricts the effect of disturbances to accommodate safety properties while being as live as required by the specification. Formally, we define e-simulation controllers as follows:

Definition 11. *Let P, S and C be objects in $\mathbf{G}$. Object C is a e-simulation controller for plant P, enforcing specification S, if the following holds:*

1. *Morphism $p : C \parallel P \to P$ is $\mathbf{P_e}$-open.*
2. *There exists a span $S \xleftarrow{s} Z \xrightarrow{cp} C \parallel P$ with s a $\mathbf{P_c}$-open morphism and cp a $\mathbf{P_e}$-open morphism, that is, S e-simulates $C \parallel P$.*

This kind of specification appears to be new since the Ramadge-Wonham framework only considers language equality, which corresponds to bisimulation in the branching time setting, or language inclusion which corresponds to simulation in the branching time setting. Simulation requirements are in fact weaker than e-simulation requirements and are discussed below.

Theorem 4. *Let P be a deterministic object in $\mathbf{G}$ and S an arbitrary object in $\mathbf{G}$. There exists an e-simulation controller C for plant P enforcing specification S iff there is a span: $S \xleftarrow{\gamma} Z \xrightarrow{\delta} P$ with γ a $\mathbf{P_c}$-open morphism and δ a $\mathbf{P_e}$-open morphism. Furthermore, when an e-simulation controller C exists, we can take $C = Z^{\#}$ which has the same set of actions as P.*

It is interesting to note that, with respect to Theorem 3, only the assumptions of the left leg of span $S \xleftarrow{\gamma} Z \xrightarrow{\delta} P$ have been weakened. The same observation holds with respect to the results of the next section where a weaker version of the control synthesis problem is considered.

Existence and Synthesis of Controllers (Simulation). We now further weaken the control synthesis problem by only requiring the specification to simulate the controlled game. To illustrate the difference with respect an e-simulation requirement, we consider the specification, plant, controller and controlled system displayed in Figure 1. Controller C enforces the specification S by preventing the occurrence of action c_2 at the initial state. By looking at the controlled game $C \parallel P$ we see that there is an obvious inclusion morphism from $C \parallel P$ to S showing that S simulates the controlled game. However, C fails to be an e-simulation controller since it violates the liveness requirement to perform action c_2 at the initial state. Simulation requirements are therefore weaker than e-simulation requirements and constitute a natural specification when e-simulation controllers

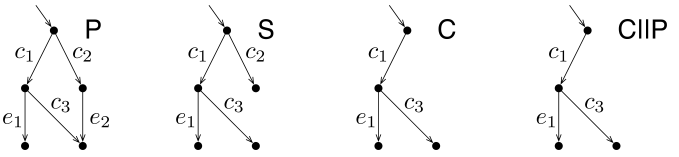


Fig. 1. Pictorial representation of the plant P, specification S, controller C and corresponding controlled system $C \parallel P$

cannot be obtained. Nevertheless, requiring the specification only to simulate the controlled game may result in a trivial control synthesis problem since a controller preventing the occurrence of any controller action may constitute a solution. To rule out such trivial controllers we follow the Ramadge-Wonham approach by imposing a mild liveness restriction on the controller. We will require the possible controller to enforce the specification without creating blocking states on the controlled game. Such nonblocking assumption is formalized in our context through the notion of maximal paths.

Definition 12. *Let X be an object in $\mathbf{G}$ and $o : O \to X$ a path in X. Path o is said to be maximal for X if given any other path $o' : O' \to X$ such that $o' \circ m = o$, there is one and only one morphism $m' : O' \to O$ satisfying $o \circ m' = o'$. A morphism $X \xrightarrow{f} Y$ is said to preserve maximal paths if for every maximal path $O \xrightarrow{o} X$, $O \xrightarrow{f \circ o} Y$ is also a maximal path.*

Given the above definitions we consider a controller C nonblocking, when the morphism $p : C \parallel P \to P$ preserves maximal paths. This definition captures the supervisory control notion of nonblocking controller as shown in the next result.

