Games for synthesis of controllers with partial observation

A. Arnold, A. Vincent, I. Walukiewicz
LaBRI, Université Bordeaux I and CNRS (UMR 5800)

Abstract

The synthesis of controllers for discrete event systems, as introduced by Ramadge and Wonham, amounts to computing winning strategies in parity games. We show that in this framework it is possible to extend the specifications of the supervised systems as well as the constraints on the controllers by expressing them in the modal μ-calculus.

In order to express unobservability constraints, we propose an extension of the modal μ-calculus in which one can specify whether an edge of a graph is a loop. This extended μ-calculus still has the interesting properties of the classical one. In particular it is equivalent to automata with loop testing. The problems such as emptiness testing and elimination of alternation are solvable for such automata.

The method proposed in this paper to solve a control problem consists in transforming this problem into a problem of satisfiability of a μ-calculus formula so that the set of models of this formula is exactly the set of controllers that solve the problem. This transformation relies on a simple construction of the quotient of automata with loop testing by a deterministic transition system. This is enough to deal with centralized control problems. The solution of decentralized control problems uses a more involved construction of the quotient of two automata.

This work extends the framework of Ramadge and Wonham in two directions. We consider infinite behaviours and arbitrary regular specifications, while the standard framework deals only with specifications on the set of finite paths of processes. We also allow dynamic changes of the set of observable and controllable events.

1 Introduction

At the end of the eighties, Ramadge and Wonham introduced the theory of control of discrete event systems (see the survey [14] and the books [8] and [4]). In this theory a process (also called a *plant*) is a deterministic non-complete finite state automaton over an alphabet A of events, which defines all possible sequential behaviours of the process. Some of the states of the plant are termed *marked*.

The alphabet A is the disjoint union of two subsets: the set A_{cont} of controllable events and the set A_{unc} of uncontrollable events. A is also the disjoint union of the sets A_{obs} of observable events and A_{uno} of unobservable events.

A controller is a process R which satisfies the two following conditions:

(C) For any state q of R, and for any uncontrollable event a, there is a transition from q labelled by a.

(O) For any state q of R, and for any unobservable event a, if there is a transition from q labelled by a then this transition is a loop over q.

In other words, a controller must react to any uncontrollable event and cannot detect the occurrence of an unobservable event.

If P is a process and R is a controller, the supervised system is the product $P \times R$. Thus, if this system is in the state (p, r) and if for some controllable event a, there is no transition labelled by a from r, the controller forbids the supervised system to perform the event a. On the other hand, if an unobservable event occurs in the supervised system, the state of the controller does not change, as if the event had not occurred.

Of course, a supervised system has less behaviours than its plant alone. In particular a supervised system may not reach states of the plant which are unwanted for some reason. On the other hand, one can a priori define a set of admissible behaviours of the plant, and the control problem is to find a controller R such that all behaviours of the supervised system are admissible. For instance, one can demand that some dangerous states are never reachable, or that one can always go back to the initial state of the plant.

More formally, the basic control problem is the following:

> Given a plant P and a set S of behaviours, does there exist a controller R satisfying (C) and (O) such that the behaviours of the supervised system $P \times R$ are all in S?

and the synthesis problem is to construct such a controller if it does exist.

Some variants of this problem take into account the distinction between terminal and non terminal behaviours (in [14] called marked and non marked) of the plant.

In their works Ramadge and Wonham are mainly interested in finding *maximal* controllers, i.e., controllers such that the behaviours of the supervised system are exactly the admissible behaviours of the plant. A less restrictive problem than finding maximal controllers is finding controllers such that the set of supervised behaviours lies between a set of admissible behaviours and a set of required behaviours, for instance, to discard controllers which forbid everything (all the behaviours of an empty set are admissible!).

Indeed, all these constraints on the behaviour of the supervised system, which amounts to saying that all paths in the system are in some regular languages, and/or that all words of a given language are paths of the system, can be expressed by formulas of the modal μ-calculus: for each such constraint there is a formula Φ such that the supervised system satisfies Φ if and only if its behaviour satisfies the constraint. Therefore, we extend the Ramadge-Wonham's approach by using any formula of the modal μ-calculus to specify the desired property of the supervised system. In this way, we can specify much more requirements on the supervised system (see, for instance, the example of Section 2.5 below).

Hence, the control problem becomes

> Given a plant P and a formula Φ , does there exist a controller R satisfying (C) and (O) such that $P \times R$ satisfies Φ?

In [20], this problem is solved when R has only to satisfy the condition (C) and the corresponding synthesis problem is solved by finding a winning strategy in a parity game.

But it turns out that the condition (C) is also expressible in the modal μ-calculus by the formula $\nu x.(\bigwedge_{b \in A_{unc}} \langle b \rangle x \wedge \bigwedge_{c \in A_{cont}} [c]x)$, taking into account the fact that this formula applies to a *deterministic* process.

Therefore, a natural generalization of the problem addressed in [20] is

> (**P**) Given a plant P and two formulas Φ and Ψ, does there exist a controller R satisfying Ψ such that $P \times R$ satisfies Φ?

An example of such a formula Ψ characterizing the controller is the following. Let a, c, f be three events where only c is controllable. The event f symbolizes a failure of the device which controls c so that after an occurrence of f, the event c becomes uncontrollable. The formula expressing this phenomenon is $\nu x.(\langle a \rangle x \wedge [c]x \wedge \langle f \rangle \nu y.(\langle a \rangle y \wedge \langle c \rangle y \wedge \langle f \rangle y))$. Another example is the case where only one out of two events c_1 and c_2 is controllable at a time. This is expressed by $\nu x.(\langle a \rangle x \wedge ((\langle c_1 \rangle x \wedge [c_2]x) \vee ([c_1]x \wedge \langle c_2 \rangle x))$.

It remains to deal with the condition (O) which, unfortunately, is not expressible in the modal μ-calculus because it is not invariant under bisimulation. That is why we extend the modal μ-calculus into a *modal-loop* μ-calculus. This extension consists in associating with each event a a basic proposition $\circlearrowleft_a$ whose standard interpretation is that a state q of a controller has this property if the event a occurs in state q and leads to the same state q. For instance the condition (O) can be expressed as $\nu x.(\bigwedge_{a\in A_{obs}}[a]x \wedge \bigwedge_{a\in A_{uno}}([a]\mathit{false}\vee \circlearrowleft_a)$. And also, we can express that an observable event becomes unobservable after a failure: $\nu x.(\cdots \wedge [a]x \wedge \langle f\rangle \nu y.(\cdots \wedge ([a]\mathit{false}\vee \circlearrowleft_a) \wedge \langle f\rangle y))$, or that at most one out of two events a and b is observable: $[a]\mathit{false}\vee \circlearrowleft_a \vee [b]\mathit{false}\vee \circlearrowleft_b$.

Therefore we consider problem (**P**) as the general form of a control problem when Φ and Ψ are modal-loop formulas.

It turns out, fortunately, that modal-loop μ-calculus has quite similar properties to the ordinary μ-calculus. For instance, and it is very convenient, modal automata have the same expressive power as the modal μ-calculus, and moreover, translating μ-formulas into automata and vice-versa is quite easy. Here we introduce modal-loop automata which are an extension of standard modal automata with the ability to test for existence of a loop. These loop automata are equivalent in expressive power to the loop μ-calculus in the same way as standard automata are equivalent to the standard μ-calculus [1]. The reason for this is simply that one can consider the property of having a loop as a local property of states. Therefore, although in this introduction we speak about formulas, in the paper we will consider only automata.

The two crucial properties of alternating automata, that are also shared by loop alternating automata are:

Eliminating alternation Every loop automaton is equivalent to a nondeterministic loop automaton(Theorem 23).

Sat The emptiness of a nondeterministic loop automaton can be effectively decided and a process accepted by the automaton can be effectively constructed (Theorem 24).

The first result of the paper is the construction of a modal-loop formula Φ/P that is satisfied by precisely those controllers R for which $P \times R \vDash \Phi$. This way a process R is a solution of the synthesis problem **P** if and only if $R \vDash (\Phi/P) \wedge \Psi$.

By the properties above, $(\Phi/P) \wedge \Psi$ can be effectively transformed into a nondeterministic loop automaton and a controller R can be synthesized.

This transformation may cause an exponential blow-up in size, and the powerset construction given in [2] for dealing with the condition (O) is indeed a special case of this transformation.

Therefore, all control problems of the form (**P**) are indeed satisfiability problems in the modal-loop μ-calculus and the synthesis problems amount to finding models of modal-loop formulas, whose effectiveness is ensured by the above properties. Indeed, finding such models consists in finding winning strategies in parity games, probably the most fascinating problem in the μ-calculus [21]. (Reciprocally, finding a winning strategy is itself a control problem: your moves are controllable and the moves of your opponent are not!)

Ramadge and Wonham have considered also the synthesis of decentralized controllers: a plant can be supervised by several independent controllers (instead of only one). But each controller has its own set of controllable and observable events. Hence the decentralized control problem is to find $R_1, \ldots, R_n$ such that the supervised system $P \times R_1 \times \cdots \times R_n$ satisfies the specification S and for each i, R_i satisfies (C$_i$) and (O$_i$). More generally, in our setting, a decentralized control problem is:

> Given a plant P and modal-loop formulas Φ, Ψ_1, $\ldots$,Ψ_n, do there exist controllers R_i satisfying Ψ_i $(i = 1, \ldots, n)$ such that $P \times R_1 \times \cdots \times R_n$ satisfies Φ?

To solve this problem we show how to construct a formula Φ/Ψ which is satisfied by a process R if and only if there exists a process P such that $P \models \Psi$ and $P \times R \models \Phi$. So, in the case when Ψ has only one model P, the meaning of Φ/Ψ is equivalent to Φ/P. If Ψ has more models then our construction works only in the case when Ψ is a formula of the standard μ-calculus, i.e., Ψ does not mention loops. Without this restriction the formula Φ/Ψ may not exist, since the set of all the processes R as described above may not be regular.

The construction of Φ/Ψ allows us to solve a decentralized control problem when at most one of the formulas Ψ_i contains loop predicates. We show that if one allows two such formulas then the existence of a solution to the problem is undecidable. Similar undecidability results in case of two controllers with partial observation have been obtained independently by Thistle and Lamouchi [11] and Tripakis [19].

This work extends the framework of Ramadge and Wonham [14] in two directions. We consider infinite behaviours and arbitrary regular specifications, while the standard framework deals only with specifications on the set of finite paths of processes. We also allow dynamic changes of the set of observable and controllable events.

Kupferman and Vardi [9, 10] consider a problem very similar to the problem **P**. They use different terminology, still it essentially amounts to the same setting but with a fixed set of observable and controllable actions. They do not consider the problem of synthetizing decentralized controllers.

Pnueli and Rosen [13] consider the problem of decentralized synthesis on given arhitectures. They show several decidablility/undecidability results depending on the shape of the architecture. Their setting is quite different from the one we have here. It is not clear that their architectures can simulate controllability/observability assumptions. There are also architectures that cannot be expressed by controllability/observability assumptions. They consider only linear time specifications.

Finally Maler, Pnueli and Sifakis [12] consider the problem of centralized synthesis in the presense of time constraints. This is an extension we have not pursued here. They consider only linear time specifications and only the case when all the actions are observable.

The second section of the paper contains only basic notions about processes and automata. After defining processes and their product we introduce simple automata. These are just standard tree automata over trees of bounded arity. As we do not assume that the trees are complete, the automata have means to check if an edge is present. We give the semantics of these automata in terms of games, and we recall their main properties: elimination of alternation and decidability of satisfiability. Next we introduce a special kind of automata called *cut automata*, which serve as a bridge between simple automata and loop automata and allow us to derive properties of loop automata from properties of simple automata. At the end of this section we introduce loop automata. We also show their two properties: elimination of alternation and decidability of satisfiability.

In the third section we define the quotient of a loop automaton over a process and over a simple automaton and we characterize the set of models of these quotients. In the fourth section we show how to use these quotients to solve some control problems, but we also give an example of undecidable control problem. We end the paper by some considerations on the complexity of synthesizing controllers.

