PUBLICATION INTERNE
N° 1648

THE CONTROL OF NONDETERMINISTIC SYSTEMS : A LOGICAL APPROACH

SOPHIE PINCHINAT AND JEAN-BAPTISTE RACLET

ISSN 1166-8687

The control of nondeterministic systems : a logical approach

Sophie Pinchinat [*] and Jean-Baptiste Raclet [**]

Systèmes communicants
Projet S4

Publication interne n° 1648 — Octobre 2004 — 23 pages

Abstract: We answer a wide range of control problems for nondeterministic discrete-event systems, relying on recent works based on a second order logic approach for deterministic systems. We investigate a pair of transformations: the first transforms a nondeterministic system into a deterministic one with a new unobservable event; the second transforms logical statements. In particular, these transformations are used to reduce control problems for nondeterministic systems to control problems under partial observation for deterministic systems.

Key-words: discrete-event systems, nondeterminism, mu-calculus, model-checking, controller synthesis.

(Résumé : tsvp)

[*] Sophie.Pinchinat@irisa.fr
[**] Jean-Baptiste.Raclet@irisa.fr

Une approche logique au contrôle des systèmes non-déterministes

Résumé : Nous abordons une large famille de problèmes de contrôle pour les systèmes à événements discrets non-déterministes en se basant sur les travaux récents qui utilisent la logique du second ordre pour traiter les systèmes déterministes. Nous proposons deux transformations : la première transforme un système non-déterministe en un système déterministe comprenant un nouvel événement inobservable; la seconde adapte les énoncés logiques. En particulier, ces transformations sont utilisées pour réduire un problème de contrôle d'un système non-déterministe en un problème de contrôle d'un système déterministe.

Mots clés : système à événements discrets, non-déterminisme, mu-calcul, model-checking, synthèse de contrôleur.

1 Introduction

Nondeterministic systems are usually understood as systems which have alternative reactions to the same external stimulus. Although it might be argued that nondeterminism does not have a realistic meaning, it occurs by abstracting concrete systems from unwanted informations : one can for example abstract from the value of messages along a communication channel. Henceforth, formal methods such as controller synthesis deserve being investigated for these models, as an intermediate step in the design of real systems.

In this paper, we show a polynomial reduction from the model-checking of a second order logic on nondeterministic systems to the model-checking of similar but pore expressive logic on some derived deterministic system. Both logics originate from [RP03b] and [RP03a], they are adequate to specify controllers for Mu-calculus definable objectives. Decision issues for both checking the existence of controllers and synthesizing them (when possible) are already made clear by [RP03a] for deterministic systems. We show here how the method can be adapted to nondeterministic systems according to the following : given a controller specification α for a nondeterministic system $\mathcal{S}$, we can transform both the nondeterministic system $\mathcal{S}$ and the specification formula α respectively into a deterministic system $\mathcal{S}'$ with an unobservable event τ and into a formula α' which specifies a controller $\mathcal{C}'$ under partial observation for the deterministic system $\mathcal{S}'$. It is proved that the two problems are equivalent in the sense that there exists a controller $\mathcal{C}$ of $\mathcal{S}$ if and only if there exists a controller $\mathcal{C}'$ for $\mathcal{S}'$. Moreover, the two problems have the same complexity, as well as the synthesis procedure since the controller $\mathcal{C}$ is simply $\mathcal{C}'$ where the τ transition are removed.

This approach seems to be very closed to the pioneer work of [HL98] who used a similar method. However, the models we propose are a lot more general. Firstly, our semantics is finer : instead of considering trajectory-model specifications, we use transition systems modulo bisimulation. Secondly, with this finer semantics, we can consider Mu-calculus definable properties - which would not be adequate for trajectory-model specifications. We recall that the Mu-calculus is the most expressive formalism to handle branching-time specifications : it subsumes regular languages, omega-regular languages, but also temporal logics like CTL, LTL, CTL* (see [Eme90] for a survey).

As the work of [RP03a] permits the synthesis of controller under partial observation for deterministic systems, we derive for free a model-checking (and a synthesis) method for the nondeterministic world. Notice that the controllers we synthesize are required deterministic, which perfectly fits the intuition we have of nondeterministic systems : nondeterminism arises as an abstraction where some events should not be distinguished, even for applying a control. Hence the solution should respect this perception.

The work is organized as follows : Section 2 presents the models; Section 3 describes the transformation on nondeterministic systems in order to have deterministic ones; Section 4 introduces the second order logical formalisms, which are applied in Section 5 for the control of nondeterministic systems. We give a short conclusion in Section 6.

2 Preliminary Definitions

All through this paper, we let $\Sigma = \{a, b, c...\}$ be a finite set of events and $AP = \{p, q, ...\}$ be a finite set of atomic propositions.

Definition 1 (Process)
A process on Γ is a tuple $\mathcal{S} = \langle \Gamma, S, s^0, t, L\rangle$ where

- $\Gamma \subseteq AP$,
- *S is a finite set of states,*
- *$s^0 \in S$ is the initial state,*
- *$t : S \times \Sigma \rightarrow 2^S$ is the transition relation,*
- *$L : S \rightarrow 2^\Gamma$ labels states by atomic propositions.*

Given s and $a \in \Sigma$ we freely write $s' \in t(s, a)$ or $(s, a, s') \in t$ or $s \xrightarrow{a} s'$, in which cases s' is called a a-successor of s or a successor of s by a.

Moreover, a process $\mathcal{S}$ is *finite* if S is finite, $\mathcal{S}$ is *complete* if for all pair $(s, a) \in S \times \Sigma$, there exists $s' \in S$ such that $s \xrightarrow{a} s'$. Finally $\mathcal{S}$ is *deterministic* if $|t(s, a)| \leq 1$ for all $(s, a) \in S \times \Sigma$.

Processes are combined using the *synchronous product*, as to define a control by the product of a plant with its controller :

Definition 2 (Synchronous product)
Let $\mathcal{S}_1 = \langle \Gamma_1, S_1, s_1^0, t_1, L_1\rangle$ and $\mathcal{S}_2 = \langle \Gamma_2, S_2, s_2^0, t_2, L_2\rangle$ be two processes with disjoint Γ_1 and Γ_2. Their synchronous product *is $\mathcal{S}_1 \times \mathcal{S}_2 = \langle \Gamma, S_1 \times S_2, (s_1^0, s_2^0), t, L\rangle$ over $\Gamma = \Gamma_1 \cup \Gamma_2$ where*

- *$(s_1, s_2) \xrightarrow{a} (s_1', s_2')$ if $s_1 \xrightarrow{a}_1 s_1'$ and $s_2 \xrightarrow{a}_2 s_2'$;*
- *$L(s_1, s_2) = L_1(s_1) \cup L_2(s_2)$.*

Since the processes are nondeterministic in general, we now explain a procedure to encode such nondeterministic objects into deterministic ones. Notice that we cannot use a standard determinisation algorithm as in language theory since we aim at preserving the branching-time properties of the behaviors. To motivate this observation, let us consider the following machines (example in Figure 1) : the machines can serve either coffee or tea.

However, when a customer introduces coins in a machine, and according to her choice, the left handside machine can either offer coffee or tea, whereas the right handside machine would chose by itself if coffee or tea would be delivered. Hence the behaviors are different because of the point of choice in the execution. The branching-time feature of behaviors is provided by the classic notion of *bisimulation* : a bisimulation is an equivalence relation between processes which stands for "having the same behavior" and would distinguish the two coffee machines above.