Proposition 3. *Let C and P be objects in $\mathbf{G}$. Morphism $p : C \parallel P \to P$ preserves maximal paths iff for any reachable state q_1 in $C \parallel P$, $p_Q(q_1) \xrightarrow{a'} q_2'$ in P implies $q_1 \xrightarrow{a} q_2$ in $C \parallel P$.*

We are now ready to formulate the simulation version of the control synthesis problem:

Definition 13. *Let P, S and C be objects in $\mathbf{G}$. Object C is a simulation controller for plant P, enforcing specification S, if the following holds:*

1. *Morphism $p : C \parallel P \to P$ is $\mathbf{P_e}$-open and preserves maximal paths.*
2. *There exists a span $S \xleftarrow{s} Z \xrightarrow{cp} C \parallel P$ with cp a $\mathbf{P}$-open morphism, that is, S simulates $C \parallel P$.*

Theorem 5. *Let P be a deterministic object in $\mathbf{G}$ and S an arbitrary object in $\mathbf{G}$. There exists a simulation controller C for plant P enforcing specification S iff there is a span $S \xleftarrow{\gamma} Z \xrightarrow{\delta} P$ with δ a $\mathbf{P_e}$-open morphism preserving maximal paths. Furthermore, when a simulation controller C exists, we can take $C = Z^{\#}$ which has the same set of actions as P.*

Once again, only the assumptions on the left leg of span $S \xleftarrow{\gamma} Z \xrightarrow{\delta} P$ have been reduced to the requirement that γ is simply a morphism. On the other hand the new nonblocking requirement is now reflected on the maximal path preservation assumption. The simplicity of Theorems 3, 4 and 5 and their applicability to a large class of concurrency models illustrates the merit of the open maps approach. To further emphasize applicability, we describe in the next section how the developed results can be used with timed transition systems.

6 Timed Transition Systems

In this section we briefly outline how the presented results can also be used for timed transition systems control synthesis problems. Timed transition systems are transition systems enriched with timing information. They correspond to timed automata [24] without acceptance conditions or accepting states. By partitioning the action set into controller and environment actions we can also talk about timed games on timed game structures:

Definition 14. *A timed game structure is a tuple* $(Q, Q_0, A, \mathbb{T}, \longrightarrow)$ *where:*

1. Q *is a finite set of states;*
2. $Q_0 \subseteq Q$ *is a finite set of initial states;*
3. A *is a finite set of actions partitioned in two components* A_c *and* A_e *satisfying* $A_c \cup A_e = A$ *and* $A_c \cap A_e = \varnothing$*. Intuitively, the set* A_c *represents the set of controller actions while* A_e *represents the set of environment actions;*
4. $\mathbb{T}$ *is a finite set of clocks;*
5. $\longrightarrow \subseteq Q \times A \times \Omega \times 2^{\mathbb{T}} \times Q$ *is a transition relation where* Ω *is a clock constraint generated by the grammar* $\Omega ::= c \sim t_1 | t_1 + c \sim t_2 | \Omega \wedge \Omega$ *with* $\sim \in \{\leq, <, \geq, >\}$, $c \in \mathbb{R}$ *and* t_1, t_2 *clock variables.*

We will resort to the more intuitive notation $q_1 \xrightarrow[\omega,\rho]{a} q_2$ to represent $(q_1, a, \omega, \rho, q_2) \in \longrightarrow$. Intuitively, the set of clocks records the passage of time which is then used to determine if and when a transition can be taken. Timing conditions on transitions are captured by clock constraints $\omega \in \Omega$. If we are using l clocks, then a clock constraint can be identified with a subset of $(\mathbb{R}_0^+)^l$, denoted by $[\omega]_{\mathbb{T}}$, representing the clock values satisfying the constraint. Given a function $g : \mathbb{T}_1 \to \mathbb{T}_2$ between two sets of clocks and a constraint ω on the clocks in $\mathbb{T}_2$, we denote by $[\omega \circ g]_{\mathbb{T}_1}$ the constraint induced by ω on the clocks in $\mathbb{T}_1$. By associating the discrete state $q_1 \in Q$ with the current value $t_1 \in \mathbb{R}^l$ of the clocks in $\mathbb{T}$, we obtain a configuration (q_1, t_1). Sequences of configurations describe how the states of a given timed transition system evolve over time. Such sequences:

$$(q_0, t_0) \xrightarrow[\tau_1]{a_1} (q_1, t_1) \xrightarrow[\tau_2]{a_2} (q_2, t_2) \xrightarrow[\tau_3]{a_3} \cdots \xrightarrow[\tau_n]{a_n} (q_n, t_n)$$

can take place when for each i, there exists a transition $q_{i-1} \xrightarrow[\omega_i,\rho_i]{a_i} q_i$ in the timed game structure, the transition time satisfies the clock constraint[1] $t_{i-1} + (\tau_i -$

[1] We denote by $\mathbf{1}$ the element of $(\mathbb{R}_0^+)^l$ in which every component is equal to 1.

$\tau_{i-1})\mathbf{1} \in [\omega_i]_{\mathbb{T}}$ and the jth clock time $(t_i)_j$ is updated by $(t_i)_j = (t_{i-1})_j + \tau_i - \tau_{i-1}$ if $j \notin \rho_i$ or $(t_i)_j = 0$ if $j \in \rho_i$. To completely describe our category of timed game structures, we define morphisms following [25].

Definition 15. *A morphism $f : X \to Y$ between two timed game structures $X = (Q_X, Q_{0X}, A_X, \mathbb{T}_X, \longrightarrow)$ and $Y = (Q_Y, Q_{0Y}, A_Y, \mathbb{T}_Y, \longrightarrow)$ is given by a pair of maps $f = (f_Q, f_{\mathbb{T}})$ with $f_Q : Q_X \to Q_Y$ and $f_{\mathbb{T}} : \mathbb{T}_Y \to \mathbb{T}_X$ satisfying:*

1. *$f_Q(Q_{0X}) \subseteq Q_{0Y}$;*
2. *$q_{1x} \xrightarrow[\omega_x, \rho_x]{a} q_{2x}$ in X implies $f_Q(q_{1x}) \xrightarrow[\omega_y, \rho_y]{a} f_Q(q_{2x})$ in Y with $\rho_y = f_{\mathbb{T}}^{-1}(\rho_x) = \{c \in \mathbb{T}_Y \mid f_{\mathbb{T}}(c) \in \rho_x\}$ and $[\omega_x \circ f_{\mathbb{T}}]_{\mathbb{T}_Y} \subseteq [\omega_y]_{\mathbb{T}_Y}$.*

Note that we are only considering timed game structures with the same labeling set A and morphisms relating actions through the identity map on actions. This means that we are in fact working on the fiber subcategory over A. This also means that assumption **A.I** is automatically satisfied since pre-cocartesian morphisms are simply identity morphisms given the fact that $V(f) = 1_A$ for every morphism f. Assumption **A.II** is also satisfied as we will consider the categorical product between timed game structures as our notion of parallel composition [25]. The path subcategory $\mathbf{P}$ required to define bisimulation is now introduced through the use of timed words.

Definition 16. *A timed word α over an alphabet A is an element of $(A \times \mathbb{R}_{\geq 0})^*$, that is, a finite sequence:*

$$\alpha = (a_1, \tau_1)(a_2, \tau_2)(a_3, \tau_3) \dots (a_n, \tau_n)$$

satisfying $a_i \in A$, $\tau_i \in \mathbb{R}_{\geq 0}$ and $\tau_{i+1} > \tau_i$ for $1 \leq i \leq n$.