2 Processes and automata

2.1 Processes

Let A be a finite set of events and let Λ be a nonempty finite set of labels (for instance Λ may be the powerset of a finite set of state properties).

A process is a tuple $P = \langle A, \Lambda, S, s^0, e, \lambda \rangle$ where e is a partial mapping from $S \times A$ to S defining the transitions of the process; and λ is a mapping from S to Λ defining the labelling of the states of the process. The state $s^0 \in S$ is the initial state of the process.

We denote by $out_P(s)$ the set of all $a \in A$ such that $e(s, a)$ is defined, and by $loop_P(s)$ the set of all $a \in out_P(s)$ such that $e(s, a) = s$.

If $P_1 = \langle A, \Lambda_1, S_1, s_1^0, e_1, \lambda_1 \rangle$ and $P_2 = \langle A, \Lambda_2, S_2, s_2^0, e_2, \lambda_2 \rangle$ are two processes, their product $P_1 \times P_2$ is the process $\langle A, \Lambda_1 \times \Lambda_2, S_1 \times S_2, (s_1^0, s_2^0), e, \lambda \rangle$ where $\lambda(s_1, s_2) = (\lambda_1(s_1), \lambda_2(s_2))$ and $e((s_1, s_2), a)$ is defined and equal to (s_1', s_2') if and only if for $i = 1, 2$, $e_i(s_i, a)$ is defined and equal to s_i'. It follows that $out_{P_1 \times P_2}(s_1, s_2) = out_{P_1}(s_1) \cap out_{P_2}(s_2)$ and that $loop_{P_1 \times P_2}(s_1, s_2) = loop_{P_1}(s_1) \cap loop_{P_2}(s_2)$.

2.2 Simple automata on processes

The automata introduced in this section are the usual alternating automata on trees of bounded degree. The degree of a node is bounded by the size of A, with each element of A interpreted as a different direction. We will define runs of these automata not just on trees but on all processes. It will be clear from the definition that such automata cannot distinguish between a process and an unwinding of a process into a tree. One more thing to note is that nodes in processes may have different degrees, and thus automata must have means to check if an edge exists.

2.2.1 Definition

A simple automaton on processes is a tuple:

$$\mathcal{A} = \langle A, \Lambda, Q, Q^{\exists}, Q^{\forall}, q^0, \delta : Q \times \Lambda \to \mathcal{P}(Moves(A, Q)), Acc \rangle$$

where $Q^{\exists}$, $Q^{\forall}$ form a partition of the finite set of states Q into existential and universal states and where $Acc \subseteq Q^{\omega}$ is a regular set of infinite sequences over Q. The set A of actions is the set of directions in which the automaton can proceed and the set Λ is the set of possible labels of the processes' nodes. The state q^0 is the initial state of the automaton and δ is the transition function assigning to each state and label a set of possible moves, where:

$$Moves(A, Q) = ((A \cup \{\varepsilon\}) \times Q) \cup (A \times \{\rightarrow, \not\rightarrow\})$$

Intuitively, a move (a, q') means to go to direction a and change the state to q'. When $a = \varepsilon$, the automaton just changes state. A move $(a, \rightarrow)$ checks that there is a transition a from the current node of the process. A move $(a, \not\rightarrow)$ checks that there is no such transition.

Definition 1 A condition $Acc \subseteq Q^\omega$ is a *parity condition* if there is a function $r : Q \to \mathbb{N}$ such that a sequence $q_0, q_1, \ldots, q_n \ldots$ belongs to Acc if and only if the number $\limsup_n r(q_n)$ is even. An automaton is said to be a *parity automaton* if its accepting condition is a parity condition.

2.2.2 Semantics

The simplest way of formalizing the notions of a run and of an acceptance of an automaton is in terms of games. Given a process $P = \langle A, \Lambda, S, s^0, e, \lambda\rangle$ we define the *acceptance game* $G(\mathcal{A}, P) = \langle V_0, V_1, v^0, E, Acc_G\rangle$ as follows:

- The set V_0 of vertices for player 0 is $(Q^\exists \times S) \cup \{\bot\}$.
- The set V_1 of vertices for player 1 is $(Q^\forall \times S) \cup \{\top\}$.
- $v^0 = (q^0, s^0)$,
- From each vertex (q, s), for every $(a, q') \in \delta(q, \lambda(s))$ we have an edge in E to (q', s) if $a = \varepsilon$ or to $(q', e(s, a))$ if $e(s, a)$ is defined.
- From each vertex (q, s), for every $(a, \rightarrow) \in \delta(q, \lambda(s))$ we have and edge in E to $\top$ if $e(s, a)$ is defined and an edge to $\bot$ otherwise. For $(a, \not\rightarrow)$ we do the same but exchanging the roles of $\top$ and $\bot$.
- The winning condition Acc_G consists of the sequences

  $$(q_0, s_0)(q_1, s_1) \ldots$$

  such that the sequence $q_0 q_1 \ldots$ is in Acc, i.e., belongs to the acceptance condition of the automaton.

A *memory* for such a game is a set H, whose elements are called *histories*, containing an initial history h^0 and equipped with a history update mapping $hist : H \times (V_0 \cup V_1) \to H$.

A *strategy* with memory H is a partial mapping $\sigma : V_0 \times H \to V_1$ such that if $(q', s') = \sigma((q, s), h)$ is defined then there is an edge in E from (q, s) to (q', s'). A strategy σ is *finite-memory* if H is finite. It is said to be *memoryless* or *positional* if H is a singleton. In the latter case σ is a partial mapping from V_0 to V_1.

A *play from* v_0 *consistent with* σ is a finite or infinite sequence (v_0, h_0), $(v_1, h_1), \ldots$ such that

- $h_0 = h^0$ and $h_i = \mathit{hist}(h_{i-1}, v_{i-1})$, for every $i > 0$,
- if $v_i \in V_1$ then $(v_i, v_{i+1}) \in E$,
- if $v_i \in V_0$ then $v_{i+1} = \sigma(v_i, h_i)$,

and which is *maximal*, i.e.:

- if $v_i \in V_1$ and there is some v' such that $(v_i, v') \in E$ then v_{i+1} is defined and $(v_i, v_{i+1}) \in E$,
- if $v_i \in V_0$ and $\sigma(v_i, h_i)$ is defined then v_{i+1} is defined and equal to $\sigma(v_i, h_i)$.

It follows that a play is finite only if its last element (v_n, h_n) satisfies one of the two conditions:

- $v_n \in V_1$ and there is no edge from v_n in E,
- $v_n \in V_0$ and $\sigma(v_n, h_n)$ is undefined.

A play is won by player 0 if

- either it is finite and v_n belongs to V_1,
- or it is infinite and the sequence $v_0 v_1 \cdots$ belongs to Acc_G.

We say that a strategy is *winning* from a position v if player 0 wins all plays consistent with this strategy starting from (v, h^0).

We say that $\mathcal{A}$ *accepts* a process P if and only if there is a winning strategy from the initial position v^0 in the game $G(\mathcal{A}, P)$. The *language recognized by* $\mathcal{A}$, denoted $\mathit{Mod}(\mathcal{A})$, is the set of processes accepted by $\mathcal{A}$. Often we write $P \vDash \mathcal{A}$ instead of $P \in \mathit{Mod}(\mathcal{A})$, and such a P is called a *model* of $\mathcal{A}$.

Later on we will make an important use of the following classical result of game theory (see, for instance, [3, 6, 18, 1]).

Theorem 2
For every game $G(\mathcal{A}, P)$ there is a wining finite-memory strategy which is maximal, i.e., winning for all the vertices from which player 0 has a winning strategy. If $\mathcal{A}$ is a parity automaton then there is a winning maximal positional strategy.

2.2.3 Nondeterministic automata

Definition 3 An automaton as above is *nondeterministic* if for every label $l \in \Lambda$ we have: $\delta(q,l) \subseteq \{\varepsilon\} \times Q$ for every $q \in Q^{\exists}$; and $\delta(q,l) \subseteq Moves(A,Q) \setminus \{\varepsilon\} \times Q$ for every $q \in Q^{\forall}$. In the case of $q \in Q^{\forall}$ we additionally require that if $(a_1, q_1), (a_2, q_2) \in \delta(q,l)$, with $q_1, q_2 \in Q$, and $a_1 = a_2$ then $q_1 = q_2$. Moreover, we assume that the initial state is existential (i.e., in $Q^{\exists}$).

A nondeterministic automaton is *bipartite* if for every $l \in \Lambda$ we have: $\delta(q,l) \subseteq \{\varepsilon\} \times Q^{\forall}$ when $q \in Q^{\exists}$ and $\delta(q,l) \cap A \times Q \subseteq A \times Q^{\exists}$ when $q \in Q^{\forall}$.

The following theorem, sometimes referred to as the Simulation theorem is one of the major results of the theory of automata and of the modal μ-calculus (see, for instance, [1, 18])

Theorem 4
Every automaton is equivalent to a bipartite nondeterministic parity automaton.

A bipartite nondeterministic automaton is *complete* if $\forall q \in Q^{\forall}, \forall l \in \Lambda$, $\forall a \in A$, $\exists q' \in Q^{\exists} : (a, q') \in \delta(q,l)$.

Proposition 5 Every bipartite nondeterministic automaton can be made complete. If the given automaton is a parity automaton then so is the resulting complete automaton.

Proof
It is sufficient to add two states $q^{\exists}_{\top}$ and $q^{\forall}_{\top}$ which accept any process (i.e., for any l, $\delta(q^{\exists}_{\top}, l) = \{(\varepsilon, q^{\forall}_{\top})\}$ and $\delta(q^{\forall}_{\top}, l) = \{(a, q^{\exists}_{\top}) : a \in A\}$, to modify *Acc* accordingly, and to add $(a, q^{\exists}_{\top})$ to $\delta(q,l)$ if needed to "complete" the automaton. $\square$

2.2.4 Satisfiability of automata

Let $\mathcal{A}$ be a bipartite nondeterministic automaton, which we can assume complete by Proposition 5.

We define the game $G(\mathcal{A}) = \langle V_0, V_1, v^0, E, Acc_G \rangle$ as follows.

- $V_0 = Q^{\exists}$,
- $V_1 = Q^{\forall} \times \Lambda \times \mathcal{P}(A)$,
- $v^0 = q^0 \in V_0$,
- there is an edge in E from $q \in V_0$ to (q_1, l, L) if $(\varepsilon, q_1) \in \delta(q, l)$, and if for any $a \in A$, $(a, \rightarrow) \in \delta(q_1, l) \Rightarrow a \in L$ and $(a, \nrightarrow) \in \delta(q_1, l) \Rightarrow a \notin L$.
- there is an edge in E from (q_1, l, L) to q_2 if $(a, q_2) \in \delta(q_1, l)$ for some $a \in L$.
- Acc_G is the set of infinite sequences whose projection on Q is in Acc.

Theorem 6
$\mathcal{A}$ has a model if and only if there exists a winning strategy in $G(\mathcal{A})$.

Proof
Without loss of generality we may assume that $\mathcal{A}$ is complete.

Let us assume that $P \models \mathcal{A}$ and let σ be a winning strategy with finite memory H in the game $G(\mathcal{A}, P)$. Let $hist : H \times V \to H$ be the history update function for σ.

We are going to define the strategy σ' in $G(\mathcal{A})$ over the set of histories $H' = Q \times S_P \times H$. We will do it in such a way that whenever we have a play $(q_0, h'_0), ((q_1, l_1, L_1), h'_1), (q_2, h'_2) \ldots$ consistent with σ' in $G(\mathcal{A})$ then the sequence $h'_0, h'_1, \ldots$ will be a play in $G(\mathcal{A}, P)$ consistent with σ. Moreover we will guarantee that the first component of h'_i is q_i for all $i = 0, 1, \ldots$. This will show that every play in $G(\mathcal{A})$ consistent with σ' is won by player 0.