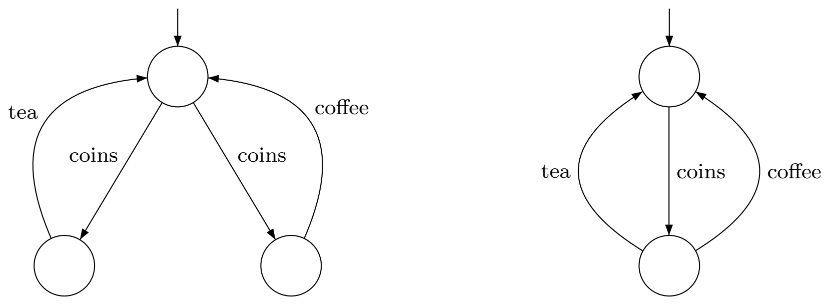


Figure 1: Two different "coffee" machines.

Definition 3 (Bisimulation)
Let $\mathcal{S}_1 = \langle \Gamma, S_1, s_1^0, t_1, L_1 \rangle$ and $\mathcal{S}_2 = \langle \Gamma, S_2, s_2^0, t_2, L_2 \rangle$ be two processes. A binary relation $\mathcal{R} \subseteq S_1 \times S_2$ is a bisimulation *between $\mathcal{S}_1$ and $\mathcal{S}_2$ if :*

1. *$\mathcal{R}$ is total;*
2. *$\mathcal{R}$ relates the initial states : $(s_1^0, s_2^0) \in \mathcal{R}$, and*
3. *for all $s_1 \in S_1$ and for all $s_2 \in S_2$, $(s_1, s_2) \in \mathcal{R}$ implies : $L_1(s_1) = L_2(s_2)$, and*
   - *$\forall s_1 \xrightarrow{a}_1 s_1'$, $\exists s_2' \in S_2$ s.t. $s_2 \xrightarrow{a}_2 s_2'$ and $(s_1', s_2') \in \mathcal{R}$;*
   - *conversely, $\forall s_2 \xrightarrow{a}_2 s_2'$, $\exists s_1' \in S_1$ s.t. $s_1 \xrightarrow{a}_1 s_1'$ and $(s_1', s_2') \in \mathcal{R}$.*

We write $\mathcal{R} : \mathcal{S}_1 \underline{\leftrightarrow} \mathcal{S}_2$ whenever $\mathcal{R}$ is a bisimulation between $\mathcal{S}_1$ et $\mathcal{S}_2$, and $\mathcal{S}_1 \underline{\leftrightarrow} \mathcal{S}_2$ whenever there exists a bisimulation between $\mathcal{S}_1$ et $\mathcal{S}_2$.

We assume the reader is familiar with the notion *execution tree* of a process : it is obtained by unfolding (possibly ad infinitum) the process. Given a process $\mathcal{S}$ we denote by $T_{\mathcal{S}}$ its execution tree. We always have $\mathcal{S} \underline{\leftrightarrow} T_{\mathcal{S}}$.

3 Encoding the Nondeterministic Processes by Deterministic Processes

As announced, we propose now an encoding of nondeterministic processes; this encoding is inspired from [Tho97]. We then establish the mathematical properties of this encoding. Basically, a new event called τ is considered. Now if we observe a nondeterminism on a in a node of the tree like in state s_1 in Figure 2, we designate a a-transition to be kept (transition $s_1 \xrightarrow{a} s_1'$ in the example) while the others are removed and the pending a-successors (namely s_2', s_3') are traversed by the addition of τ-transitions between them (in dashed arrows in the figure).

This principle is totally clear when applied on execution trees and as expected produces a deterministic object. However, for this encoding to be effective, we would rather performed

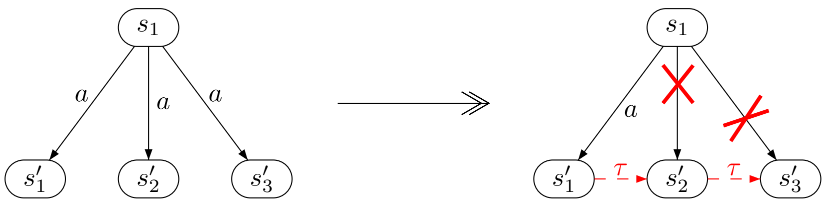


Figure 2: Encoding of the execution trees

it on the process itself. However, a naive approach as for trees, does not work. Figure 3 gives an example where the procedure applied on the process generates a nondeterminism in τ in the encoded process :

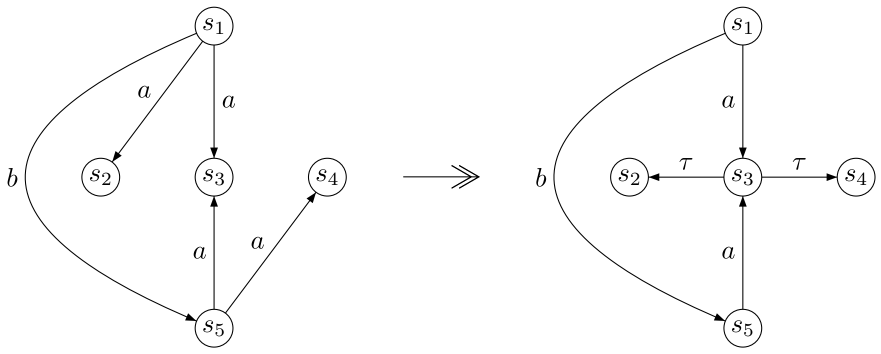


Figure 3: Generation of nondeterminism in τ.

Alternatively, we propose two successive transformations that need being applied in general to get a correct encoding. Intuitively, given a nondeterministic process $\mathcal{S}$, we chose a total order $\leq$ on its set of states and apply two transformations T_{ip} and $\mathcal{T}_{\leq}$, to obtain a correct deterministic process $\mathcal{S}^{\tau}$, as in bisimulation with the encoded execution tree $T^{\tau}_{\mathcal{S},\leq}$.

Remarks : we can notice that if we choose to keep the transitions $s_1 \xrightarrow{a} s_2$ and $s_5 \xrightarrow{a} s_4$ and to cut the transitions $s_1 \xrightarrow{a} s_3$ and $s_5 \xrightarrow{a} s_3$ in Figure 3, there is no generation of τ-nondeterminism. A more complicated example such that a *"good"* way to cut transitions doesn't exist can be found in Appendix A.

3.1 The Transformation T_{ip}

Transformation T_{ip} ("ip" for immediate past) consists in storing in the current state of an execution the previous state and the last event. To obtain it, we have to unfold one level of the process :

Definition 4 (Transformation T_{ip})
Let $\mathcal{S} = \langle \Gamma, S, s^0, t, L\rangle$ be a nondeterministic process. The process $T_{ip}(\mathcal{S})$ is $T_{ip}(\mathcal{S}) = \langle \Gamma, \widetilde{S}, \widetilde{s^0}, \widetilde{t}, \widetilde{L}\rangle$ with the set of events Σ and where :

- $\widetilde{S} = \{s^0\} \cup \{s'_{(s,a)} \mid s \xrightarrow{a} s'\}$, $\widetilde{s^0} = s^0$;
- *if* $s \xrightarrow{b} s'$ *then* $s_{(r,a)} \xrightarrow{b} s'_{(s,b)}$ *for all* $r \in S$ *and for all* $a \in \Sigma$ *such that* $s_{(r,a)} \in \widetilde{S}$
  if moreover $s = \widetilde{s^0}$ *then* $\widetilde{s^0} \xrightarrow{b} s'_{(\widetilde{s^0},b)}$;
- $\widetilde{L}(\widetilde{s^0}) = L(s^0)$ *and* $\widetilde{L}(s'_{(s,a)}) = L(s')$.

