SUPERVISORY CONTROL PROBLEMS FOR NONDETERMINISTIC DISCRETE-EVENT SYSTEMS: A LOGICAL APPROACH

Sophie Pinchinat [*,1] **Jean-Baptiste Raclet** [*,1]

[*] *IRISA, Campus de Beaulieu, 35042 RENNES CEDEX*

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.

Keywords: Discrete-Event systems, Controlled systems, Formal specification, Computer-aided system design, Nondeterminism, Mu-calculus.

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 information: one can, for example, abstract from the content of messages along communication channels, while keeping the channels' name. 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 more expressive logic on some derived deterministic system. Both logics originate from (Riedweg and Pinchinat, 2003) and (Riedweg and Pinchinat, 2005) they are adequate to specify controllers for Mu-calculus definable control objectives. Hence, a wide range of control problems reduce to checking some formula on a model - which is known as a *model-checking problem.* Decision issues for both model-checking the existence of controllers and synthesizing them (when possible) are already made clear by (Riedweg and Pinchinat, 2005) for deterministic systems.

We show here how the method can be adapted to nondeterministic systems according to the following: given a controller specification formula α for a nondeterministic system $\mathcal{S}$, we transform both the nondeterministic system $\mathcal{S}$ and the formula α respectively into a deterministic system $\mathcal{S}'$ with a new unobservable event τ and into a formula α' which specifies a controller under partial observation (it cannot observe τ) for $\mathcal{S}'$. Moreover, the problems have the same complexity, since the controller for the former problem is obtained by removing the τ-transitions in a solution of the latter.

This approach seems very close to the pioneering work of (Heymann and Lin, 1998). However, the models we propose are a lot more general. Firstly,

[1] Supported by INRIA

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 features of the systems' behaviors: it subsumes regular languages, ω-regular languages, but also temporal logics like CTL, LTL, CTL*, ... See (Emerson, 1990) for a survey.

As the work of (Riedweg and Pinchinat, 2005) permits the synthesis of a 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 to be deterministic, which perfectly fits the intuition we have: the nondeterminism arises as an abstraction where some events should not be distinguished, even for applying a control.

The paper is organized as follows: Section 2 presents the models; Section 3 describes the transformation on systems; 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 $\Gamma(\subseteq AP)$ is a tuple $\mathcal{S} = \langle \Gamma, S, s^0, t, L\rangle$ where

- S is a finite set of states; $s^0 \in S$,
- $t : S \times \Sigma \to 2^S$ is the transition relation,
- $L : S \to 2^\Gamma$ labels states by 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 and $\mathcal{S}$ 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*: 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 machines shown in Figure 1: the machines can serve either coffee or tea.

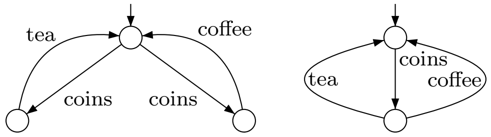

Fig. 1. Two different "coffee" machines.

However, when a customer introduces coins in a machine, and according to her choice, the left-hand side machine can either offer coffee or tea, whereas the right-hand side 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.

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 $\mathcal{S}_1$ and $\mathcal{S}_2$, 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 of *execution tree* of a process: it is obtained by unfolding the process as a tree which (possibly infinite) branches are maximal executions. 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 (Thomas, 1997). We then establish the mathematical properties of this encoding. Basically, a new event called τ is considered. Now if we observe 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).

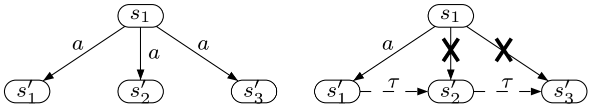

Fig. 2. Encoding of the execution trees

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 perform it on the process itself, but 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.

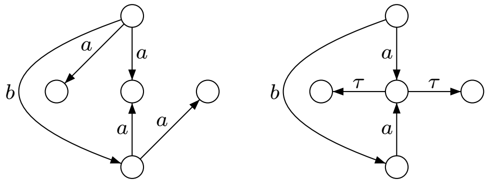

Fig. 3. Generation of nondeterminism in τ.