Suppose that we have a position q in $G(\mathcal{A})$, a state $s \in S_P$ and a history h such that $(q_1, s) = \sigma((q, s), h)$ is defined. Then we set $\sigma'(q, (q, s, h)) = (q_1, \lambda(s), out(s))$.

We also need to define the mapping $hist' : H' \times (V_0 \cup V_1) \to H'$. For $(q_1, l, L) \in V_1$ we set

$$hist'((q, s, h), (q_1, l, L)) = (q_1, s, hist(h, (q_1, s)))$$

For $q_2 \in V_0$ we set

$$hist'((q_1, s, h), q_2) = (q_2, s_2, hist(h, (q_2, s_2)))$$

where $s_2 = e(s, a)$ for $a \in out(s)$ such that $(a, q_2) \in \delta(q_1, \lambda(s))$. Actually there may be more than one a satisfying this condition. It does not matter which one we choose, so we can assume that we have some order on the set of actions A and we choose the smallest a that satisfies the condition.

Conversely, let us assume that there exists a winning strategy σ' with finite memory H' in the game $G(\mathcal{A})$. Let us define a process P as follows:

- $S_P = \{(q, h') \in Q^{\exists} \times H' : \sigma'(q, h') \text{ is defined.}\}$,
- The initial state is (q^0, h'^0), i.e., the initial state and the initial history respectively,
- $\lambda(q, h') = l$ where $(q_1, l, L) = \sigma(q, h')$,
- $e((q, h'), a) = (q_2, h_2)$ if $a \in L$ and $(a, q_2) \in \delta(q_1, l, L)$, with $(q_1, l, L) = \sigma(q, h')$ (in this case there is and edge from (q_1, l, L) to q_2 in $G(\mathcal{A})$).

Note that this process is deterministic: because of the requirement, given in Definition 3, for any a there is at most one q_2 such that $(a, q_2) \in \delta(\sigma(q, h'))$.

It is not difficult to construct from σ' a strategy σ in $G(\mathcal{A}, P)$ that is winning from every position $(q, (q, h))$ such that $\sigma'(q, h)$ is defined.

□

RemarkAs shown in the second part of the proof of the previous theorem, each winning finite-memory strategy in $G(\mathcal{A})$ defines a model of $\mathcal{A}$. The proof of the first part amounts to showing that up to bisimulation equivalence, every model of $\mathcal{A}$ can be obtained in this way.

Since the existence of a winning strategy in a game is decidable, and that if it exists such a strategy can be effectively computed (see the last section of this article on the complexity issues), a consequence of Theorem 6 is that the satisfiability problem is decidable.

Corollary 7 It is decidable to know whether a nondeterministic automaton $\mathcal{A}$ is satisfiable. Moreover, if it is, one can effectively find a process P such that $P \models \mathcal{A}$.

Definition 8 A nondeterministic automaton $\mathcal{A}$ is *pruned* if for each state q the automaton $\mathcal{A}_q$, obtained from $\mathcal{A}$ by taking q as an initial state, has a model.

Fact 9 Every nondeterministic automaton can be effectively pruned.

Proof
Let $Q_\emptyset$ be the set of all states that have no model. This set can be effectively computed because of Theorem 6. For any $q \in Q^\exists \setminus Q_\emptyset$ let $\delta'(q,l) = \delta(q,l) \setminus (\{\varepsilon\} \times Q_\emptyset)$. For any $q \in Q^\forall \setminus Q_\emptyset$, let $\delta'(q,l)$ be the set obtained by replacing in $\delta(q,l)$, each (a,q') with $q' \in Q_\emptyset$ by $\nrightarrow_a$. □

2.3 Cut automata

In this subsection we present some constructions on automata that we will need in the proofs. The goal is to reuse known results for simple automata in the context of loop automata that we will introduce in the next section. The first step is to code the information about the loops into the labels. Then we also study an operation that cuts out all the loops in the process.

Definition 10 A process $P = \langle A, \Lambda \times \mathcal{P}(A), S, s^0, e, \lambda\rangle$ over a set of labels $\Lambda \times \mathcal{P}(A)$ is *well-labelled* if for every $s \in S$, the second projection $\lambda(s)\!\downarrow_2$ of $\lambda(s)$ is equal to $loop_P(s)$.

With every process P over the label set Λ we associate a well-labelled process $\widehat{P}$ over the label set $\Lambda \times \mathcal{P}(A)$:

$$\widehat{P} = \langle A, \Lambda \times \mathcal{P}(A), S, s^0, e, \widehat{\lambda}\rangle$$

such that for every s, $\widehat{\lambda}(s) = (\lambda(s), loop_P(s))$.

Definition 11 We say that an automaton $\mathcal{A}$ over a set of labels $\Lambda \times \mathcal{P}(A)$ is a *cut automaton* if for every q, l, L: $\delta(q,(l,L)) \cap (L \times (Q \cup \{\rightarrow\})) = \emptyset$.

Definition 12 If P is a process over $\Lambda \times \mathcal{P}(A)$ then $Cut(P)$ is the process whose components are the same as in P but for the transition function where we make $Cut(e)(s,a)$ undefined if $a \in \lambda(s)\!\downarrow_2$ and we make $Cut(e)(s,a) = e(s,a)$ otherwise.

Fact 13 If $\mathcal{A}$ is a cut automaton then $P \models \mathcal{A}$ if and only if $Cut(P) \models \mathcal{A}$.

Proof
It is easy to check that the games $G(\mathcal{A}, P)$ and $G(\mathcal{A}, Cut(P))$ are identical.
□

Lemma 14 If $\mathcal{A}$ is a cut automaton then there exists a bipartite nondeterministic cut automaton equivalent to $\mathcal{A}$.

Proof
Let $\mathcal{B}$ be a bipartite nondeterministic automaton equivalent to $\mathcal{A}$. Let $\mathcal{B}'$ be the bipartite nondeterministic cut automaton whose components are the same as in $\mathcal{B}$, except that for any universal state q, $\delta'(q,(l,L)) = \delta(q,(l,L)) \setminus L \times (Q \cup \{\rightarrow\})$.

It is obvious that for any P, the games $G(\mathcal{B}, \mathit{Cut}(P))$ and $G(\mathcal{B}', \mathit{Cut}(P))$ are identical, hence, $\mathit{Cut}(P) \models \mathcal{A}$ if and only if $\mathit{Cut}(P) \models \mathcal{B}$ if and only if $\mathit{Cut}(P) \models \mathcal{B}'$.

It follows that $P \models \mathcal{A}$ if and only if $\mathit{Cut}(P) \models \mathcal{A}$ if and only if $\mathit{Cut}(P) \models \mathcal{B}'$ if and only if $P \models \mathcal{B}'$.

□

Definition 15 If $\mathcal{A}$ is an automaton over $\Lambda \times \mathcal{P}(A)$ then $\mathit{Cut}(\mathcal{A})$ is the cut automaton obtained by the following modification of the transition function of $\mathcal{A}$. For each universal state q we have:

$$\begin{aligned}\mathit{Cut}(\delta)(q,(l,L)) =& \delta(q,(l,L)) \setminus (L \times (Q \cup \{\rightarrow\})) \\ & \cup \{(\varepsilon, q') : (a, q') \in \delta(q,(l,L)), a \in L\}\end{aligned}$$

For each existential state q we have:

$$\begin{aligned}\mathit{Cut}(\delta)(q,(l,L)) =& \delta(q,(l,L)) \setminus (L \times (Q \cup \{\rightarrow, \nrightarrow\})) \\ & \cup \{(\varepsilon, q') : (a, q') \in \delta(q,(l,L)), a \in L\} \\ & \cup \{(\varepsilon, q_\top) : \delta(q,(l,L)) \cap (L \times \{\rightarrow\}) \neq \emptyset\}\end{aligned}$$

where $q_\top$ is a universal state with $\delta(q_\top,(l,L)) = \emptyset$.

Fact 16 For every well-labelled process P we have:

$$P \vDash \mathcal{A} \quad \text{if and only if} \quad P \vDash \mathit{Cut}(\mathcal{A})$$

Proof
It is easy to check that if P is well-labelled, the games $G(\mathcal{A}, P)$ and $G(\mathit{Cut}(\mathcal{A}), P)$ are identical. □

2.4 Automata with loop testing

2.4.1 Definition

Here we extend the automata with the feature of testing for loops in the process. We will call these automata *loop automata*. Loop automata have the same components as simple automata, but for a transition function which has extended range:

$$\delta : Q \times \Lambda \to \mathcal{P}(Moves(A, Q) \cup A \times \{\circlearrowleft, \overline{\circlearrowleft}\})$$

We will write $\to_a$, $\nrightarrow_a$, $\circlearrowleft_a$ and $\overline{\circlearrowleft}_a$ instead of $(a, \to)$, $(a, \nrightarrow)$, $(a, \circlearrowleft)$ and $(a, \overline{\circlearrowleft})$, respectively. Intuitively, the meaning of a move $\circlearrowleft_a$ is to check that there is a loop on a action. Similarly $\overline{\circlearrowleft}_a$ checks that there is no such loop.

2.4.2 Semantics

The semantics of such an automaton is defined via game as before. Given a process P, the game $G(\mathcal{A}, P)$ is defined as for simple automata but with additional edges given by the clause:

- From each vertex (q, s), for every $(a, \circlearrowleft) \in \delta(q, \lambda(s))$ we have an edge to $\top$ if $e(s, a) = s$ and an edge to $\bot$ otherwise. For $(a, \overline{\circlearrowleft})$ we have the same but exchanging the roles of $\top$ and $\bot$.

2.4.3 Nondeterministic loop automata

Definition 17 A loop automaton is *bipartite nondeterministic* if it satisfies the condition of Definition 3 and if moreover, for any universal state q and any label l: if $\circlearrowleft_a \in \delta(q, l)$ then $\delta(q, l) \cap (\{a\} \times Q) = \emptyset$.

Therefore, a bipartite nondeterministic automaton is *complete* if $\forall q \in Q^\forall, \forall l \in \Lambda$, $\forall a \in A$, if $\circlearrowleft_a \notin \delta(q, l)$ then $\exists q' \in Q : (a, q') \in \delta(q, l)$.

The proof of the following result is similar to the proof of Proposition 5

Proposition 18 Every bipartite nondeterministic automaton can be made complete. If we are given a parity automaton, the resulting complete automaton is also a parity automaton.

Definition 19 With every loop automaton $\mathcal{A} = \langle A, \Lambda, Q, Q^\exists, Q^\forall, q^0, \delta, Acc\rangle$, we associate a simple automaton $\widehat{\mathcal{A}}$ over the alphabet $\Lambda \times \mathcal{P}(A)$.

This automaton will have the same components but for two new states $q_\bot$, $q_\top$ and a transition function changed as described below. We make $q_\bot$ and $q_\top$ an existential and a universal state respectively. We set $\widehat{\delta}(q_\bot, (l, L)) = \widehat{\delta}(q_\top, (l, L)) = \emptyset$ for every label $(l, L) \in \Lambda \times \mathcal{P}(A)$. For any other state q we make $\widehat{\delta}(q, (l, L))$ contain $\delta(q, l) \setminus (A \times \{\circlearrowleft, \overline{\circlearrowleft}\})$ and we add:

- $(\varepsilon, q_\bot) \in \widehat{\delta}(q, (l, L))$ if $q \in Q^\forall$ and $\{a : \circlearrowleft_a \in \delta(q, l)\} \not\subseteq L$ or $L \cap \{a : \overline{\circlearrowleft}_a \in \delta(q, l)\} \neq \emptyset$,

- $(\varepsilon, q_\top) \in \widehat{\delta}(q,(l,L))$ if $q \in Q^\exists$
  and $\{a : \overline{\circlearrowleft}_a \in \delta(q,l)\} \not\subseteq L$ or $L \cap \{a : \circlearrowleft_a \in \delta(q,l)\} \neq \emptyset$,

Proposition 20 For every loop automaton $\mathcal{A}$ over a label alphabet Λ and every process P,

$$P \vDash \mathcal{A} \quad \text{if and only if} \quad \widehat{P} \vDash \widehat{\mathcal{A}}$$

Proof

For every process P the games $G(\mathcal{A}, P)$ and $G(\widehat{\mathcal{A}}, \widehat{P})$ are "almost" isomorphic. For every position (q,s) there are the same edges from (q,s) in $G(\mathcal{A}, P)$ and in $G(\widehat{\mathcal{A}}, \widehat{P})$ with the exception that there may be an edge to $\bot$ or to $\top$ in $G(\mathcal{A}, P)$ and this will be matched by an edge to $(q_\bot, s)$ or $(q_\top, s)$, respectively, in $G(\widehat{\mathcal{A}}, \widehat{P})$. Still it is easy to see there is a winning strategy from a position (q,s) in $G(\mathcal{A}, P)$ if and only if there is a winning strategy from this position in $G(\widehat{\mathcal{A}}, \widehat{P})$. $\square$

Definition 21 For any nondeterministic cut automaton $\mathcal{B}$ over $\Lambda \times \mathcal{P}(A)$ we construct the nondeterministic loop automaton $\mathcal{B}^\vee$ over Λ as follows.