The set of events of $T_{ip}(\mathcal{S})$ remains equal to the set of events of $\mathcal{S}$.

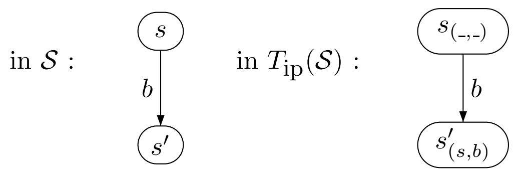


Figure 4: An illustration for Definition 4

Remark that for each transition in $T_{\text{ip}}(\mathcal{S})$ of the form $r'_{(r,a)} \xrightarrow{\sigma} s'_{(s,b)}$ we can deduce, by construction of $\widetilde{t}$, that $s = r'$ on the one hand, and that $\sigma = b$ on the other hand.

Assume the process of Figure 5, with no atomic propositions and the result $T_{\text{ip}}(\mathcal{S})$. It can be checked that the binary relation $\{(1,1),(2,2_{(1,b)}),(2,2_{(2,a)}),(3,3_{(2,a)}),(4,4_{(2,a)})\}$ is a bisimulation between $\mathcal{S}$ and $T_{\text{ip}}(\mathcal{S})$, which leads us to the statement of Proposition 1 :

Proprosition 1
For any process $\mathcal{S}$, we have $\mathcal{S} \underline{\leftrightarrow} T_{ip}(\mathcal{S})$.

Proof
One can check that $\mathcal{R} \subseteq S \times \widetilde{S}$ defined by $(s^0, \widetilde{s^0}) \in \mathcal{R}$ and $(s', s'_{(s,a)}) \in \mathcal{R}$, for all $s \in S$, and for all $a \in \Sigma$ s.t. $s_{(s,a)} \in \widetilde{S}$, is such that $\mathcal{R} : \mathcal{S} \underline{\leftrightarrow} T_{ip}(\mathcal{S})$.

$\diamondsuit$

For complexity issues, it is clear that the size of $T_{\text{ip}}(S)$ is in $O(|\mathcal{S}| \times |\mathcal{S}| \times |\Sigma|)$.

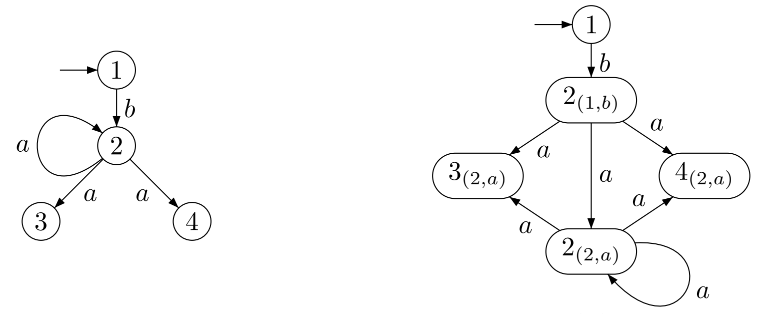


Figure 5: Process $\mathcal{S}$ and process $T_{\text{ip}}(\mathcal{S})$

3.2 The Transformation $\mathcal{T}_{\leq}$

The principle for $\mathcal{T}_{\leq}$ relies on the original method as in Figure 2.

Let $\mathcal{S} = \langle \Gamma, S, s^0, t, L\rangle$ and let $\leq$ be a total order on S. Define $Succ(s,a)$ by $\{t(s,a), \leq\}$, as the set of a-successors of s ordered by $\leq$. We write $s_1 <_i s_2$ whenever $s_1 \neq s_2$ are both in $Succ(s,a)$, and $s_1 \leq s_2$, and there is nothing in between, that is no $s_3 \in Succ(s,a)$ s.t. $s_1 \leq s_3 \leq s_2$.

Definition 5 (Transformation $\mathcal{T}_{\leq}$)
The process $\mathcal{T}_{\leq}(\mathcal{S}) = \langle \Gamma, S, s^0, t^\tau, L\rangle$ on the set of events $\Sigma^\tau = \Sigma \cup \{\tau\}$ is essentially defined by its transition relation t^τ since the remaining is left unchanged : if $|Succ(s,a)| \leq 1$ then $t^\tau(s,a) = t(s,a)$; otherwise let $Succ(s,a) = \{s_1, s_2, ..., s_n\}$ with $s_1 <_i s_2 <_i ... <_i s_n$. Then:

$$\begin{cases} t^\tau(s,a) = s_1 \text{ with } s_1 = min(Succ(s,a)), \text{ and} \\ t^\tau(s_i,\tau) = s_{i+1}, \forall i < |Succ(s,a)| \end{cases}$$

In the sequel, we shall say that s_1, s_2, ... s_n belong to the same *chain* of τ (transitions) from the *source state* s_1.

The figure below Figure 6 shows an example of applying $\mathcal{T}_{\leq}$ for the total order $5 \leq 2 \leq 3 \leq 4 \leq 1$. It has two cases of nondeterminism to solve, namely :

- $Succ(2,a) = \{3,4,5\}$ with $5 <_i 3 <_i 4$;
- $Succ(3,a) = \{3,4,5\}$ with $5 <_i 3 <_i 4$.

The complexity for the transformation $\mathcal{T}_{\leq}$ is clearly linear since each τ transition added comes from the removal of some original transition.

Now, T_{ip} and $\mathcal{T}_{\leq}$ are composed. Write $cod_{\leq}$ for the transformation $(\mathcal{T}_{\leq} \circ T_{\text{ip}})$. and write $\mathcal{S}^\tau$ the result of applying $cod_{\leq}$ to $\mathcal{S}$.

$$cod_{\leq}(\mathcal{S}) = (\mathcal{T}_{\leq} \circ T_{\text{ip}})(\mathcal{S}) = \mathcal{T}_{\leq}(T_{\text{ip}}(\mathcal{S})) = \mathcal{S}^\tau.$$

The following fundamental result can be proved :

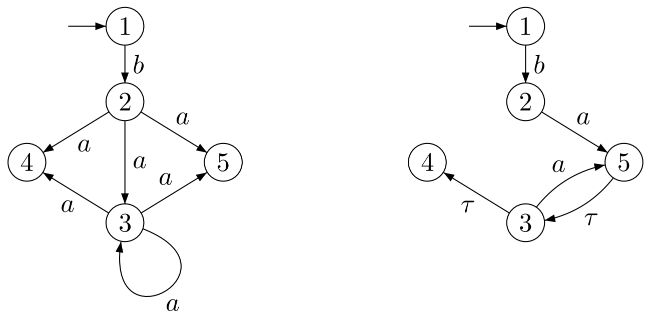


Figure 6: Process $\mathcal{S}$ and process $\mathcal{T}_{\leq}(\mathcal{S})$

Theorem 1
Let $\mathcal{S}$ be a nondeterministic process, then the process $cod_{\leq}(\mathcal{S})$ is deterministic with size in $O(|\mathcal{S}|^2)$.