As detailed in [25], timed words can be embedded into **TG** as the following objects:

$$0 \xrightarrow[\omega_1, \rho_1]{a_1} 1 \xrightarrow[\omega_2, \rho_2]{a_2} 2 \xrightarrow[\omega_3, \rho_3]{a_3} \dots \xrightarrow[\omega_n, \rho_n]{a_n} n \tag{4}$$

where ω_i and ρ_i are appropriately chosen to create a full and faithful functor from the category of timed words and morphisms describing timed word extensions into **TG**. We refer the interested readers to [25] for the details of such embedding and consider $\mathbf{P}$ as the category of objects of the form (4) with morphisms describing how such objects can be extended. We also define the controller (environment) length of an object M in $\mathbf{P}$, denoted by $l_c(M)$ ($l_e(M)$) as the number of not necessarily distinct controller (environment) actions appearing in M. Controller and environment lengths allow to define $\mathbf{P_c}$ and $\mathbf{P_e}$ as the subcategories of $\mathbf{P}$, where morphisms $o : M \to N$ satisfy $l_e(M) = l_e(N)$ and $l_c(N) \geq l_c(N)$ when M and N are objects of $\mathbf{P_c}$ and $l_c(M) = l_c(N)$ and $l_e(N) \geq l_e(N)$ when M and N are objects of $\mathbf{P_e}$. Similarly to the un-timed case we only consider deterministic timed game structures.

With respect to definitions 10, 11 and 13, we now have the following characterization for the different control synthesis problems on timed game structures.

Theorem 6. *Let P and S be objects in* **TG**.

1. *There exists a bisimulation controller C for plant P enforcing specification S iff there is a span $S \xleftarrow{\gamma} Z \xrightarrow{\delta} P$ with γ a $\mathbf{P}$-open morphism and δ a $\mathbf{P_e}$-open morphism.*
2. *There exists an e-simulation controller C for plant P enforcing specification S iff there is a span $S \xleftarrow{\gamma} Z \xrightarrow{\delta} P$ with γ a $\mathbf{P_c}$-open morphism and δ a $\mathbf{P_e}$-open morphism.*
3. *There exists a simulation controller C for plant P enforcing specification S iff there is a span $S \xleftarrow{\gamma} Z \xrightarrow{\delta} P$ with δ a $\mathbf{P_e}$-open morphism preserving maximal paths.*

Furthermore, when a bisimulation (e-simulation or simulation) controller C exists, we can take $C = Z$.

7 Future and Ongoing Work

We have only considered the control synthesis problem for deterministic systems. Determinism is a natural assumption when nondeterminism in the effect of controller (environment) actions is captured by the environment (controller) actions. However, nondeterminism may also exist due to other causes such as abstraction. It is therefore natural to extend the presented results to the nondeterministic case, especially since some of the proofs use determinism in a essential way. Other unexplored avenues include the instantiation of the developed results for other classes of systems such as Petri nets for which purely linear algebraic techniques [26] exist for controller synthesis. A different direction being currently investigated is the extension of the presented work to accommodate different notions of parallel composition.

References

1. Ramadge, P., Wonham, W.M.: The control of discrete event systems. Proceedings of IEEE **77** (1989) 81–98
2. de Alfaro, L., Henzinger, T.A.: Interface theories for component-based design. In: Proceedings of the First International Workshop on Embedded Software. Volume 2211 of Lecture Notes in Computer Science. (2001) 148–165
3. Benveniste, A., Haar, S., Fabre, E., Jard, C.: Distributed monitoring of concurrent and asynchronous systems. In Amadio, R.M., Lugiez, D., eds.: Proceedings of the 14th International Conference on Concurrency Theory (CONCUR). Volume 2761 of Lecture Notes in Computer Science., Springer-Verlag (2003) 1–26
4. Tabuada, P., Pappas, G.J.: Linear time logic control of linear systems. (2004) Submitted for publication, available at `www.nd.edu/~ptabuada`.
5. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Proceedings of the 16th ACM Symposium on Principles of Programming Languages. (1989) 170–190
6. Pnueli, A., Rosner, R.: Distributed reactive systems are hard to synthesize. In Press, I., ed.: Proceedings of the 31st Annual Symposium on Foundations of Computer Sience, St. Louis, Missouri (1990) 746–757