The set of states of $\mathcal{B}^\vee$ is $Q^\exists \cup (Q^\forall \times \mathcal{P}(A))$. The states from $Q^\exists$ are existential, the states from $Q^\forall \times \mathcal{P}(A)$ are universal. The acceptance set is the set of infinite sequences in $Q^\exists \cup (Q^\forall \times \mathcal{P}(A))$ whose projection on $(Q^\exists \cup Q^\forall)^\omega$ are in Acc.

The transition function for a universal state q is defined by:

$$\delta^\vee((q,L),l) = \delta(q,(l,L)) \cup \{\circlearrowleft_a : a \in L\} \cup \{\overline{\circlearrowleft}_a : a \notin L\}$$

For an existential state q we set:

$$\delta^\vee(q,l) = \{(\varepsilon,(q',L)) : (\varepsilon, q') \in \delta(q,(l,L)),\ L \subseteq A\}$$

Fact 22 For any nondeterministic cut automaton $\mathcal{B}$, $\mathcal{B}$ is equivalent to $\widehat{\mathcal{B}^\vee}$.

Proof

For any universal state q of $\mathcal{B}$ and for any $L' \subseteq A$, state (q, L') is an universal state of $\mathcal{B}^\vee$ and of $\widehat{\mathcal{B}^\vee}$. Moreover

$$\delta^\vee((q,L'),l) = \delta(q,(l,L')) \cup \{\circlearrowleft_a : a \in L'\} \cup \{\overline{\circlearrowleft}_a : a \notin L'\}$$

It follows that $\widehat{\delta^\vee}((q,L'),(l,L))$ contains $(\varepsilon, q_\bot)$ if and only if $L \neq L'$. Otherwise $\widehat{\delta^\vee}((q,L),(l,L)) = \delta(q,(l,L))$.

For any existential state q of $\mathcal{B}$:

$$\widehat{\delta^\vee}(q,(l,L)) = \delta^\vee(q,(l,L)) = \{(\varepsilon,(q',L')) : (\varepsilon,q') \in \delta(q,(l,L))\}$$

But, for any process P, in an existential position (q,s) of the game $G(\widehat{\mathcal{B}^\vee},P)$ with $\lambda(s) = (l,L)$, every move to $((q',L'),s)$ with $L' \neq L$ leads to a losing position. Thus we do not change the semantics of $\widehat{\mathcal{B}^\vee}$ if we set $\widehat{\delta^\vee}(q,(l,L)) = \{(\varepsilon,(q',L)) : (\varepsilon,q') \in \delta(q,(l,L))\}$. Now, the two games $G(\widehat{\mathcal{B}^\vee},P)$ and $G(\mathcal{B},P)$ are isomorphic since each universal position (q,s) of the second one is associated with $((q,loop_P(s)),s)$ in the first one.

□

Theorem 23
For every loop automaton $\mathcal{A}$ there exists a bipartite nondeterministic loop automaton equivalent to $\mathcal{A}$.

Proof

By Lemma 14 there exists a nondeterministic cut automaton $\mathcal{B}$ equivalent to $Cut(\widehat{\mathcal{A}})$.

By Proposition 20 and Facts 16, 13 and 22, for any process P over Λ, we have

$$\begin{array}{llllll} P \models \mathcal{A} & \text{if and only if} & \widehat{P} \models \widehat{\mathcal{A}} & \text{if and only if} & Cut(\widehat{P}) \models \mathcal{B} \\ & \text{if and only if} & \widehat{P} \models \mathcal{B} & \text{if and only if} & \widehat{P} \models \widehat{\mathcal{B}^\vee} \\ & \text{if and only if} & P \models \mathcal{B}^\vee. \end{array}$$

□

The decidability of satisfiability of bipartite nondeterministic loop automata is proved in a way quite similar to the case of simple automata but for a small difference in the definition of the game $G(\mathcal{A})$: in a universal position, to the subset of A corresponding to $out(s)$, we add a second subset corresponding to $loop(s)$.

If we assume that $\mathcal{A}$ is complete (see Proposition 18), we define the game $G(\mathcal{A}) = \langle V_0, V_1, v^0, E, Acc_G\rangle$ as follows.

- $V_0 = Q^\exists$,
- $V_1 = Q^\forall \times \Lambda \times \mathcal{P}(A) \times \mathcal{P}(A)$,
- $v^0 = q^0 \in V_0$,

- there is an edge in E from $q \in V_0$ to (q', l, L, L') if there exists $l \in \Lambda$ such that $(\varepsilon, q') \in \delta(q, l)$, $L' \subseteq L$, and if for any $a \in A$, $\rightarrow_a \in \delta(q', l) \Rightarrow a \in L$, $\nrightarrow_a \in \delta(q', l) \Rightarrow a \notin L$, $\circlearrowleft_a \in \delta(q', l) \Rightarrow a \in L'$, $\overline{\circlearrowleft}_a \in \delta(q', l) \Rightarrow a \notin L'$,
- there is an edge in E from (q, l, L, L') to q' if and only if $(a, q') \in \delta(q, l)$ for some $a \in L \setminus L'$,
- Acc_G is the set of infinite sequences whose projection on Q is in Acc.

Theorem 24
A complete bipartite nondeterministic loop automaton $\mathcal{A}$ has a model if and only if there exists a winning strategy in $G(\mathcal{A})$.

Here again, all models of $\mathcal{A}$ are defined by winning finite-memory strategies in $G(\mathcal{A})$ up to a bisimulation equivalence modified to take into account the existence of loops.

2.5 Example

As an example of properties that we can express in our setting we consider the diagnosability problem [16]. Suppose that we have a set $F \subseteq \Lambda$ of labels that mark failure states of a plant, i.e., a failure state s has $\lambda(s) \in F$. We also have a set $A_o \subseteq A$ of actions of the plant that we can observe. The objective is to determine if the plant passed through a failure state provided we can only observe actions from A_o. The diagnosability problem is to decide if it is possible to reach this objective for a given plant P.

We can reformulate the problem in terms of controllers. Say that an *observer* is a controller that cannot forbid any action. So the problem is to construct for a given plant P an observer R over the set of labels $\Lambda' = \{w, f\}$ such that for every infinite path $v_0, v_1, \ldots$ of the controlled plant $P \times R$ we have:

- if $\{\lambda(v_0), \lambda(v_1), \ldots\} \cap (F \times \Lambda') = \emptyset$ then $\{\lambda(v_0), \lambda(v_1), \ldots\} \subseteq (\Lambda \times \{w\})$ (if there is no failure then the controller does not signal it),
- if $\lambda(v_i) \in F \times \Lambda'$ then there is $j \geq i$ with $\lambda(v_j) \in \Lambda \times \{f\}$ (if a failure occurs then it is reported).

In the original definition of the problem it was required that there is a constant n depending only on the plant such that in the last clause we have $j < i + n$. Intuitively this means that there is a uniform bound on the delay between the occurrence of an error and its notification. We do not have an explicit requirement on a uniform bound in our specification. Still, as $P \times R$ is finite, there is such a bound by König's lemma (if there were an arbitrary long delay between an error and its notofication then there would also be an infinite delay).

The following automaton $\mathcal{A}$ describes the desired behaviour of the observed plant. The states of $\mathcal{A}$ are $Q = \{q_0, q_1\}$, all of the states are universal. The transition function is defined by

- $\delta(q_0, (l, l')) = \{(a, q_0) : a \in A\}$ for $l \notin F$ and $\delta(q_0, (l, l')) = \{(a, q_1) : a \in A\}$ for $l \in F$,
- $\delta(q_1, (l, w)) = \{(a, q_1) : a \in A\}$ and $\delta(q_1, (l, f)) = \{(a, q_0) : a \in A\}$

The acceptance condition is that q_0 must appear infinitely often.

The automaton $\mathcal{A}$ stays in the state q_0 until a failure appears. Then it changes state to q_1 and stays there until the failure is notified. Then it changes state back to q_0.

Now we want to specify that we are interested in a controller which is an observer, so it cannot prohibit an action and it can see only actions in A_o. This is done with another automaton $\mathcal{B}$. The automaton has only one state $\{q^0\}$ and this state is universal. The transition function is defined by:

$$\delta(q^0, l') = \{(a, q^0) : a \in A\} \cup \{\rightarrow_a : a \in A_o\} \cup \{\circlearrowleft_a : a \in A \setminus A_o\}$$

The diagnosability problem may be now rephrased as: given P, is there a controller R such that $P \times R \vDash \mathcal{A}$ and $R \vDash \mathcal{B}$. We will show in the next sections that R is such a controller if and only if it satisfies $\mathcal{A}/P$ and $\mathcal{B}$. In particular such an R exists if and only if $Mod(\mathcal{A}/P) \cap Mod(\mathcal{B})$ is not empty.

3 Quotients of loop automata

In this section we introduce the quotient operation which is an adjoint operation to the product. For each loop automaton $\mathcal{A}$ and a process P we will construct an automaton $\mathcal{A}/P$ such that for every process R:

$$R \vDash \mathcal{A}/P \quad \text{if and only if} \quad P \times R \vDash \mathcal{A}$$

Then we generalize this operation to $\mathcal{A}/\mathcal{B}$, where $\mathcal{B}$ is a simple automaton, so that $Mod(\mathcal{A}/\mathcal{B}) = \bigcup_{P \in Mod(\mathcal{B})} Mod(\mathcal{A}/P)$.

3.1 Quotient over processes

Definition 25 We define the automaton $\mathcal{A}/P$ where the automaton $\mathcal{A} = \langle A, \Lambda\times\Lambda', Q, Q^\exists, Q^\forall, q^0, \delta, Acc\rangle$ is a loop automaton and $P = \langle A, \Lambda, S, s^0, e, \lambda\rangle$ is a process.

The set of labels of $\mathcal{A}/P$ is Λ'. Its set of states is $(Q\times S)\cup\{q_\perp, q_\top\}$ with the set of existential states being $(Q^\exists \times S)\cup\{q_\perp\}$.

The acceptance condition $Acc_{/P}$ of $\mathcal{A}/P$ is the set of all infinite sequences $(q_0, s_0)(q_1, s_1)\dots$ in $(Q\times S)^\omega$ such that $q_0q_1\dots$ belongs to Acc.