Proof
We put :

- $\mathcal{S} =< \Gamma, S, s^0, t, L >$
- $\widetilde{\mathcal{S}} = T_{ip}(\mathcal{S}) =< \Gamma, \widetilde{S}, \widetilde{s^0}, \widetilde{t}, \widetilde{L} >$
- $\mathcal{S}^\tau = \mathcal{T}_{\leq}(\widetilde{\mathcal{S}}) =< \Gamma, \widetilde{S}, \widetilde{s^0}, t^\tau, \widetilde{L} >$

Assume that $\mathcal{S}^\tau$ is not deterministic. Two cases can occur :
***First case :** we are in the following situation :*

$$\begin{tikzcd}
 & r'_{(r,a)} \arrow[dl, "\sigma"'] \arrow[dr, "\sigma"] & \\
s'_{1(s_1,b)} & & s'_{2(s_2,c)}
\end{tikzcd}$$

- *if $\sigma \neq \tau$:*
  Necessarily, $\widetilde{t}(r'_{(r,a)}, \sigma) = s'_{1(s_1,b)}$ and $\widetilde{t}(r'_{(r,a)}, \sigma) = s'_{2(s_2,c)}$ because $\sigma \neq \tau$.
  By definition of t^τ, we have :

$$\begin{aligned} {s'_1}_{(s_1,b)} &= min(Succ(r'_{(r,a)}, b)) \quad \text{and} \\ {s'_2}_{(s_2,c)} &= min(Succ(r'_{(r,a)}, b)) \end{aligned}$$

because $\sigma \neq \tau$. Then : ${s'_1}_{(s_1,b)} = {s'_2}_{(s_2,c)}$.

- *if $\sigma = \tau$:*

Assume that ${s'_2}_{(s_2,c)}$, $r'_{(r,a)}$ and ${s'_1}_{(s_1,b)}$ belong to the same chain of τ. Let $q'_{(q,e)}$ be the source state such that :

$$\{{s'_2}_{(s_2,c)}, r'_{(r,a)}, {s'_1}_{(s_1,b)}\} \in Succ(q'_{(q,e)}, d).$$

But according to the τ-transitions, we have also :

$$r'_{(r,a)} <_i {s'_1}_{(s_1,b)} \text{ et } r'_{(r,a)} <_i {s'_2}_{(s_2,c)}$$

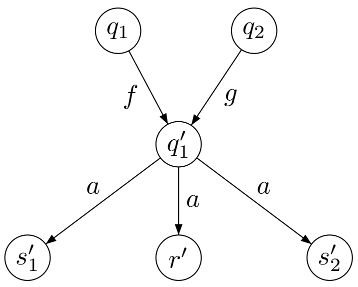
The τ-transitions belong to two different chains of τ. Then the process $\widetilde{\mathcal{S}}$ looks like that :

$$\begin{tikzcd} & {q'_1}_{(q_1,f)} \arrow[dl, "d"'] \arrow[dr, "d"] & & {q'_2}_{(q_2,g)} \arrow[dl, "e"'] \arrow[dr, "e"] & \\ {s'_1}_{(s_1,b)} & & r'_{(r,a)} & & {s'_2}_{(s_2,c)} \end{tikzcd}$$

By definition of $\widetilde{t}$, we have on the one hand $r = q'_1 = s_1$ and $b = d = a$ and, on the other hand, $r = q'_2 = s_2$ and $a = e = c$. Then the process $\widetilde{\mathcal{S}}$ is :

$$\begin{tikzcd} & {q'_1}_{(q_1,f)} \arrow[dl, "a"'] \arrow[dr, "a"] & & {q'_1}_{(q_2,g)} \arrow[dl, "a"'] \arrow[dr, "a"] & \\ {s'_1}_{(q'_1,a)} & & r'_{(q'_1,a)} & & {s'_2}_{(q'_1,a)} \end{tikzcd}$$

According to the immediate past stored in the state, the process $\mathcal{S}$ should be :

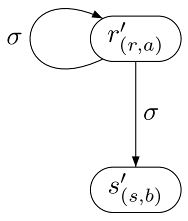


But, in $\widetilde{\mathcal{S}}$, we should have :

$$Succ(q'_{1(q_1,f)}) = \{s'_{1(q'_1,a)}, r'_{(q'_1,a)}, s'_{2(q'_1,a)}\}$$

that is to say that $s'_{1(q'_1,a)}$, $r'_{(q'_1,a)}$ and $s'_{2(q'_1,a)}$ belong to the same chain of τ. Contradiction ! This case is impossible.

Second case : *we are in the following situation :*



- *if $\sigma \neq \tau$:*
  By definition of t^τ, we have necessarily $\widetilde{t}(r'_{(r,a)}, \sigma) = r'_{(r,a)}$ and $\widetilde{t}(r'_{(r,a)}, \sigma) = s'_{(s,b)}$ because $\sigma \neq \tau$. And moreover :

$$\begin{array}{rcl} r'_{(r,a)} & = & min(Succ(r'_{(r,a)}, a)) \quad \text{and} \\ s'_{(r,a)} & = & min(Succ(r'_{(r,a)}, a)) \end{array}$$

  Then we have : $r'_{(r,a)} = s'_{(r,a)}$.

- *if $\sigma = \tau$:*
  This case is impossible because, by definition of t^τ we don't create loops on τ : $t^\tau(r'_{(r,a)}, \tau) = r'_{(r,a)}$ means that $r'_{(r,a)} <_i r'_{(r,a)}$ which is impossible.

$\diamondsuit$

4 The Logical Framework

In this section, we assume given a set $Var = \{Y, Z, \dots\}$ of variables.

Definition 6 (Syntax of Mu-calculus [Koz83])
Given $\Gamma \subseteq AP$, the set of formulas of the Mu-calculus over Γ, written $L_\mu(\Gamma)$, is inductively defined by :

$$\top \mid p \mid Y \mid \langle a \rangle \, \beta_1 \mid \neg\beta_1 \mid \beta_1 \vee \beta_2 \mid \mu Y.\beta_1(Y)$$

with $p \in \Gamma$, $Y \in Var$, $a \in \Sigma$, and $\beta_1, \beta_2 \in L_\mu(\Gamma)$. Moreover, in order to make consistent the semantics of fix-point formulas, like $\mu Y.\beta_1(Y)$, we require variable Y to range under the scope of an even number of $\neg$ symbol in $\beta_1(Y)$. A variable Y is free in the formula β if it is not under the scope of a fix-point operator μ.

Finally, we introduce the following simplifying notations :

- $[a]\beta_1 = \neg \langle a \rangle (\neg\beta_1)$
- $\beta_1 \wedge \beta_2 = \neg(\neg\beta_1 \vee \neg\beta_2)$
- $\nu Y.\beta_1(Y) = \neg\mu Y.\beta_1(\neg Y)$.
- $AG(\beta_1) = \neg\mu Y. \bigwedge_{a\in\Sigma} \langle a \rangle \, Y \wedge \neg\beta_1$.
  The notation AG comes from the temporal logic CTL; $AG(\beta)$ means "β always holds" or equivalently "β is an invariant".
- $\langle ab^* \rangle \, \beta = \langle a \rangle \, (\mu Y. \langle b \rangle \, Y \vee \beta)$.

Mu-calculus formulas are interpreted over a nondeterministic process $\mathcal{S} = \langle \Gamma, S, s^0, t, L \rangle$. Each formula interprets as a subset of states, which by convention are those which satisfy the formula. Due to variable formulas like Y, we need to assume given a *valuation val* : $Var \to 2^S$.