7. Emerson, E.A., Clarke, E.M.: Using branching time temporal logic to synthesize synchronization skeletons. Science of Computer Programming **2** (1982) 241–266
8. Manna, Z., Wolper, P.: Synthesis of communication processes from temporal logic specifications. ACM Transactions on Programming Languages and Systems **6** (1984) 68–93
9. Joyal, A., Nielsen, M., Winskel, G.: Bisimulation from open maps. Information and Computation **127** (1996) 164–185
10. Winskel, G., Nielsen, M.: Models for concurrency. In Abramsky, Gabbay, Maibaum, eds.: Handbook of Logic and Foundations of Theoretical Computer Science. Volume 4. Oxford University Press, London (1994)
11. Nielsen, M., Cheng, A.: Observing behaviour categorically. In Thiagarajan, P.S., ed.: Foundations of Software Technology and Theoretical Computer Science. Volume 1026 of Lecture Notes in Computer Science., Springer (1995) 263–278
12. Haghverdi, E., Tabuada, P., Pappas, G.: Bisimulation relations for dynamical and control systems. In Blute, R., Selinger, P., eds.: Electronic Notes in Theoretical Computer Science. Volume 69., Elsevier (2003)
13. Alur, R., Henzinger, T., Kupferman, O., Vardi, M.: Alternating refinement relations. In: CONCUR 97: Concurrency Theory, 8th International Conference. Number 1466 in Lecture Notes in Computer Science (1998) 163–178
14. Madhusudan, P., Thiagarajan, P.: Branching time controllers for discrete event systems. Theoretical Computer Science **274** (2002) 117–149
15. Barret, G., Lafortune, S.: Bisimulation, the supervisor control problem and strong model matching for finite state machines. Journal of Discrete Event Systems **8** (1998) 337–429
16. Rutten, J.: Coalgebra, concurrency, and control. In Kluwer, ed.: Proceedings of the 5th Workshop on Discrete Event Systems (WODES 2000). (2000) 31–38
17. Overkamp, A.: Supervisory control using failure semantics and partial specifications. IEEE Transactions on Automatic Control **42** (1997) 498–510
18. Kupferman, O., Madhusudan, P., Thiagarajan, P.S., Vardi, M.Y.: Open systems in reactive environments: Control and synthesis. In: Proceedings of the 11th International Conference on Concurency Theory. Volume 1877 of Lecture Notes in Computer Science., Springer-Verlag (2000) 92–107
19. Antoniotti, M., Mishra, B.: NP-completeness of the supervisor synthesis problem for unrestricted CTL specifications. In Smedinga, R., Spathopoulos, M., Kozk, P., eds.: Proceedings on the International Workshop on Discrete Event Systems, WODES96, Edinburgh, Scotland, UK (1996)
20. Shengbing, J., Kumar, R.: Supervisory control of discrete event systems with CTL* temporal logic specifications. In: Proceedings of the 40th IEEE Conference on Decision and Control. Volume 5. (2001) 4122–4127
21. Park, D.: Concurrency and automata on infinite sequences. Volume 104 of Lecture Notes in Computer Science. (1981) 167–183
22. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
23. Borceux, F.: Handbook of Categorical Algebra. Cambridge University Press (1994)
24. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science **126** (1994) 183–235
25. Nielsen, M., Hune, T.S.: Bisimulation and open maps for timed transition systems. Fundamenta Informaticae **38** (1999) 61–77
26. Moody, J.O., Antsaklis, P.J.: Supervisory Control of Discrete Event Systems Using Petri Nets. Kluwer Academic Publishers (1998)