Next we define the transition function of $\mathcal{A}/P$. For a universal state (q, s) we set

$$\delta_{/P}((q, s), l') = \{(\varepsilon, q_\perp)\}$$

if there exists $\rightarrow_a \in \delta(q, (\lambda(s), l'))\setminus out(s)$ or there exists $\circlearrowleft_a \in \delta(q, (\lambda(s), l'))\setminus loop(s)$. Otherwise we set

$$\begin{aligned}\delta_{/P}((q, s), l') =&\{(a, (q', s')) : (a, q') \in \delta(q, (\lambda(s), l')), e(s, a) = s'\}\\ &\cup \delta(q, (\lambda(s), l')) \cap (A \times \{\rightarrow, \circlearrowleft\})\\ &\cup \delta(q, (\lambda(s), l')) \cap (out(s) \times \{\nrightarrow\})\\ &\cup \delta(q, (\lambda(s), l')) \cap (loop(s) \times \{\overline{\circlearrowleft}\})\end{aligned}$$

The transition function for an existential state (q, s) is defined dually. We set

$$\delta_{/P}((q, s), l) = \{(\varepsilon, q_\top)\}$$

if there exists $\nrightarrow_a \in \delta(q, (\lambda(s), l'))\setminus out(s)$ or there exists $\overline{\circlearrowleft}_a \in \delta(q, (\lambda(s), l'))\setminus loop(s)$. Otherwise we set

$$\begin{aligned}\delta_{/P}((q, s), l') =&\{(a, (q', s')) : (a, q') \in \delta(q, (\lambda(s), l')), e(s, a) = s'\}\\ &\cup \delta(q, (\lambda(s), l')) \cap (A \times \{\nrightarrow, \overline{\circlearrowleft}\})\\ &\cup \delta(q, (\lambda(s), l')) \cap (out(s) \times \{\rightarrow\})\\ &\cup \delta(q, (\lambda(s), l')) \cap (loop(s) \times \{\circlearrowleft\})\end{aligned}$$

Finally, we set $\delta_{/P}(q_\perp, l') = \delta_{/P}(q_\top, l') = \emptyset$.

Fact 26 If $\mathcal{A}$ is nondeterministic, then $\mathcal{A}/P$ can be made nondeterministic.

Proof
The automaton defined above is not nondeterministic because it may have $\delta_{/P}((q', s), l') = \{(\varepsilon, q_\perp)\}$ for some universal state q'. In this case clearly the automaton does not accept anything from (q', s). So, for every $q \in Q^\exists$, we

can remove the move $(\varepsilon, (q', s))$ from all $\delta_{/P}((q, s), l')$ without changing the semantics of the automaton. Then the value of the transition function for (q', s) and l becomes irrelevant and we can set $\delta_{/P}((q', s), l') = \emptyset$.

The other thing to check is that whenever $(a, (q', s')) \in \delta_{/P}((q, s), l')$ then $\circlearrowleft_a \notin \delta_{/P}((q, s), l')$. If $(a, (q', s')) \in \delta_{/P}((q, s), l')$ then by the definition of $\delta_{/P}$ we have $(a, q') \in \delta(q, (\lambda(s), l'))$. But $\mathcal{A}$ is nondeterministic, so $\circlearrowleft_a \notin \delta(q, (\lambda(s), l'))$. Hence $\circlearrowleft_a \notin \delta_{/P}((q, s), l')$. $\square$

Theorem 27
For every loop automaton $\mathcal{A}$ over the set of labels $\Lambda \times \Lambda'$, every process P over the set of labels Λ, and every process R over the set of labels Λ',

$$P \times R \vDash \mathcal{A} \quad \textit{if and only if} \quad R \vDash \mathcal{A}/P$$

Proof

We want to show that for every process R the games $G_\times = G(\mathcal{A}, P \times R)$ and $G_/ = G(\mathcal{A}/P, R)$ are isomorphic but for some parts that are immediately winning or immediately losing. Take positions $(q, (s_1, s_2))$ and $((q, s_1), s_2)$ in $G_\times$ and $G_/$ respectively. There are several cases to consider:

- If there is an edge from $((q, s_1), s_2)$ to $(q_\perp, s_2)$ in $G_/$ then q is a universal state and in $G_\times$ there is an edge from $(q, (s_1, s_2))$ to $\perp$.
- Similarly, if there is an edge to $(q_\top, s_2)$ in $G_/$ then q is existential and in $G_\times$ there is an edge from $(q, (s_1, s_2))$ to $\top$.
- Otherwise, a direct inspection of the definitions shows that there is an edge from $(q, (s_1, s_2))$ to $(q, (s'_1, s'_2))$ in $G_\times$ if and only if there is an edge from $((q, s_1), s_2)$ to $((q, s'_1), s'_2)$ in $G_/$.

Using the above three observations it is easy to see that there is a winning strategy from $(q, (s_1, s_2))$ in $G_\times$ if and only if there is one from $((q, s_1), s_2)$ in $G_/$. $\square$

3.2 Quotient over simple automata

Definition 28 Let $\mathcal{A}$ be a bipartite nondeterministic parity loop automaton over $\Lambda \times \Lambda'$ and let $\mathcal{C}$ be a bipartite nondeterministic parity cut automaton over $\Lambda' \times \mathcal{P}(A)$. We are going to define an automaton $\mathcal{A}/\mathcal{C}$.

Without loss of generality we may assume that $\mathcal{C}$ is complete, i.e., for any universal state q, if $a \notin L$, then $\delta_\mathcal{C}(q, (l, L)) \cap (\{a\} \times Q) \neq \emptyset$ (see

Proposition 18). Moreover, we may assume that $\mathcal{C}$ is pruned (see Definition 8 and Fact 9).

We will also assume that $\mathcal{A}$ satisfies a strange condition which is useful at one point of the following proof. Namely, we will require that for all states q, q_1, q_2, every action a, and every label (l, l'), if $(\varepsilon, q_1) \in \delta_{\mathcal{A}}(q, (l, l'))$ and $(a, q_2) \in \delta_{\mathcal{A}}(q_1, (l, l'))$ then $q \neq q_2$. By duplicating some states of the automaton one can easily guarantee this condition.

We construct the automaton $\mathcal{A}/\mathcal{C}$ as follows.

The sets of existential and universal states of $\mathcal{A}/\mathcal{C}$ are respectively:

$$Q^{\exists}_{/} = Q^{\exists}_{\mathcal{A}} \times Q^{\exists}_{\mathcal{C}} \quad \text{and} \quad Q^{\forall}_{/} = Q^{\forall}_{\mathcal{A}} \times Q^{\forall}_{\mathcal{C}} \times \Lambda' \times \mathcal{P}(A)$$

The acceptance condition $Acc_{/}$ of $\mathcal{A}/\mathcal{C}$ contains all those sequences over $Q^{\exists}_{/} \cup Q^{\forall}_{/}$ for which the projection on the first component belongs to $Acc_{\mathcal{A}}$ and the projection on the second component belongs to $Acc_{\mathcal{C}}$.

For an existential state (q, q') the transition function $\delta_{/}((q, q'), l)$ contains all the tuples $(\varepsilon, (q_1, q'_1, l', L'))$ such that

- $l' \in \Lambda'$,
- $(\varepsilon, q_1) \in \delta_{\mathcal{A}}(q, (l, l'))$,
- $L' = \{\circlearrowleft_a : \circlearrowleft_a \in \delta_{\mathcal{A}}(q_1, (l, l'))\}$,
- $(\varepsilon, q'_1) \in \delta_{\mathcal{C}}(q', (l', L'))$,
- if $\nrightarrow_a \in \delta_{\mathcal{C}}(q'_1, (l', L'))$ then $\rightarrow_a \notin \delta_{\mathcal{A}}(q_1, (l, l'))$,

For a universal state (q, q', l', L') let us first denote $M = \delta_{\mathcal{A}}(q, (l, l'))$ and $M' = \delta_{\mathcal{C}}(q', (l', L'))$. We put into $\delta_{/}((q, q', l', L'), l)$ the moves:

- $(a, (q_1, q'_1))$, if $\rightarrow_a \in M \cup M'$, $(a, q_1) \in M$, and $(a, q'_1) \in M'$,
- $\rightarrow_a$ and $\circlearrowleft_a$, if they appear in M,
- $\nrightarrow_a$, if $\nrightarrow_a \in M$ and $\rightarrow_a \in M'$,

Fact 29 The automaton $\mathcal{A}/\mathcal{C}$ is bipartite nondeterministic.

Proof
If there exist a, q_1, q'_1 such that $\circlearrowleft_a$ and $(a, (q_1, q'_1))$ are both contained in $\delta_{/}((q, q', l', L'), l)$ then $\circlearrowleft_a$ and (a, q_1) are both in $\delta_{\mathcal{A}}(q, (l, l'))$. This is impossible since $\mathcal{A}$ is nondeterministic. $\square$

Proposition 30 Let $\mathcal{A}$ and $\mathcal{C}$ be as in the previous definition. For every process P over Λ: $P \models \mathcal{A}/\mathcal{C}$ if and only if there is a process R over Λ' such that $\widehat{R} \models \mathcal{C}$ and $P \times R \models \mathcal{A}$.

Proof
Suppose that there is a winning strategy in $G(\mathcal{A}/\mathcal{C}, P)$ for some process P. We want to find a process R such that there is a winning strategy in $G(\mathcal{A}, P \times R)$ and in $G(\mathcal{C}, \widehat{R})$.

Let $\sigma : (Q_/^\exists \times S_P) \times H \to Q_/^\forall \times S_P$ be a winning strategy in $G(\mathcal{A}/\mathcal{C}, P)$ and let $hist : H \times (Q_/ \times S_P) \to H$ be the corresponding history update function. As the acceptance conditions of the game are regular, we know that it is enough to have a finite set H of histories.

We have assumed that $\mathcal{C}$ is pruned, so for every state q' of $\mathcal{C}$ there is a process $R_{q'}$ with $\widehat{R_{q'}} \models \mathcal{C}_{q'}$, where $\mathcal{C}_{q'}$ is $\mathcal{C}$ with the initial state changed to q'.

The states of R will be the quadruples $(q, q', s, h) \in Q_\mathcal{A} \times Q_\mathcal{C} \times S_P \times H$ such that $\sigma((q, q'), s, h)$ is defined. Additionally R will contain the states of all $R_{q'}$ for every $q' \in Q_\mathcal{C}$. The initial state of R is $(q_\mathcal{A}^0, q_\mathcal{C}^0, s^0, h^0)$, where the first three components are the initial states of $\mathcal{A}$, $\mathcal{C}$ and P respectively and the last component is the initial history.

To define the transition function of R from $(q, q', s, h) \in Q_\mathcal{A} \times Q_\mathcal{C} \times S_P \times H$ consider $((q_1, q_1', l', L'), s) = \sigma((q, q'), s, h)$. Let us denote $M = \delta_\mathcal{A}(q_1, (l, l'))$, $M' = \delta_\mathcal{C}(q_1', (l', L'))'$ and $h_1 = hist(h, ((q, q'), s))$. We set:

1. $e_R((q, q', s, h), a) = (q, q', s, h)$ if $a \in L'$,
2. $e_R((q, q', s, h), a) = (q_2, q_2', s_2, h_2)$ if $\rightarrow_a \in M \cup M'$, $(a, q_2') \in M'$, $(a, q_2) \in M$, $e_P(s, a) = s_2$ and $h_2 = hist(h_1, ((q_1, q_1', l', L'), s))$,
3. $e_R((q, q', s, h), a) = s_{q_1'}^0$ if $\rightarrow_a \in M \cup M'$, $(a, q_1') \in M'$ and $(\{a\} \times Q) \cap M = \emptyset$. (Here $s_{q_1'}^0$ is the initial state of $R_{q_1'}$.)

As expected, the transitions from a state in $R_{q'}$ are the same as in $R_{q'}$.

The following two claims show that R satisfies the properties stated in the theorem.

Claim 30.1 There is a winning strategy in $G(\mathcal{A}, P \times R)$ from the position $(q^0, (s^0, (q_\mathcal{A}^0, q_\mathcal{C}^0, s^0, h^0)))$.

We will describe a strategy from the position $(q, (s, (q, q', s, h)))$ in $G(\mathcal{A}, P \times R)$ provided $\sigma((q, q'), s, h)$ is defined. We know that $\sigma((q_\mathcal{A}^0, q_\mathcal{C}^0), s^0, h^0)$ is defined.

Let $\sigma((q,q'),s,h) = ((q_1,q_1',l',L'),s)$. The strategy from the position

$$(q,(s,(q,q',s,h)))$$

is to move to $(q_1,(s,(q,q',s,h)))$.