Definition 7 (Semantics of Mu-calculus)
The semantics of a formula α is written $[\![\alpha]\!]_{\mathcal{S}}^{[val]}$ and is inductively defined by induction on the structure of α :

$$[\![\top]\!]_{\mathcal{S}}^{[val]} = S \qquad [\![p]\!]_{\mathcal{S}}^{[val]} = \{s \in S \mid p \in L(s)\}$$

$$[\![Y]\!]_{\mathcal{S}}^{[val]} = val(Y) \qquad [\![\neg\beta_1]\!]_{\mathcal{S}}^{[val]} = S \setminus [\![\beta_1]\!]_{\mathcal{S}}^{[val]}$$

$$[\![\beta_1 \vee \beta_2]\!]_{\mathcal{S}}^{[val]} = [\![\beta_1]\!]_{\mathcal{S}}^{[val]} \cup [\![\beta_2]\!]_{\mathcal{S}}^{[val]}$$

$$[\![\langle a \rangle \, \beta_1]\!]_{\mathcal{S}}^{[val]} = \{s \in S \mid \exists s' : s' \in t(s,a) \text{ and } s' \in [\![\beta_1]\!]_{\mathcal{S}}^{[val]}\}$$

$$[\![\mu Y.\beta_1(Y)]\!]_{\mathcal{S}}^{[val]} = \bigcap\{V \subseteq S \mid [\![\beta_1]\!]_{\mathcal{S}}^{[val(V/Y)]} \subseteq V\}$$

where $val(V/Z) : Var \rightarrow 2^S$ *is the valuation defined by* $val(V/Z)(Y) = val(Y)$ *if* $Y \neq Z$, V *otherwise.*

According to the definitions above, it can be shown that if a formula β does not contain free variables, its semantics is independent of val, in which case we simply write $[\![\beta]\!]_{\mathcal{S}}$. By default, we consider formulas with no free variables.

As first proposed by [RP03b], the Mu-calculus extends to the *Quantified Mu-calculus* by allowing quantifications of the form $\exists\Lambda$ for a set of atomic propositions Λ. For simplicity in this paper, we actually will focus on a fragment of the logic where Λ is a set of atomic propositions indexed by Σ; however the results definitely generalize to any $\Lambda \subseteq AP$.

Definition 8 (Syntax of Quantified Mu-calculus)
The syntax of the (in this paper, restricted) Quantified Mu-calculus over Γ $(\subseteq AP)$, *written* $\mathrm{Q}L_\mu(\Gamma)$, *is :*

$$\exists X.\ \alpha | \neg\alpha_1 | \alpha_1 \vee \alpha_2 | \beta$$

where $X \subseteq AP$ *is a set of propositions* $X = \{x_a | a \in \Sigma\}$ *indexed by* Σ *and disjoint from* Γ, α *is a formula of* $\mathrm{Q}L_\mu(\Gamma \cup X)$, α_1 *and* α_2 *are formulas of* $\mathrm{Q}L_\mu(\Gamma)$, *and* $\beta \in L_\mu(\Gamma)$. *Notice that it is not allowed to use quantification inside fix-point formulas; quantifications and fix-point operators do not commute in general.*

The semantics of $\mathrm{Q}L_\mu(\Gamma \cup X)$ is given by means of labeling processes :

Definition 9 (Labeling process)
An X-labeling process *is a* **complete deterministic process** *over* X. *The set of* X*-labeling processes is written* Lab_X, *and a typical element will be* $\mathcal{E}$.

For a process $\mathcal{S} = \langle\Gamma, S, s^0, t, L\rangle$ with Γ is disjoint from X, the synchronous product $\mathcal{S} \times \mathcal{E}$ is *a labeling of* $\mathcal{S}$ *(by* $\mathcal{E}$*) over* X or an X*-labeling* (of $\mathcal{S}$ by $\mathcal{E}$ over X). An X-labeling of $\mathcal{S}$ hence is some unfolding of $\mathcal{S}$ with propositions x_a put somehow.

Definition 10 (Semantics of Quantified Mu-calculus)
$[\![\exists X.\ \alpha]\!]_{\mathcal{S}}^{[val]}$ *is the set of states* $s \in S$ *for which there exists* $\mathcal{E} = \langle X, E, \epsilon^0, t', L'\rangle \in Lab_X$ *with* $(s, \epsilon^0) \in [\![\alpha]\!]_{\mathcal{S}\times\mathcal{E}}^{[val\times E]}$, *where* $(val \times E)$ *maps each* $Y \in Var$ *onto* $val(Y) \times E$.

We write $\mathcal{S} \models \alpha$, *and say that* $\mathcal{S}$ satisfies the formula α *or* $\mathcal{S}$ is a model of α, *whenever* $s^0 \in [\![\alpha]\!]_{\mathcal{S}}$, *which we write* $\mathcal{S} \models \alpha$.

Essentially, $\mathcal{S} \models \exists X.\ \alpha$ ensures the existence of $\mathcal{E} \in Lab_X$ s.t. $\mathcal{S} \times \mathcal{E} \models \alpha$. In other words, there is a way to place propositions of X in $\mathcal{S}$ so that formula α holds. Notice that although processes might be nondeterministic, labeling processes are always deterministic. *De facto*, the propositions added by the labeling are placed in a consistent way : a proposition $x \in X$ is put on some a-successor if and only if it is put on any of the a-successors.

We finally introduce "looping propositions" to the logic $\mathrm{Q}L_\mu(\Gamma)$, as originally considered by [AVW03], and later [RP03a]. Such propositions have the form $\circlearrowleft^a$ which semantics is the existence of a looping a-transition. Actually, we will only need to interpret $\circlearrowleft^a$ onto labeling processes, hence deterministic processes.

Definition 11 (Looping propositions)
$[\![\circlearrowleft^a]\!]_{\mathcal{S}}$ *is the set of states s.t. s is its own a-successor.*

For example, a process which invariantly loops on τ-transitions (say because τ is unobservable) would satisfy the (Mu-Calculus + loops)-formula $\mathit{Loop}(\tau)$ defined by :

$$\mathit{Loop}(\tau) \stackrel{\text{def}}{=} AG(\circlearrowleft^\tau)$$

In the *Loop Quantified Mu-Calculus*, introduced by [RP03a, Rie03], which extends $\mathrm{Q}L_\mu$, it is possible to enforce the membership of labeling processes in (Mu-Calculus + loops)-definable classes, like $\mathit{Loop}(\tau)$: assertions like $\exists X.\ \alpha$ are enriched to state $\exists X \in \beta^{\circlearrowleft}.\ \alpha$, where $\beta^{\circlearrowleft}$ is a (Mu-Calculus + loops)-formula, possibly containing propositions $\circlearrowleft^a$. The full theory of this logic can be found in [Rie03] and [RP03a]. However, in the context of this paper, we only need a limited syntactic fragment, which enables us to write the formula of Theorem 2.

Let us denote by $^{\circlearrowleft}\mathrm{Q}L_\mu(\Gamma)$ this logic.

5 The Control of Nondeterministic Systems

Assume that in the framework of **deterministic processes** there is a method for solving control problems under partial observation when the objectives are given in the mu-calculus; this is precisely the case in [RP03a] (and also [Rie03]) which we recall the principles in the next section.

Now, by using Theorem 3 further, we infer in Section 5.2 a method for answering control problems for nondeterministic processes.

5.1 The Control of Deterministic Processes

We briefly recall the narrow link between control problems under partial observation and the model-checking of $^{\circlearrowleft}\mathrm{Q}L_\mu(\Gamma)$-formulas, as originally explained in details by [RP03a].

In order to ease the reading, we use Σ', X', β' ... instead of respectively Σ, X, β ... to put the emphasis on the deterministic framework.