Alternatively, we propose two successive transformations that need to be 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 as in bisimulation with the encoded execution tree. The proof is omitted here but we refer to (Raclet and Pinchinat, 2004).

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: let $\mathcal{S} = \langle \Gamma, S, s^0, t, L\rangle$ be a nondeterministic process. The process $T_{\text{ip}}(\mathcal{S})$ is $T_{\text{ip}}(\mathcal{S}) = \langle \Gamma, \widetilde{S}, \widetilde{s^0}, \widetilde{t}, \widetilde{L}\rangle$ 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 $\widetilde{L}(s'_{(s,a)}) = L(s')$.

Assume the process of Figure 4, with no 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 2.

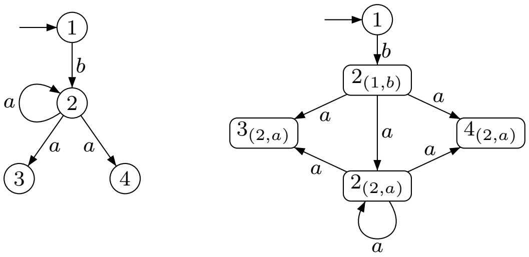

Fig. 4. Process $\mathcal{S}$ and process $T_{\text{ip}}(\mathcal{S})$

Proposition 2. For any process $\mathcal{S}$, $\mathcal{S} \underline{\leftrightarrow} T_{\text{ip}}(\mathcal{S})$.

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 $\mathcal{R} : \mathcal{S} \underline{\leftrightarrow} T_{\text{ip}}(\mathcal{S})$.

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

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. $s_1 \leq s_3 \leq s_2$. We define $\mathcal{T}_{\leq}(\mathcal{S}) = \langle \Gamma, S, s^0, t^\tau, L\rangle$ with the set of events $\Sigma^\tau = \Sigma \cup \{\tau\}$; $\mathcal{T}_{\leq}(\mathcal{S})$ 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

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

The figure below shows an example of applying $\mathcal{T}_{\leq}$ for the total order $5 \leq 2 \leq 3 \leq 4 \leq 1$.

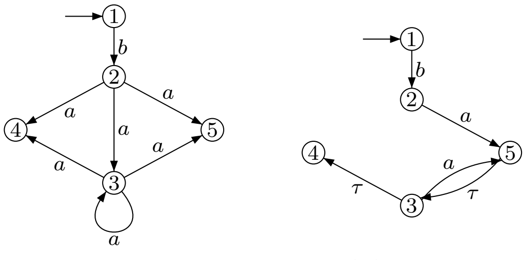

Fig. 5. Process $\mathcal{S}$ and process $\mathcal{T}_{\leq}(\mathcal{S})$

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 $(\mathcal{T}_{\leq} \circ T_{\text{ip}})$. The following fundamental result can be proved (omitted here):

Theorem 3. Let $\mathcal{S}$ be a nondeterministic process, then the process $cod_{\leq}(\mathcal{S})$ is deterministic with size in $O(|\mathcal{S}|^2)$.

4. THE LOGICAL FRAMEWORK

Assume given a set $Var = \{Y, Z, \ldots\}$ of variables:

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

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

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 syntactic constraints in the formulas. 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". Finally, $\langle ab^* \rangle\, \beta_1 = \langle a \rangle\, (\mu Y. \langle b \rangle\, Y \vee \beta_1)$ means that there exists a sequence of events in the regular language ab^*.

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 assume given a valuation $val : Var \to 2^S$. The semantics of a formula α is written $[\![\alpha]\!]_{\mathcal{S}}^{[val]}$ and is 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 $val(V/Y) : Var \to 2^S$ is the valuation defined by $val(V/Y)(Z) = val(Z)$ if $Z \neq Y$, 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 (Riedweg and Pinchinat, 2003), 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 $\Lambda \subseteq AP$.