We will show that this is a winning strategy by showing that each edge from $(q_1,(s,(q,q',s,h)))$ leads to a position of the from $(q_2,(s_2,(q_2,q_2',s_2,h_2)))$ such that there is an edge to $((q_2,q_2'),s_2)$ from $\sigma((q,q'),s)$ and

$$h_2 = \mathit{hist}(\mathit{hist}(h,((q,q'),s)),\sigma((q,q'),s)).$$

Let $M = \delta_{\mathcal{A}}(q,(l,l'))$ and $M' = \delta_{\mathcal{C}}(q',(l',L'))$. Suppose that there is an edge from $(q_1,(s,(q,q',s,h)))$ to $(q_2,(s_2,(q_3,q_3',s_3,h_3)))$. By definition, this means that for some letter a we have:

$$(a,q_2)\in M, \quad e_P(s,a)=s_2, \quad \text{and} \quad e_R((q,q',s,h),a) = (q_3,q_3',s_3,h_3).$$

We get immediately that $h_3 = h_2$. Let us examine what q_3, q_3' and s_3 might be. We have $\circlearrowleft_a \notin M$ since $(a,q_2)\in M$ and $\mathcal{A}$ is nondeterministic. Hence, $a\notin L'$ and the first clause of the definition of e_R cannot hold. Since $\mathcal{C}$ is complete, there is q_2' with $(a,q_2')\in M'$. We also must have $\rightarrow_a \in M\cup M'$ since there is an edge from (q,q',s) in R. Hence, we have $e_R((q,q',s,h),a) = (q_2,q_2',s_2,h_2)$. This means that $q_3=q_2$, $q_3'=q_3$ and $s_3=s_2$. So every edge from $(q,(s,(q,q',s,h)))$ leads to a node of the form $(q_2,(s,(q_2,q_2',s_2,h_2)))$ and it can be matched with an edge from $((q,q'),s)$ to $((q_2,q_2'),s_2)$.

We also want to check that there is no edge to $\perp$ from $(q_1,(s,(q,q',s,h)))$. If $\circlearrowleft_a \in M$ then $a\in L'$ and $a\in \mathit{loop}_R((q,q',s,h))$. As $((q,q',l',L'),s)$ is a winning position in $G(\mathcal{A}/\mathcal{C},P)$ we must have that $a\in \mathit{loop}_P(s)$. So $P\times R$ has a loop on action a in state $(s,(q,q',s,h))$.

If $\overline{\circlearrowleft}_a \in M$ then $\circlearrowleft_a \notin M$ so $a\notin L'$. Then we do not have a loop on action a in process R from (q,q',s,h).

If $\rightarrow_a \in M$ then $a\in \mathit{out}_P(s)$ because the position $((q,q',l',L'),s)$ is winning. We also have $a\in \mathit{out}_R(q,q',s,h)$ by the definition of R because we have assumed that $\mathcal{C}$ has the property that for every letter a, either $a\in L'$ or $(a,q_1')\in\delta_{\mathcal{C}}(q',(l',L'))$ for some q_1'.

Suppose $\not\rightarrow_a \in M$. If $\rightarrow_a \in M'$ then $a\notin \mathit{out}_P(s)$, because $((q,q',l',L'),s)$ is a winning position. Otherwise $\rightarrow_a \notin M\cup M'$, so R has no a transition from (q,q',s,h).

This finishes the proof of the above claim. So now we know that $P\times R\vDash \mathcal{A}$. It remains to show that $\widehat{R}\vDash\mathcal{C}$.

Claim 30.2 There is a winning strategy in $G(\mathcal{C}, \widehat{R})$ from the position $(q^0_{\mathcal{C}}, (q^0_{\mathcal{A}}, q^0_{\mathcal{C}}, s^0, h^0))$.

Consider a position $(q', (q, q', s, h))$ in the game $G(\mathcal{C}, \widehat{R})$ such that $\sigma((q, q'), s, h)$ is defined. Let $\sigma((q, q'), s, h) = ((q_1, q'_1, l', L'), s)$. The strategy is to go to the position $(q'_1, (q, q', s, h))$. Let us set $M = \delta_{\mathcal{A}}(q, (l, l'))$ and $M' = \delta_{\mathcal{C}}(q', (l', L'))$.

Suppose that there is an edge from $(q'_1, (q, q', s, h))$ to some $(q'_2, (q_3, q'_3, s_3))$. This means that $(a, q'_2) \in M'$ and $e_R((q, q', s, h), a) = (q_3, q'_3, s_3, h_3)$.

We analyze the definition of e_R case by case. We cannot have $a \in L$ because $(a, q'_2) \in M'$ and $\mathcal{C}$ is a cut automaton.

Suppose $\rightarrow_a \in M \cup M'$ and $(a, q'_3) \in M'$ and $(a, q_3) \in M$ and $e(s, a) = s_3$. Then $q'_2 = q'_3$ and we have a transition from $((q_1, q'_1, l', L'), s)$ to $((q_3, q'_3), s_3)$. This is a matching transition as $\sigma((q_3, q'_3), s_3, h_3)$ is defined because $h_3 = \mathit{hist}(\mathit{hist}(h, ((q, q'), s)), \sigma((q, q'), s))$.

The third case gives a transition from $(q'_1, (q, q', s))$ to $(q'_3, s^0_{q'_3})$, which is a winning position.

If $\rightarrow_a \in M'$ then $a \in \mathit{out}_R((q, q'))$ by the definition of the e_R relation.

If $\nrightarrow_a \in M'$ then $\rightarrow_a \notin M'$ and $\rightarrow_a \notin M$ by the consistency conditions. So $a \notin \mathit{out}_R((q, q', s))$. This finishes the proof of the claim.

Now we want to prove the converse implication of the theorem. Suppose we have $\widehat{R} \vDash \mathcal{C}$ and $P \times R \vDash \mathcal{A}$. We want to show that $P \vDash \mathcal{A}/\mathcal{C}$. This is implied by the following claim.

Claim 30.3 If $\widehat{R} \vDash \mathcal{C}$ and $P \times R \vDash \mathcal{A}$ then there is a winning strategy in $G(\mathcal{A}/\mathcal{C}, P)$ from $((q^0_{\mathcal{A}}, q^0_{\mathcal{C}}), s^0)$.

Let $\sigma_{\mathcal{A}}$ be the winning positional strategy in $G(\mathcal{A}, P \times R)$. As the first preparatory step we need to unwind R a little. We construct a process R_1 with states $(S_R \times Q_{\mathcal{A}} \times S_P) \cup S_R$. The initial state is $(s^0_R, q^0_{\mathcal{A}}, s^0_P)$ and the label of (s, q, s') is the label of s. From states in S_R the edges are exactly like in R. There is an edge on action a from (s', q, s) in R_1 if and only if there is s'_2 with $e_R(s', a) = s'_2$. The target of the edge depends on the set $M = \delta_{\mathcal{A}}(q_1, (\lambda(s), \lambda(s')))$ where q_1 is such that $\sigma_{\mathcal{A}}(q, (s, s')) = (q_1, (s, s'))$.

1. (s'_2, q_2, s_2), if $e_P(s, a) = s_2$ and $(a, q_2) \in M$,

2. (s'_2, q, s), if $\circlearrowleft_a \in M$,

3. s_2, otherwise.

In the first case, the edge is not a loop since $q_2 \neq q$. This follows from an assumption we have made about $\mathcal{A}$ at the beginning of the proof. In the second case the edge is a loop because $\circlearrowleft_a \in M = \delta_{\mathcal{A}}(q_1, (\lambda(s), \lambda(s')))$ so $a \in loop_P(s)$ and $a \in loop_R(s')$. Let us also remark that the first and the second conditions are mutually exclusive because we cannot have both (a, q_2) and $\circlearrowleft_a$ in $M = \delta_{\mathcal{A}}(q_1, (\lambda(s), \lambda(s')))$ as $\mathcal{A}$ is a nondeterministic automaton. Hence $loop_{R_1}(s', q, s) = \{a : \circlearrowleft_a \in M\}$.

By definition, R and R_1 have the same unwindings, so $\mathcal{C}$ accepts R_1 as $\mathcal{C}$ is a simple automaton. Let $\sigma_{\mathcal{C}}$ be a winning strategy in $G(\mathcal{C}, Cut(\widehat{R_1}))$.

We are now ready to describe the automaton strategy in $G(\mathcal{A}, P \times R)$. Suppose we are in a position $((q, q'), s)$ of $G(\mathcal{A}/\mathcal{C}, P)$ and moreover the strategies in positions $(q, (s, s'))$ and $(q', (s', q, s))$ are defined (these are positions in the games $G(\mathcal{A}, P \times R)$ and $G(\mathcal{C}, Cut(\widehat{R_1}))$ respectively). Let

$$(q_1, (s, s')) = \sigma_{\mathcal{A}}(q, (s, s')) \quad \text{and} \quad (q_1', (s', q, s)) = \sigma_{\mathcal{C}}(q', (s', q, s))$$

The strategy in $((q, q'), s)$ is to choose $((q_1, q_1', l', L'), s)$ such that :

$$l' = \lambda'(s'), \quad \text{and} \quad L' = \{\circlearrowleft_a : \circlearrowleft_a \in \delta_{\mathcal{A}}(q_1, (\lambda(s), l'))\}$$

Let us denote $M = \delta_{\mathcal{A}}(q, (l, l'))$ and $M' = \delta_{\mathcal{C}}(q', (l', L'))$.

To see that the above is a valid move it remains to check that if $\nrightarrow_a \in M'$ then $\rightarrow_a \notin M$. If $\nrightarrow_a \in M'$ then $a \notin out_R(s')$ as the position $(q_1', (s', q, s))$ is winning. But then in $P \times R$ there is no a edge from (s, s') so $\rightarrow_a \notin M$ as $(q_1, (s, s'))$ is winning. By the property of R_1 mentioned above we have $L' = loop_{R_1}(s', q, s)$. Hence $M' = \delta_{\mathcal{C}}(q_1', (l', L')) = \delta_{\mathcal{C}}(q_1', \lambda'(s', q, s))$.

Suppose now that we have an edge from $((q_1, q_1', l', L'), s)$ to some position $((q_2, q_2'), s_2)$. By the definition of $\mathcal{A}/\mathcal{C}$, this is because

$$\rightarrow_a \in M \cup M', \quad (a, q_2) \in M, \quad (a', q_2') \in M', \quad \text{and} \quad e_P(s, a) = s_2.$$

Because $\rightarrow_a \in M \cup M'$ we must have $a \in out_Q(s')$, say $e_Q(s', a) = s_2'$. If so, we have an edge from $(q_1, (s, s'))$ to $(q_2, (s_2, s_2'))$ in $G(\mathcal{A}, P \times R)$. From $(q_1', (s', q, s))$ we have an edge to $(q_2', (s_2', q_2, s_2))$ by the definition of R.

It remains to check that there is no edge to $\perp$ from $((q_1, q_1', l', L'), s)$. This follows from the fact that $(q_1, (s, s'))$ is a winning position. For example, if $\rightarrow_a \in \delta_/((M, M', l', L'), \lambda(s))$ then $\rightarrow_a \in d_{\mathcal{A}}(q_1, (\lambda(s), l'))$ and P must have an a transition since $P \times R$ has one. Similarly for $\nrightarrow_a$, $\circlearrowleft_a$ and $\overline{\circlearrowleft}_a$. $\square$

Theorem 31
Let $\mathcal{A}$ be a loop automaton over the set of labels $\Lambda \times \Lambda'$ and let $\mathcal{B}$ be a simple automaton over Λ'. There exists an automaton, denoted $\mathcal{A}/\mathcal{B}$, over Λ such that for every process P over Λ:

$P \models \mathcal{A}/\mathcal{B}$ if and only if there is a process R over Λ' such that $R \models \mathcal{B}$ and $P \times R \models \mathcal{A}$.

Proof
By Theorem 23 and Proposition 18 we may assume without loss of generality that $\mathcal{A}$ is a bipartite nondeterministic parity loop automaton.