Basically, given an unobservable event $\tau \in \Sigma'$, and a partition of Σ' into $\Sigma'_u \uplus \Sigma'_c$, the following holds [RP03a, Rie03] :

Theorem 2
Assume given a process $\mathcal{S}'$ with events in $\Sigma' = \Sigma'_u \uplus \Sigma'_c$ which is deterministic, since for example resulting from a transformation by $cod_{\leq}$ (see Section 3), assume also given an unobservable event $\tau \in \Sigma'_u$ and a Mu-calculus formula β'. Then, there exists a controller $\mathcal{C}'$ of $\mathcal{S}'$ for β' which does not observe $\tau \in \Sigma'_u$[1] iff :

$$\mathcal{S}' \models \exists X' \in (AG(\textstyle\bigwedge_{a \in \Sigma'_u} x_a) \wedge Loop(\tau).\ f(\beta', X')$$

where $f(\beta', X') \in L_\mu$ depends on β' and propositions $x_a \in X'$ ($a \in \Sigma'$); it expresses in particular the uncontrollability of events in Σ_u.

According to [RP03a], $\exists X' \in (AG(\bigwedge_{a \in \Sigma'_u} x_a) \wedge Loop(\tau).\ f(\beta', X')$ means there exists an X'-labeling process, say $\mathcal{E}^{\circlearrowleft}$, s.t. : $\mathcal{E}^{\circlearrowleft}$ loops on every τ-transition and all its states possess propositions x_a for $a \in \Sigma'_u$, in particular x_τ.

[RP03a] have established a synthesis procedure for $\mathcal{E}^{\circlearrowleft}$, inspired from [AVW03]. To obtain the controller $\mathcal{C}'$, $\mathcal{E}^{\circlearrowleft}$ needs being pruned : for each state and each proposition $x_a \in X'$, we remove the outgoing a-transition whenever the state is not labeled by x_a. We write $(\mathcal{E}^{\circlearrowleft})_{X'_{\rightarrow}}$ the resulting. In particular, because proposition x_a always holds when $a \in \Sigma'_u$ (see the formula in Theorem 2), all uncontrollable remain in $(\mathcal{E}^{\circlearrowleft})_{X'_{\rightarrow}}$. By construction, the process $(\mathcal{E}^{\circlearrowleft})_{X'_{\rightarrow}}$ achieves $\mathcal{S}' \times (\mathcal{E}^{\circlearrowleft})_{X'_{\rightarrow}} \models \beta'$ and always allows uncontrollable transitions. Hence it is a controller of $\mathcal{S}'$ for β'.

We recall that $(\mathcal{E}^{\circlearrowleft})_{X_{\rightarrow}}$ is such that $\mathcal{S}' \times (\mathcal{E}^{\circlearrowleft})_{X_{\rightarrow}} \models \beta'$, and that $(\mathcal{E}^{\circlearrowleft})_{X_{\rightarrow}}$ has an a-loop in each state for every $a \in \Sigma_{uo}$.

Actually, Theorem 2 can be made a lot more general : for example, several controllers with different sets of observation can be specified (but not necessarily synthesized due to undecidability reasons), or universal quantifications can be used to deal with maximally permissive controllers [RP04].

5.2 The Control of Nondeterministic Processes

We first explain how the model-checking of a $\mathrm{Q}L_\mu$-formula α on a nondeterministic process $\mathcal{S}$ reduces to the model-checking of some $^{\circlearrowleft}\mathrm{Q}L_\mu$-formula, written $\mathrm{Tr}(\alpha)$, on the process $cod_{\leq}(\mathcal{S})$.

Since the size of $cod_{\leq}(\mathcal{S})$ is quadratic in the size of $\mathcal{S}$ and because the size of $\mathrm{Tr}(\alpha)$, as we will see, is linear in the size of α, this reduction is polynomial and the two problems belong to the same complexity class.

We propose a translation Tr from $\mathrm{Q}L_\mu$ to $^{\circlearrowleft}\mathrm{Q}L_\mu$ which aims at telling a $\mathrm{Q}L_\mu$-statement on a nondeterministic process $\mathcal{S}$ as a statement on $cod_{\leq}(\mathcal{S})$: for example, the existence of a a-successor in $\mathcal{S}$ translates to the existence of an $a\tau^*$-successor in $cod_{\leq}(\mathcal{S})$, hence the translation for formulas like $\langle a \rangle\, \beta_1$ below. Also, since $cod_{\leq}(\mathcal{S})$ has now transitions labeled on $\Sigma' = \Sigma \cup \{\tau\}$, we turn existential quantifications $\exists X$ into $\exists X \cup \{x_\tau\} \in (AG(x_\tau) \wedge Loop(\tau))$ so that τ-transitions in $cod_{\leq}(\mathcal{S})$ will not be observed nor disallowed by the controllers. In

[1]Here, τ is also supposed uncontrollable.

the following, we take the convention that

$$\textsc{loop}(\tau) \overset{\text{def}}{=} AG(x_\tau) \wedge \mathit{Loop}(\tau)$$

Definition 12
Let $\mathrm{Tr} : \mathrm{Q}L_\mu(\Gamma) \to^{\circlearrowleft} \mathrm{Q}L_\mu(\Gamma \cup \{x_\tau\})$ *be the translation inductively defined by :*

$\mathrm{Tr}(\top) = \top \quad \mathrm{Tr}(p) = p \quad \mathrm{Tr}(Y) = Y$
$\mathrm{Tr}(\langle a \rangle \beta_1) = \langle a\tau^* \rangle \mathrm{Tr}(\beta_1)$
$\mathrm{Tr}(\neg\beta_1) = \neg\mathrm{Tr}(\beta_1)$
$\mathrm{Tr}(\beta_1 \vee \beta_2) = \mathrm{Tr}(\beta_1) \vee \mathrm{Tr}(\beta_2)$
$\mathrm{Tr}(\mu Y.\beta_1(Y)) = \mu Y.\mathrm{Tr}(\beta_1(Y))$
$\mathrm{Tr}(\exists X.\ \alpha) = \exists X \cup \{x_\tau\} \in \textsc{loop}(\tau).\ \mathrm{Tr}(\alpha)$

The translation Tr indeed has the desired property :

Theorem 3
Let $\mathcal{S}$ *be a process and let* $\alpha \in \mathrm{Q}L_\mu$*,*

$$\mathcal{S} \models \alpha \textit{ if and only if } \mathit{cod}_{\leq}(\mathcal{S}) \models \mathrm{Tr}(\alpha)$$

The proof of Theorem 3 is done by induction over the structure of α. In particular, for existential quantifications, which specify the existence of a labeling process such that something holds, the proof goes through an intermediate powerful result stated in Proposition 3.

Definition 13
Let $\mathcal{E}^{\circlearrowleft}$ *be an* $(X \cup \{x_\tau\})$*-labeling process with event set* $\Sigma \cup \{\tau\}$ *and satisfying* $\textsc{loop}(\tau)$*. Notice that such a labeling process is specified by all formulas* $\mathrm{Tr}(\exists X.\ \ldots)$*. We apply the* τ-forgetting application l *to get the* X*-labeling process written* $l(\mathcal{E}^{\circlearrowleft})$*: it has the same states as* $\mathcal{E}^{\circlearrowleft}$ *and its events set is* Σ*; moreover,* $t_{l(\mathcal{E}^{\circlearrowleft})}(e, a) = e'$ *whenever* $t_{\mathcal{E}^{\circlearrowleft}}(e, a) = e'$ *and* $a \neq \tau$*, and* $L_{l(\mathcal{E}^{\circlearrowleft})}(e) = L_{\mathcal{E}^{\circlearrowleft}}(e) \setminus \{x_\tau\}$

The application l have the following property :