The syntax of the (in this paper, restricted) Quantified Mu-calculus over Γ ($\subseteq AP$), written $\text{Q}L_\mu(\Gamma)$, 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 $\text{Q}L_\mu(\Gamma \cup X)$, α_1 and α_2 are formulas of $\text{Q}L_\mu(\Gamma)$, and $\beta \in L_\mu(\Gamma)$. Notice that one can not use quantification inside fix-point formulas; quantifications and fix-point operators do not commute in general.

The semantics of $\text{Q}L_\mu(\Gamma \cup X)$ is given by means of labeling processes: 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 $\mathcal{S} = \langle \Gamma, S, s^0, t, L \rangle$ where 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). As $\mathcal{E}$ is complete, an X-labeling of $\mathcal{S}$ amounts to adding propositions from X to the states of $\mathcal{S}$.

Now, $[\![\exists X.\ \alpha]\!]_{\mathcal{S}}^{[val]}$ 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 $s^0 \in [\![\alpha]\!]_{\mathcal{S}}$.

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 $\text{Q}L_\mu(\Gamma)$, as originally considered by (Arnold *et al.*, 2003), and later (Riedweg and Pinchinat, 2005). Such propositions have the form $\circlearrowleft^a$ which semantics is the existence of a looping a-transition. Actually, we will only need to interpret $\circlearrowleft^a$ over labeling processes, hence deterministic processes. Formally $[\![\circlearrowleft^a]\!]_{\mathcal{S}}$ 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:

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

In the *Loop Quantified Mu-Calculus*, introduced by (Riedweg and Pinchinat, 2005; Riedweg, 2003), and which extends $\mathrm{Q}L_\mu$, it is possible to enforce the membership of labeling processes in (Mu-Calculus + loops)-definable classes, like $Loop(\tau)$: 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 (Riedweg, 2003) and (Riedweg and Pinchinat, 2005). However, in the context of this paper, we only need a limited syntactic fragment, which enables us to write the formula of Theorem 4.

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 (Riedweg and Pinchinat, 2005) (and also (Riedweg, 2003)) from which we recall the principles in the next section.

Now, by using Theorem 5 further, we infer in Section 5.2 a method for solving 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 detail by (Riedweg and Pinchinat, 2005).

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 (Riedweg and Pinchinat, 2005; Riedweg, 2003):

Theorem 4. Assume given a deterministic process $\mathcal{S}'$ with events in $\Sigma' = \Sigma'_u \uplus \Sigma'_c$, 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 $\tau \in \Sigma'_u$ [2] 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 (Riedweg and Pinchinat, 2005), $\exists X' \in (AG(\bigwedge_{a\in\Sigma'_u} x_a) \wedge Loop(\tau)).\ f(\beta', X')$ 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.

(Riedweg and Pinchinat, 2005) have established a synthesis procedure for $\mathcal{E}^{\circlearrowleft}$, inspired by (Arnold *et al.*, 2003). To obtain the controller $\mathcal{C}'$, $\mathcal{E}^{\circlearrowleft}$ needs to be 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 $(\mathcal{E}^{\circlearrowleft})_{X'\to}$ for the resulting process. In particular, because proposition x_a always holds when $a \in \Sigma'_u$ (see the formula in Theorem 4), all uncontrollable events remain in $(\mathcal{E}^{\circlearrowleft})_{X'\to}$. By construction, the process $(\mathcal{E}^{\circlearrowleft})_{X'\to}$ achieves $\mathcal{S}' \times (\mathcal{E}^{\circlearrowleft})_{X'\to} \models \beta'$ and always allows uncontrollable transitions. Hence it is a controller of $\mathcal{S}'$ for β'.

Actually, Theorem 4 can be made a lot more general: for example, several controllers with different sets of observation can be specified (but not necessarily synthesized for undecidability reasons), or universal quantifications can be used to deal with maximally permissive controllers (Riedweg and Pinchinat, 2004).

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 transforms a $\mathrm{Q}L_\mu$-statement on a nondeterministic process $\mathcal{S}$ into a statement on $cod_{\leq}(\mathcal{S})$: for example, the existence of an a-successor in $\mathcal{S}$ translates to the existence of an $a\tau^*$-successor in $cod_{\leq}(\mathcal{S})$, hence the result for $\mathrm{Tr}(\langle a\rangle\,\beta_1)$ 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 the following, we take the convention that

$$\mathrm{LOOP}(\tau) \overset{\text{def}}{=} AG(x_\tau) \wedge Loop(\tau)$$

[2] Here, τ is also supposed uncontrollable.