By Lemma 14, there exists a bipartite nondeterministic parity cut automaton $\mathcal{C}$ equivalent to $Cut(\widehat{\mathcal{B}})$. Then by Fact 16 and Proposition 20, $\widehat{R} \models \mathcal{C}$ if and only if $\widehat{R} \models \widehat{\mathcal{B}}$ if and only if $R \models \mathcal{B}$.

We define $\mathcal{A}/\mathcal{B}$ as the automaton $\mathcal{A}/\mathcal{C}$ and use Proposition 30. $\square$

4 Synthesis of controllers

4.1 Product of automata

If $\mathcal{A}$ and $\mathcal{B}$ are automata over the same alphabet, then it is easy to construct an automaton $\mathcal{A} \times \mathcal{B}$ such that for every P: $P \vDash \mathcal{A} \times \mathcal{B}$ if and only if $P \vDash \mathcal{A}$ and $P \vDash \mathcal{B}$. It is enough to take the disjoint union of $\mathcal{A}$ and $\mathcal{B}$, to add a new universal state q^0, and to define $\delta(q^0, l) = \{(\varepsilon, q^0_{\mathcal{A}}), (\varepsilon, q^0_{\mathcal{B}})\}$. Then a process is accepted from q^0 if and only if it is accepted by $\mathcal{A}$ and $\mathcal{B}$.

In case $\mathcal{A}$ and $\mathcal{B}$ are nondeterministic, this construction is equivalent to the following one (assuming that the set of states of these automata are disjoint). The universal states of the product are $Q^{\forall}_{\mathcal{A}} \cup Q^{\forall}_{\mathcal{B}} \cup (Q^{\forall}_{\mathcal{A}} \times Q^{\forall}_{\mathcal{B}})$, its existential states are $Q^{\exists}_{\mathcal{A}} \cup Q^{\exists}_{\mathcal{B}} \cup (Q^{\exists}_{\mathcal{A}} \times Q^{\exists}_{\mathcal{B}})$, the initial state is $(q^0_{\mathcal{A}}, q^0_{\mathcal{B}})$. The accepting set is the set of all sequences whose projections over $Q_{\mathcal{A}}$ and $Q_{\mathcal{B}}$ are finite or are respectively in $Acc_{\mathcal{A}}$ and $Acc_{\mathcal{B}}$.

If (q, q') is existential, then

$$\delta((q, q'), l) = \{(\varepsilon, (q_1, q'_1)) : (\varepsilon, q_1) \in \delta_{\mathcal{A}}(q, l), (\varepsilon, q'_1) \in \delta_{\mathcal{B}}(q', l)\}.$$

If (q, q') is universal, then $\delta((q, q'), l) =$
$(\delta_{\mathcal{A}}(q, l) \cup \delta_{\mathcal{B}}(q', l)) \cap (A \times \{\to, \not\to, \circlearrowleft, \overline{\circlearrowleft}\})$
$\cup\{(a, (q_1, q'_1)) : a \in A, (a, q_1) \in \delta_{\mathcal{A}}(q, l), (a, q'_1) \in \delta_{\mathcal{B}}(q', l)\}$
$\cup\{(a, q_1) : a \in A, (a, q_1) \in \delta_{\mathcal{A}}(q, l), (\{a\} \times Q_{\mathcal{B}}) \cap \delta_{\mathcal{B}}(q', l) = \emptyset\}$
$\cup\{(a, q'_1) : a \in A, (\{a\} \times Q_{\mathcal{A}}) \cap \delta_{\mathcal{A}}(q', l) = \emptyset,\ (a, q'_1) \in \delta_{\mathcal{B}}(q', l)\}$.

However this product may not be nondeterministic. It is the case when $\circlearrowleft_a \in \delta_{\mathcal{A}}(q, l)$ and $(a, q'_1) \in \delta_{\mathcal{B}}(q, l)$.

4.2 Centralized control problems

For a fixed set A of actions and two label alphabets Λ, Λ' the centralized control problem is the following:

> For a given process P over Λ, and two automata $\mathcal{A}$ and $\mathcal{B}$ over $\Lambda \times \Lambda'$ and Λ', respectively, find R over Λ', such that:
>
> $$P \times R \models \mathcal{A} \text{ and } R \models \mathcal{B}.$$

Let $Sol(P, \mathcal{A}, \mathcal{B})$ be the set of all R such that $P \times R \models \mathcal{A}$ and $R \models \mathcal{B}$.

Using Theorem 27, the following result is obvious.

Corollary 32 $R \in Sol(P, \mathcal{A}, \mathcal{B})$ if and only if $R \models (\mathcal{A}/P) \times \mathcal{B}$.

4.3 Decentralized control problems

We study the following form of decentralized control problem. Given a process P over A and Λ, an automaton $\mathcal{A}$ over A and $\Lambda \times \Lambda_0 \times \cdots \times \Lambda_n$ and $n+1$ automata $\mathcal{B}_i$ over A and Λ_i (for $i = 0, \ldots, n$), we have to find $n+1$ processes R_i over A and Λ_i such that $P \times R_0 \times \cdots \times R_n \models \mathcal{A}$ and $R_i \models \mathcal{B}_i$ for $i = 0, \ldots, n$.

Let Sol be the set of all $(n+1)$-tuples of processes $(R_0, \ldots, R_n)$ such that $P \times R_0 \times \cdots \times R_n \models \mathcal{A}$ and $R_i \models \mathcal{B}_i$ for $i = 0, \ldots, n$.

In the next subsection we will show that the emptiness of Sol is decidable when at most one of $\mathcal{B}_i$ is a loop automaton (and the other $\mathcal{B}_i$' automata are simple automata). We then show that the emptiness of Sol is undecidable if two of the $\mathcal{B}_i$'s are allowed to be loop automata. In the last subsection we point out that the borderline between decidability and undecidability is not that obvious. We introduce a notion of deterministic automata, and show that for some specifications the emptiness of Sol is decidable if all $\mathcal{B}_i$ are determinisitic loop automata. Actually this last case corresponds to one of the classic formulations of decentralized synthesis problems [15].

4.3.1 Decidable control problems

In this section, we assume that $\mathcal{B}_0$ is a loop automaton and $\mathcal{B}_1, \ldots, \mathcal{B}_n$ are simple automata.

For $i = 0, \ldots, n$, let Sol_i be the set of all $(i+1)$-tuples $(R_0, \ldots, R_i)$ such that there exists a $(n+1)$-tuple $(R_0, \ldots, R_i, R_{i+1}, \ldots, R_n) \in Sol$. In particular $Sol_n = Sol$.

For $i = 0, \ldots, n$, let $Sol'_i = \{(R_0, \ldots, R_i) : R_0 \models \mathcal{B}_0, \ldots, R_{i-1} \models \mathcal{B}_{i-1}, R_i \models \mathcal{B}_i \times (\mathcal{A}/\mathcal{B}_n/\cdots/\mathcal{B}_{i+1}/(P \times R_0 \times \cdots \times R_{i-1}))\}$.

By definition, $Sol'_0 = \{R_0 : R_0 \models \mathcal{B}_0 \times (\mathcal{A}/\mathcal{B}_n/\cdots/\mathcal{B}_1/P)\}$.

Proposition 33 For $i = 0, \ldots, n$, $Sol_i = Sol'_i$.

Proof
Using Theorems 31 and 27, it is easy to show that $(R_1, \ldots, R_i) \in Sol'_i$ if and only if there exists R_{i+1} such that $(R_1, \ldots, R_i, R_{i+1}) \in Sol'_{i+1}$. By Theorem 31, $Sol'_n = Sol$, and the proposition is proved by induction. $\square$

Corollary 34 *Sol* is not empty if and only if $\mathcal{B}_0 \times (\mathcal{A}/\mathcal{B}_n/\cdots/\mathcal{B}_1/P)$ has a model.

All the elements of *Sol* can be found by the following algorithm. Find a model R_0 of $\mathcal{B}_0 \times (\mathcal{A}/\mathcal{B}_n/\cdots/\mathcal{B}_1/P)$. For $i = 1, \ldots n$ find a model Q_i of $\mathcal{B}_i \times (\mathcal{A}/\mathcal{B}_n/\cdots/\mathcal{B}_{i+1}/(P \times R_0 \times \cdots \times R_{i-1}))$.

4.3.2 An undecidable control problem

Here we show that the decentralized control problem becomes undecidable if at least two of the $\mathcal{B}_i$'s are loop automata.

First let us remark that, because of Theorem 27, any decentralized control problem

$$P \times R_0 \times \cdots \times R_n \models \mathcal{A} \quad \text{and} \quad R_i \models \mathcal{B}_i$$

can be put in the form $R_0 \times \cdots \times R_n \models \mathcal{A}/P$ and $R_i \models \mathcal{B}_i$. Moreover, there is a one-state process 1_A over the alphabet A such that $\mathcal{A}/1_A = \mathcal{A}$. So, the family of problems stated in the following theorem is indeed a family of decentralized control problems.

Theorem 35
It is undecidable to check whether for given $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{B}'$ there exist P and P' such that: $P \models \mathcal{B}$, $P' \models \mathcal{B}'$, and $P \times P' \models \mathcal{A}$. The problem is undecidable even if $\mathcal{A}$ is required to be simple.

Proof
We show below that the Post correspondence problem can be effectively reduced to a control problem of this kind.

Let $\{(u_i, v_i) : i = 1, \ldots, n\}$ be a Post system over an alphabet A. Let B be the alphabet $\{\$_i : i = 1, \ldots, n\}$. Define

$$L_1 = \{\$_i u_i : i = 1, \ldots n\}^* \quad \text{and} \quad L_2 = \{\$_i v_i : i = 1, \ldots n\}^*$$

It is obvious that the associated correspondence problem has a solution if and only if there are words $x \in A^*$ and $y \in B^*$ such that $x \sqcup\!\sqcup y \cap L_1 \neq \emptyset$ and $x \sqcup\!\sqcup y \cap L_2 \neq \emptyset$. Here $x \sqcup\!\sqcup y$, the shuffle of x and y is the set of all words w in $(A \cup B)^*$ such that $\pi_A(w) = x$ and $\pi_B(x) = y$.

Let # be a new symbol. It is easy to define two nondeterministic simple automata $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $P \models \mathcal{A}_i$ if and only if P contains a path (starting in the initial state) labelled by a word in $L_i\#$.

As a last preparatory step for the proof consider for each word $x \in A^*$ a process $P^A_x = \langle A \cup B, \{l\}, S_x, s^0_x, e_x, \lambda \rangle$ over the set of actions $A \cup B$ and an irrelevant set of labels. We define: $S_x = \{1, \ldots, |x| + 2\}$; $s^0_x = 1$; $\lambda(i) = l$ for all $i \in S$; and

$$e_x(i,d) = \begin{cases} i+1 & \text{if } d = x_i \text{ or } (d = \# \text{ and } i = n+1) \\ i & \text{if } d \in B \\ \text{undefined} & \text{otherwise} \end{cases}$$

Similarly we define P^B_y for $y \in B^*$.

Lemma 36 The set of paths of the form $u\#$ in $P^A_x \times P^B_y$ is exactly $(x \sqcup\!\sqcup y)\#$. So $P^A_x \times P^B_y \vDash \mathcal{A}_1 \times \mathcal{A}_2$ if and only if $x \in A^*$ and $y \in B^*$ describe a solution to the Post correspondence problem.