Proprosition 2
$l(\mathcal{E}^{\circlearrowleft} \times \mathcal{E}'^{\circlearrowleft}) = l(\mathcal{E}^{\circlearrowleft}) \times l(\mathcal{E}'^{\circlearrowleft})$

Proof
We put $\mathcal{E}^{\circlearrowleft} =< X \cup \{x_\tau\}, E, \epsilon^0, t_{\mathcal{E}^{\circlearrowleft}}, L_{\mathcal{E}^{\circlearrowleft}} >$ *and* $\mathcal{E}'^{\circlearrowleft} =< X \cup \{x_\tau\}, E', \epsilon'^0, t_{\mathcal{E}'^{\circlearrowleft}}, L_{\mathcal{E}'^{\circlearrowleft}} >$*, and respectively* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *the systems* $l(\mathcal{E}^{\circlearrowleft} \times \mathcal{E}'^{\circlearrowleft})$ *and* $l(\mathcal{E}^{\circlearrowleft}) \times l(\mathcal{E}'^{\circlearrowleft})$*. Let's prove that* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *are the same :*

First, $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *have the same set of state in* $E \times E'$ *and the same initial state* $(\epsilon^0, \epsilon'^0)$*.*

Moreover, by synchronous product, the set of atomic propositions for all states (ϵ, ϵ') *of the systems* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *is :* $(L_{\mathcal{E}^{\circlearrowleft}}(\epsilon) \cup L_{\mathcal{E}'^{\circlearrowleft}}(\epsilon')) \setminus \{x_\tau\}$*.*

Last, the transition relation (with $a \neq \tau$*) :*

$$\begin{aligned} t_{\mathcal{S}_2}((\epsilon_1,\epsilon'_1),a)=(\epsilon_2,\epsilon'_2) &\Leftrightarrow t_{l(\mathcal{E}^{\circlearrowleft})}(\epsilon_1,a)=\epsilon_2 \text{ and } t_{l(\mathcal{E}'^{\circlearrowleft})}(\epsilon'_1,a)=\epsilon'_2 \\ &\Leftrightarrow t_{\mathcal{E}^{\circlearrowleft}}(\epsilon_1,a)=\epsilon_2 \text{ and } t_{\mathcal{E}^{\circlearrowleft}}(\epsilon_1,\tau)=\epsilon_1 \\ &\qquad \text{and } t_{\mathcal{E}'^{\circlearrowleft}}(\epsilon'_1,a)=\epsilon'_2 \text{ and } t_{\mathcal{E}'^{\circlearrowleft}}(\epsilon'_1,\tau)=\epsilon'_1 \\ &\Leftrightarrow t_{\mathcal{E}^{\circlearrowleft}\times\mathcal{E}'^{\circlearrowleft}}((\epsilon_1,\epsilon'_1),a)=(\epsilon_2,\epsilon'_2) \\ &\qquad \text{and } t_{\mathcal{E}^{\circlearrowleft}\times\mathcal{E}'^{\circlearrowleft}}((\epsilon_1,\epsilon'_1),\tau)=(\epsilon_1,\epsilon'_1) \\ &\Leftrightarrow t_{\mathcal{S}_1}((\epsilon_1,\epsilon'_1),a)=(\epsilon_2,\epsilon'_2) \end{aligned}$$

Then $\mathcal{S}_1$ and $\mathcal{S}_2$ have the same transition relation and the systems $l(\mathcal{E}^{\circlearrowleft}\times\mathcal{E}'^{\circlearrowleft})$ and $l(\mathcal{E}^{\circlearrowleft})\times l(\mathcal{E}'^{\circlearrowleft})$ are the same.

$\diamondsuit$

Moreover, we can prove that the application l is such that :

Proprosition 3
For any process $\mathcal{S}$ and $\alpha \in \text{QL}_\mu$,

$$\mathcal{E}^{\circlearrowleft} \in Lab_{X\cup\{x_\tau\}} \textit{ satisfying } \text{LOOP}(\tau) \textit{ is s.t. } cod_{\leq}(\mathcal{S})\times\mathcal{E}^{\circlearrowleft} \models \text{Tr}(\alpha)$$
$$\textit{iff}$$
$$\mathcal{S}\times l(\mathcal{E}^{\circlearrowleft}) \models \alpha$$

Proof
This proof is done by induction over the structure of $\alpha \in \text{QL}_\mu$.

- ***case of the quantification** $\alpha = \exists X'.\alpha'$:*

$$\begin{aligned} \mathcal{S}^\tau\times\mathcal{E}^{\circlearrowleft} \models \text{Tr}(\exists X'.\alpha') &\Leftrightarrow \mathcal{S}^\tau\times\mathcal{E}^{\circlearrowleft} \models \exists X'\cup\{x_\tau\}\in\text{LOOP}(\tau).\text{Tr}(\alpha') \\ &\Leftrightarrow \exists\mathcal{E}'^{\circlearrowleft}\in Lab_{X'\cup\{x_\tau\}} : \\ &\qquad (\mathcal{S}^\tau\times\mathcal{E}^{\circlearrowleft})\times\mathcal{E}'^{\circlearrowleft} \models \text{Tr}(\alpha') \wedge \mathcal{E}'^{\circlearrowleft} \models \text{LOOP}(\tau) \\ &\Leftrightarrow \mathcal{S}^\tau\times(\mathcal{E}^{\circlearrowleft}\times\mathcal{E}'^{\circlearrowleft}) \models \text{Tr}(\alpha') \\ &\qquad \wedge\ \mathcal{E}'^{\circlearrowleft} \models \text{LOOP}(\tau) \textit{ and,} \\ &\qquad \textit{by assumption, } \mathcal{E}^{\circlearrowleft} \models \text{LOOP}(\tau) \\ &\Leftrightarrow \mathcal{S}^\tau\times(\mathcal{E}^{\circlearrowleft}\times\mathcal{E}'^{\circlearrowleft}) \models \text{Tr}(\alpha') \wedge \mathcal{E}^{\circlearrowleft}\times\mathcal{E}'^{\circlearrowleft} \models \text{LOOP}(\tau) \\ &\qquad \textit{according to the definition of the synchronous product} \\ &\Leftrightarrow \mathcal{S}\times l(\mathcal{E}^{\circlearrowleft}\times\mathcal{E}'^{\circlearrowleft}) \models \alpha' \\ &\qquad \textit{according to the induction assumption} \\ &\Leftrightarrow \mathcal{S}\times l(\mathcal{E}^{\circlearrowleft})\times l(\mathcal{E}'^{\circlearrowleft}) \models \alpha' \textit{ according to Proposition 2} \\ &\Leftrightarrow \mathcal{S}\times\mathcal{E}\times\mathcal{E}' \models \alpha' \\ &\Leftrightarrow \exists\mathcal{E}'\in Lab_{X'} : (\mathcal{S}\times\mathcal{E})\times\mathcal{E}' \models \alpha' \\ &\Leftrightarrow \mathcal{S}\times\mathcal{E} \models \exists X'.\alpha' \end{aligned}$$

- *Negation $\alpha = \neg\alpha_1$ and disjunction $\alpha = \alpha_1 \vee \alpha_2$ are obvious cases : you have to go back to the semantic of the logic and use the induction assumption to conclude.*

- ***if α is a formula of the pure Mu-calculus :***

  By induction, one can easily prove that :

$$\mathcal{S} \models \alpha \text{ if and only if } cod_{\leq}(\mathcal{S}) \models \mathrm{Tr}(\alpha)$$

  So we can deduce that :

$$\mathcal{S} \times \mathcal{E} \models \alpha \text{ if and only if } cod_{\leq}(\mathcal{S} \times \mathcal{E}) \models \mathrm{Tr}(\alpha)$$