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

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

Theorem 5. Let $\mathcal{S}$ be a process and let $\alpha \in \mathrm{Q}L_\mu$,

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

The proof of Theorem 5 is done by induction over the structure of α. It is tedious, we refer to (Raclet and Pinchinat, 2004) for full proofs. 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 6.

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 *$\tau$-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, $e \xrightarrow{a}_{l(\mathcal{E}^{\circlearrowleft})} e'$ whenever $e \xrightarrow{a}_{\mathcal{E}^{\circlearrowleft}} e'$ and $a \neq \tau$, and $L_{l(\mathcal{E}^{\circlearrowleft})}(e) = L_{\mathcal{E}^{\circlearrowleft}}(e) \setminus \{x_\tau\}$

Proposition 6. For any process $\mathcal{S}$ and $\alpha \in \mathrm{Q}L_\mu$,

$$\begin{gathered}
\mathcal{E}^{\circlearrowleft} \in \mathit{Lab}_{X \cup \{x_\tau\}} \text{ satisfying } \textsc{loop}(\tau) \text{ is s.t.}\\
\mathit{cod}_{\leq}(\mathcal{S}) \times \mathcal{E}^{\circlearrowleft} \models \mathrm{Tr}(\alpha)\\
\text{iff}\\
\mathcal{S} \times l(\mathcal{E}^{\circlearrowleft}) \models \alpha
\end{gathered}$$

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:

Theorem 7. 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 $\mathit{cod}_{\leq}(\mathcal{S})$ 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 (Riedweg and Pinchinat, 2005) 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 5 actually says more, since it gives a polynomial reduction of the model-checking of $\mathrm{Q}L_\mu$ for nondeterministic systems into the model-checking of ${}^{\circlearrowleft}\mathrm{Q}L_\mu$ for deterministic systems. Notice also that the key Proposition 6 is more powerful as it holds for the full logic $\mathrm{Q}L_\mu$. Consequently, nested quantifications can be considered. Hence Theorem 7 still holds when maximally permissive controllers are required (Riedweg and Pinchinat, 2004). In fact, Proposition 6 holds of the full logic ${}^{\circlearrowleft}\mathrm{Q}L_\mu$ of (Riedweg and Pinchinat, 2005), where maximally permissive controllers in the class of controllers under partial observation can be specified and synthesized.

REFERENCES

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

Emerson, E.A. (1990). Temporal and modal logic. In: *Handbook of Theoretical Computer Science, vol. B* (J. van Leeuwen, Ed.). Chap. 16, pp. 995–1072. Elsevier.

Heymann, M. and F. Lin (1998). Discrete-event control of nondeterministic systems. **43**(1), 3–17.

Raclet, J.B. and S. Pinchinat (2004). The control of non-deterministic systems : a logical approach. RR 1648. Irisa.

Riedweg, S. (2003). Logiques pour le controle d'automatismes discrets. PhD thesis. IRISA Rennes.

Riedweg, S. and S. Pinchinat (2003). Quantified mu-calculus for control synthesis. In: *Mathematical Foundations of Computer Science.* Bratislava, Slovak Republic. pp. 642–651.

Riedweg, S. and S. Pinchinat (2004). Maximally permissive controllers in all contexts. In: *Proc of 7th Workshop on Discrete Event Systems, WODES 2004.* Reims, France. pp. 283–288.

Riedweg, S. and S. Pinchinat (2005). You can always compute maximally permissive controllers under partial observation when they exist. In: *Proc. 2005 American Control Conference.*. Portland, Oregon.

Thomas, W. (1997). Languages, automata and logic. In: *Handbook of Formal Languages* (A. Salomaa and G. Rozenberg, Eds.). Vol. 3, Beyond Words. Springer-Verlag. Berlin.