It is not difficult to construct loop automata $\mathcal{C}^A$ and $\mathcal{C}^B$ accepting the structures of the form P^A_x and P^B_y respectively. Below we show how to construct $\mathcal{C}^A$, the construction of $\mathcal{C}^B$ is analogous. The existential states of $\mathcal{C}^A$ are $\{q^0, q^1\}$ and the universal states are $\{q_\emptyset\} \cup \{q_a : a \in A \cup \{\#\}\}$. The acceptance condition of the automaton is $\emptyset$, so it can only accept finite words. The transition function is defined by:

$$\begin{aligned} \delta(q^0, l) &= \{(\varepsilon, q_a) : a \in A \cup \{\#\}\} \\ \delta(q_a, l) &= \{(a, q^0)\} \cup \{\not\rightarrow_d : d \in A \cup \{\#\}, d \neq a\} \cup \{\circlearrowleft_b : b \in B\} \quad \text{if } a \neq \# \\ \delta(q_\#, l) &= \{(\#, q^1)\} \cup \{\not\rightarrow_d : d \in A\} \cup \{\circlearrowleft_b : b \in B\} \\ \delta(q^1, l) &= \{(\varepsilon, q_\emptyset)\} \\ \delta(q_\emptyset, l) &= \{\not\rightarrow_d : d \in A \cup B \cup \{\#\}\} \end{aligned}$$

Lemma 37 For every process P: $P \vDash \mathcal{C}^A$ if and only if it is isomorphic to P^A_x for some $x \in A^*$. Similarly for $\mathcal{C}^B$.

It follows that there exist P and P' such that $P \models \mathcal{C}^A$, $P' \models \mathcal{C}^B$, $P \times P' \models \mathcal{A}_1 \times \mathcal{A}_2$ if and only if the Post correspondence problem has a solution. $\square$

4.3.3 Some other cases

Although a decentralized control problem is in general undecidable if two of the controllers are specified by loop automata, there are interesting decidable subcases. For instance, one of the most well known versions of decentralized control problem [15] is decidable. We show how to solve it in our framework.

For a given process P over the set of actions A, the *language of* P is the set $L(P) \subseteq A^*$ of all words w such that w is a valid sequence of actions of P from the initial state.

The statement of the problem is as follows. We are given a plant P over an alphabet A and a singleton label set Λ, a language $K \subseteq L(P)$ and alphabets $A_c^1, A_o^1, A_c^2, A_o^2 \subseteq \Sigma$. The goal is to find (if they exist) controllers R_1 and R_2 such that:

$$L(P \times R_1 \times R_2) = K$$

and moreover for $i = 1, 2$ the controller R_i can control only actions from A_c^i and can observe only actions from A_o^i.

To solve this problem we consider two nondeterministic parity automata $\mathcal{A}_K$ and $\mathcal{A}_{\supseteq K}$ such that $R \vDash \mathcal{A}_K$ if and only if $L(R) = K$ and $R \vDash \mathcal{A}_{\supseteq K}$ if and only if $L(R) \supseteq K$. It is not difficult to construct these two automata from the usual word automaton recognizing K. Moreover, these automata are *deterministic*, i.e., for every existential state, $\delta(q)$ contains at most one element (ε, q').

Consider an automaton $\mathcal{B}_1$ such that $R \vDash \mathcal{B}_1$ if and only if every state of R has an edge on actions in $A \setminus A_c^1$ and has a loop on actions in $A \setminus A_o^1$. In other words $\mathcal{B}_1$ says that R is a controller that can control only actions in A_c^1 and can see only actions in A_o^1. Similarly we define $\mathcal{B}_2$. These automata are also parity deterministic automata.

Therefore our problem can now be stated as follows. We have to find R_1 and R_2 such that $R_1 \times R_2 \times P \models \mathcal{A}_K$ and $R_i \models \mathcal{B}_i$. But if $R_1 \times R_2 \times P \models \mathcal{A}_K$ then $L(R_i) \supseteq K$ hence $R_i \models \mathcal{A}_{\supseteq K}$. So, the problem can be reformulated as: find R_1 and R_2 such that $R_1 \times R_2 \times P \models \mathcal{A}_K$ and $R_i \models \mathcal{A}_{\supseteq K} \times \mathcal{B}_i$.

Let $\mathcal{C}_i$ be a nondeterministic loop automaton equivalent to $\mathcal{A}_{\supseteq K} \times \mathcal{B}_i$. It is easy to see that this automaton is still a deterministic parity automaton. For deterministic automata the following holds.

Fact 38 Let $\mathcal{C}$ be a determnistic automaton over a singleton label set. If $\mathcal{C}$ has a model then it has a smallest model, i.e., a model R^{min} such that $R^{min} \vDash \mathcal{C}$ and for every other model $R \vDash \mathcal{C}$ we have $L(R^{min}) \subseteq L(R)$.

The proof of this fact follows from the examination of the proof of Theorem 6. All the models of $\mathcal{C}$ can be obtained from winning strategies in $G(\mathcal{C})$. In general a strategy from an existential state q has to choose $l \in \Lambda$, $(\varepsilon, q_1) \in \delta(q,l)$ and $L \in A$ such that A contains $\{a :\to_a \in \delta(q_1,l)\}$ and has no a from $\{a :\not\to_a \in \delta(q_1,l)\}$. When $\mathcal{C}$ is deterministic and Λ is a singleton, the choice of l and q_1 is determined. So the strategy can only choose between different L. It is then easy to see that the best strategy to win the game is to choose the smallest possible L. This strategy determines R^{min}.

Now, if (R_1, R_2) is a solution to our problem then R_1^{min} and R_2^{min} exist. It follows that:

$$K = K \cap L(P) \subseteq L(R_1^{min}) \cap L(R_2^{min}) \cap L(P) \subseteq L(R_1) \cap L(R_2) \cap L(P) = K$$

hence (R_1^{min}, R_2^{min}) is also a solution.

Therefore, the method to solve the problem is to compute (R_1^{min}, R_2^{min}) and to check whether $R_1^{min} \times R_2^{min} \times P \models \mathcal{A}_K$.

5 Complexity issues

Here we will just very shortly discuss the complexity of finding a solution of a control problem. We do not intend to precisely analyze this complexity because there is a multitude of possible parameters that can vary. For example one can assume that some automata in question are nondeterministic or even deterministic of some special form, as it is the case in the previous section. It is not clear to us at the present moment which cases are interesting and which are not. So the goal of this section is just to say that our constructions have reasonable complexity in most obvious cases.

For an automaton $\mathcal{A}$, its size, denoted $|\mathcal{A}|$, is the number of its states. If moreover $\mathcal{A}$ is a parity automaton, its index is the maximal rank of its states.

First of all, the basic construction needed to solve a control problem is to find a winning strategy in game $G(\mathcal{A})$ where $\mathcal{A}$ is a nondeterministic automaton. By Theorem 6 finding such a strategy is equivalent to checking whether $\mathcal{A}$ has a model. The most interesting case for us here is the case when $\mathcal{A}$ is a parity automaton. In this case the exact complexity of the problem is not yet known. It is in NP$\cap$co-NP. So far, the best upper bound is $\mathcal{O}(n^{(k/2)+1})$ where n is the size of the automaton and k is its index [1, 17, 21].

The other basic result we want to recall concerns translation from alternating to nondeterministic automata. If $\mathcal{A}$ is a parity automaton of size n then there is a parity nondeterministic automaton equivalent to $\mathcal{A}$ of size $2^{\mathcal{O}(n\log(n))}$, and of index $\mathcal{O}(n)$ [5, 18].

The automaton $\mathcal{A}/P$ is of size $|\mathcal{A}||P|$. The construction works for alternating as well as for nondeterministic loop automata $\mathcal{A}$. If $\mathcal{A}$ is nondeterministic then $\mathcal{A}/P$ is nondeterministic. If $\mathcal{A}$ is a parity automaton then so is $\mathcal{A}/P$.

If $\mathcal{A}$ and $\mathcal{B}$ are both parity alternating automata then $\mathcal{A}/P \times \mathcal{B}$ is an alternating parity automaton of size $n = |\mathcal{A}||P| + |\mathcal{B}|$. Then there is a nondeterministic automaton $\mathcal{C}$ equivalent to $\mathcal{A}/P \times \mathcal{B}$ of size $2^{\mathcal{O}(n \log(n))}$ and index $\mathcal{O}(n)$. But a more careful look at the construction allows to see that this size does not depend exponentially on P. This is because when we transform $\mathcal{A}/P \times \mathcal{B}$ into a nondeterministic automaton, the states of this nondeterministic automaton are sets of triples (q, s, q') of states from $\mathcal{A}$, P and $\mathcal{B}$ respectively. It can be checked that because of the special form of the transition function of $\mathcal{A}/P$, in the resulting nondeterministic automaton all reachable states are sets of triples with the same second component. So the size of $\mathcal{C}$ is $|P| \times 2^{\mathcal{O}(m \log(m))}$ where $m = |\mathcal{A}| + |\mathcal{B}|$.

Next, let us look at the complexity of the construction of $\mathcal{A}/\mathcal{B}$. If $\mathcal{B}$ is a simple automaton, then $Cut(\widehat{\mathcal{B}})$ is equivalent to a nondeterministic automaton of size $m' = 2^{\mathcal{O}(m \log(m))}$ with $m = \mathcal{O}(|\mathcal{B}|)$. Thus $\mathcal{A}/\mathcal{B}$ is of size $\mathcal{O}(|\mathcal{A}|m')$. The automaton $\mathcal{A}/\mathcal{B}$ is a nondeterministic Rabin automaton but not a parity automaton. Still, it can be transformed to a parity automaton of the same size because the Rabin condition is small.

The analysis presented above is far from being complete or detailed enough. For example, it can be shown that our solution to the diagnosability problem discussed in Section 2.5 is implementable in PTIME. The fact that diagnosibility is in PTIME seems to have been noticed only recently [7, 22]. In these papers a more general diagnosability problem is considered in the sense that there are several different kinds of failures. Actually an obvious extension of our solution to this general case would still work in PTIME.

Acknowledgments The starting point of this work was a stimulating discussion on the very nature of a control problem, in the final meeting of the Action coopérative INRIA *MARS* on "Control synthesis by Petri nets".

References

[1] A. Arnold and D. Niwiński. *Rudiments of μ-calculus*, volume 146 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 2001.

[2] A. Bergeron. A unified approach to control problems in discrete event processes. *RAIRO-ITA*, 27:555–573, 1993.

[3] J. R. Büchi. State strategies for games in $F_{\sigma\delta} \cap G_{\delta\sigma}$. *Journal of Symbolic Logic*, 48:1171–1198, 1983.

[4] C. G. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Kluwer Academic Pub., 1999.

[5] E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs. In *29th FOCS*, 1988.

[6] Y. Gurevich and L. Harrington. Trees, automata and games. In *14th ACM Symp. on Theory of Computations*, pages 60–65, 1982.

[7] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial time algorithm for diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 46(8), 2001.

[8] R. Kumar and V. K. Garg. *Modeling and control of logical discrete event systems*. Kluwer Academic Pub., 1995.

[9] O. Kupferman and M. Vardi. Synthesis with incomplete information. In *2nd International Conference on Temporal Logic*, pages 91–106, 1997.

[10] O. Kupferman and M. Vardi. μ-calculus synthesis. In *MFCS 2000*, pages 497–507. LNCS 1893, 2000.

[11] H. Lamouchi and J. G. Thistle. Effective control synthesis for DES under partial observations. In *Proc. 39th IEEE Conf. on Decision and Control*, December 2000.

[12] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In E.W. Mayr and C. Puech, editors, *STACS'95*, volume 900 of *LNCS*, pages 229–242, 1995.

[13] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *31th IEEE Symposium Foundations of Computer Science (FOCS 1990)*, pages 746–757, 1990.

[14] P. J. G. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77, 1989.

[15] K. Rudie and W. Wonham. Think globally, act locally. *IEEE Transactions on Automatic Control*, 37(11):1692–1708, 1992.

[16] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

[17] H. Seidl. Fast and simple nested fixpoints. *Information Processing Letters*, 59:303–308, 1996.

[18] W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3. Springer-Verlag, 1997.

[19] S Tripakis. Undecidable problems of decentralized observation and control. In *IEEE Conference on Decision and Control*, 2001.

[20] A. Vincent. Synthèse de contrôleurs et stratégies gagnantes dans les jeux de parité. In *to appear in MSR2001*, 2001.

[21] J. Vöge and M. Jurdzinski. A discrete strategy improvement algorithm for solving parity games. In *CAV 2000*, pages 202–215, 2000.

[22] T. Yoo and S. Lafortune. On the computataional complexity of some problems arising in partially-observed discrete-event systems. In *American Control Conference 2001*, 2001.