  Furthermore, we can prove, by examining their executive trees, that :

$$cod_{\leq}(\mathcal{S} \times \mathcal{E}) \leftrightarrows cod_{\leq}(\mathcal{S}) \times \mathcal{E}^{\circlearrowleft}$$

  Finally : $\mathcal{S} \times \mathcal{E} \models \alpha$ *if and only if* $cod_{\leq}(\mathcal{S}) \times \mathcal{E}^{\circlearrowleft} \models \mathrm{Tr}(\alpha)$

$\diamondsuit$

We can now prove Theorem 3 :

Proof
This proof is done by induction over the structure of $\alpha \in \mathrm{QL}_{\mu}$. First, the case of the quantification :

$$\begin{array}{lll} \mathcal{S} \models \exists X.\alpha & \Leftrightarrow & \exists \mathcal{E} \in Lab_X \text{ s.t. } \mathcal{S} \times \mathcal{E} \models \alpha \\ & \Leftrightarrow & \exists \mathcal{E}^{\circlearrowleft} \in Lab_{X \cup \{x_\tau\}} \text{ s.t. } \mathcal{S}^{\tau} \times \mathcal{E}^{\circlearrowleft} \models \mathrm{Tr}(\alpha) \\ S & & \text{with } \mathcal{E}^{\circlearrowleft} \models \mathrm{LOOP}(\tau) \\ & \Leftrightarrow & \exists X \cup \{x_\tau\} \in \mathrm{LOOP}(\tau).\mathrm{Tr}(\alpha) \end{array}$$

Here again, the other cases are obvious : you have to go back to the semantic of the logic and use the induction assumption to conclude.

$\diamondsuit$

We can now relate controllers for nondeterministic processes to controllers with unobservable and uncontrollable event τ for the deterministic processes, since, already explained in the previous section, controllers are simply pruned labeling processes. We can formally demonstrate :

Corollary 1
There exists a controller of a nondeterministic process $\mathcal{S}$ for a mu-calculus definable control objective β if and only if there exists a controller of $cod_{\leq}(\mathcal{S})$ for $\mathrm{Tr}(\beta)$ which does not observe nor control event τ. Moreover, the former controller is obtained by forgetting all the τ-loops in the latter. Finally, the complexity classes of control problems for nondeterministic processes are the complexity classes of control problems for deterministic processes with partial observation.

We refer to [RP03a] for the complexity classes of control problems for deterministic processes with partial observation.

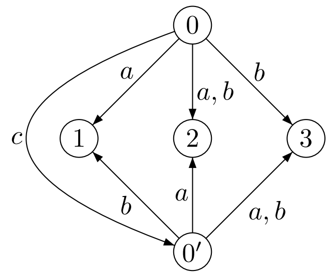
6 Conclusion

We have proposed two transformations that are used to reduce control problems for nondeterministic systems to control problems under partial observation for deterministic systems. As a corollary, the classes of complexity are identical.

It is worthwhile noting that Theorem 3 actually says more, since it gives a polynomial reduction of the model-checking of $\mathrm{Q}L_\mu$ for nondeterministic system into the model-checking of ${}^{\circlearrowleft}\mathrm{Q}L_\mu$ for deterministic systems. Notice also that the key Proposition 3 is more powerful as it holds for the full logic $\mathrm{Q}L_\mu$. Consequently, nested quantifications can be considered. Hence Corollary 1 still holds when maximally permissive controllers are required [RP04]. In fact, Proposition 3 holds of the full logic ${}^{\circlearrowleft}\mathrm{Q}L_\mu$ of [RP03a], where maximally permissive controllers in the class of controllers under partial observation can be specified and synthesized.

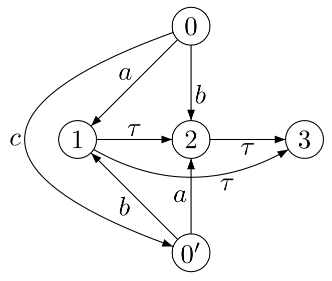
A Example : we can't apply on systems the encoding on trees from [Tho97]

This example points out that we can't apply on systems the encoding on trees from [Tho97]:

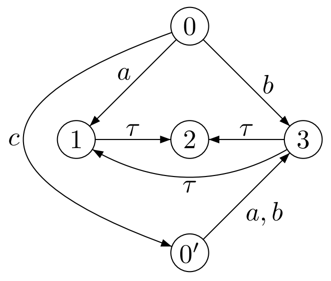


We enumerate the different cases according to the order between the states 1, 2 and 3 (we don't need to sort the states 0 and $0'$) :

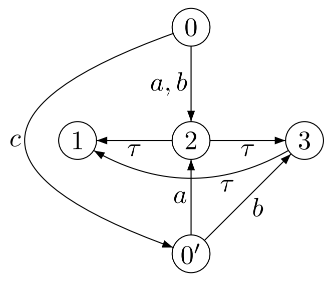
$1 \leq 2 \leq 3$:



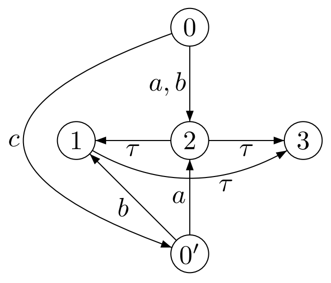
$1 \leq 3 \leq 2$:



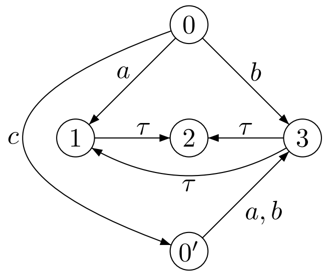
$2 \leq 3 \leq 1$:



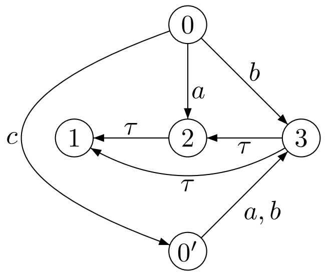
$2 \leq 1 \leq 3$:



$3 \leq 1 \leq 2$:



$3 \leq 2 \leq 1$:



There is no order between the states 1, 2 and 3 that doesn't generate a nondeterminism in τ.

References

[AVW03] A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theorical Computer Science*, 303(1):7–34, 2003.

[Eme90] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B*, chapter 16, pages 995–1072. ELSEVIER, 1990.

[HL98] Michael Heymann and Feng Lin. Discrete-event control of nondeterministic systems. *Proceedings of the IEEE; Transaction on Automatic Control*, 43(1):3–17, January 1998.

[Koz83] D. Kozen. Results on the propositional mu-calculus. *Theorical Computer Science*, 27:333–354, 1983.

[Rie03] S. Riedweg. *Logiques pour le contrle d'automatismes discrets.* PhD thesis, IRISA Rennes, 2003.

[RP03a] S. Riedweg and S. Pinchinat. Loop quantified mu-calculus for control synthesis, 2003. To appear, a research report version is available as an INRIA-RR n4949.

[RP03b] S. Riedweg and S. Pinchinat. Quantified mu-calculus for control synthesis. In *Mathematical Foundations of Computer Science*, Bratislava, Slovak Republic, aot 2003.

[RP04] S. Riedweg and S. Pinchinat. Maximally permissive controllers in all contexts. In *Proc of 7th Workshop on Discrete Event Systems, WODES 2004*, Reims, France, sept 2004.

[Tho97] W. Thomas. Languages, automata and logic. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, Beyond Words. Springer-Verlag, Berlin, 1997.

Contents