MAXIMALLY PERMISSIVE CONTROLLERS IN ALL CONTEXTS

Stéphane Riedweg, Sophie Pinchinat

IRISA-INRIA, F-35042, Rennes, France
{sriedweg,pinchina}@irisa.fr

Abstract: We propose a logical formalism for the supervisory control specifications where the maximal permissiveness is explicitly handled, independently of the plant. The approach relies on a modal logic with quantified propositions and additionally provides a powerful machinery for decision and synthesis procedures.

Keywords: Supervision, Discrete-events systems, Optimal control, Temporal logic

1. INTRODUCTION

The control theory for discrete event systems was initiated by (Ramadge and Wonham, 1989), followed by (Thistle and Wonham, 1994) and many others. More recently, temporal logics specifications were considered by (Kupferman *et al.*, 2000) and (Arnold *et al.*, 2003) in order to solve a larger class of problems.

Typically, control problems require to find a supervision of the plant such that its (operational) behavior meets some desired properties, often called *the control objectives*. In regard to the objectives, they most often concern standard properties such as *admissibility*, *non-blocking*, *safety* etc. Among those, the fact that the controllers should allow any sound event, hence fulfilling the *maximal permissiveness property*, is generally taken for granted. As studied in depth by (Ramadge and Wonham, 1989) and others, the theory of regular languages offers clear results on the subject: the existence of a controller implies the existence of a unique maximally permissive one. The proposed algorithms hence synthesize precisely this optimal solution. Unfortunately, this cozy situation does not extends to larger frameworks: for example, the ω-closure assumption is necessary for ω-regular definable objectives (as in (Thistle and Wonham, 1994)), the set of control patterns of (Golaszewski and Ramadge, 1987) must be union-closed, etc. Worse is the case of branching-time logic specifications as in (Kupferman *et al.*, 2000; Arnold *et al.*, 2003) where neither the unicity and nor the existence of a maximally permissive controller are guaranteed anymore. For instance, there is no optimal supervision of the plant $(a + b)^\omega$ which eventually disables event b. Surprisingly, to our knowledge, the maximal permissiveness property has never been approached on its own, like a possible parameter of the supervisory control problems, likely because regular language semantics withstand this subject.

In this paper, we propose a logical formalism which allows an explicit handling of maximally permissiveness, in addition to standard supervisory control specifications. The approach relies on a logical framework and provides a powerful machinery for decision procedures and effective synthesis.

The logic is an extension of the propositional mu-calculus (Kozen, 1983; Arnold and Niwinski, 2001), called the *Quantified mu-calculus* (QL_μ), as originally proposed by (Riedweg and Pinchinat, 2003*a*). The extension feature relies on the use of quantifications over (atomic) propositions. The resulting logic remains decidable as well as its model-checking thanks to its equivalence with infinite tree-automata, as already the case for the mu-calculus (see (Emerson and Lei, 1986)). For this reason, effective procedures to synthesis supervisory controllers are made possible by computing a finite model of a (infinite) tree-automaton (see also (Riedweg, 2003)).

In this paper, we do not aim at explaining the synthesis procedure, but instead we shall take it for granted from (Riedweg and Pinchinat, 2003*a*). Rather, we demonstrate how the logical formalism elegantly specifies supervisory control problems with an explicit mention of the *maximal permissiveness property* inside. Moreover, we make a proof that no other framework known in the literature can reach this expressive power.

The paper is organized as follows : Section 2 introduces the logic $\mathrm{Q}L_\mu$. Section 3 explains how control objectives can be specified. Next, Section 4 dedicates to the maximal permissiveness specification with a proof of correctness. In Section 5, expressiveness issues are discussed, followed by a section of concluding remarks.

2. QUANTIFIED MU-CALCULUS

Models are deterministic finite state machines with marked states, called *processes* in Definition 2, as plants in supervisory control problems normally are. Since branching-time temporal statements will be considered, the natural notion of behavior is the execution tree, namely the (possibly infinite) unfolding of the finite state machine. Given that controllers aim at pruning the execution tree, the proposed logic can state in particular the existence of some pruning of the tree that satisfies a (mu-calculus definable) desired property.

The prunings are simply represented by placing a fresh atomic proposition on the tree to delimit the remaining subtree after pruning: in this way, an edge of the tree where the source node is positively labeled by the proposition whereas the target node is not is meant to be pruned.

Technically, an atomic proposition p is added to the unfolding of a process by composing the process with a particular one, namely a *p-labeling process* $\mathcal{E}$ as in Definition 3; controllers are actually derived from those. To get the tree pruning induced from the proposition done, we prune the labeling process according to Definition 5 to get $\mathcal{E}_{(\to p)}$, before we compose it with the system.

So that, as stated by Proposition 7, wondering whether a given pruning/control $\mathcal{S} \times \mathcal{E}_{(\to p)}$ of the system verifies some desired property α reduces to wonder whether the corresponding p-labeling by $\mathcal{S} \times \mathcal{E}$ is appropriated satisfies some adjustment $\alpha{*}p$ of the original property.

We assume given a finite set of events $\Sigma = \{a, b, \dots\}$, a set of atomic propositions $AP = \{p, p', c, c', \dots\}$, and a set of variables $Var = \{X, Y, \dots\}$.

Definition 1. **(Syntax of $\mathrm{Q}L_\mu$)** We first recall the syntax of the pure mu-calculus, written L_μ. The set of formulas of L_μ is defined by the following grammar:

$$\top \mid p \mid X \mid \neg\beta \mid <a>\beta \mid \beta \vee \beta' \mid \mu X.\beta(X)$$

where $a \in \Sigma$, $p \in AP$ and $X \in Var$.

Fix-points formulas $\mu X.\beta(X)$ can properly be interpreted (in Definition 4) whenever each occurrence of X in $\beta(X)$ is under an even number of negation symbols $\neg$.

The quantified mu-calculus, written $\mathrm{Q}L_\mu$, extends L_μ as follows. The set of formulas of $\mathrm{Q}L_\mu$ is defined by:

$$\exists p.\alpha \mid \neg\alpha \mid \alpha \vee \alpha' \mid \beta$$

where $p \in AP$ and $\beta \in L_\mu$.

Freely extending the classical terminology of the mu-calculus to the quantified mu-calculus, we name *sentences* all the formulas where each occurrence of a variable X is binded by a fix-point symbol μ. Also, we write $\bot$, $[a]\alpha$, $\alpha \wedge \alpha'$, $\nu X.\alpha(X)$, and $\forall p.\alpha$ respectively for $\neg\top$, $\neg <a>\neg\alpha$, $\neg(\neg\alpha \vee \neg\alpha')$, $\neg\mu X.\neg\alpha(\neg X)$ and $\neg\exists p.\neg\alpha$, as well as $\xrightarrow{a}$, $[\,]\alpha$ and $\alpha \Rightarrow \alpha'$ respectively for $<a>\top$, $\bigwedge_{a\in\Sigma}[a]\alpha$ and $\neg\alpha \vee \alpha'$.

Lastly, for $\beta \in L_\mu$, $\mathtt{INV}(\beta)$ is a notation for $\nu X.[\,]X \wedge \beta$: according to Definition 4 further, it states “from now on, the property β always holds”.

Since in general, fixed-point operators and quantifiers do not commute, we forbid the use of quantifications inside fixed-point terms, all the more since the formulas will have enough expressive power for control problems specifications, as shown in the next section.

The semantics of the quantified mu-calculus (Definition 4) relies on models called *processes*:

Definition 2. **(Processes)** Given a finite set $\Gamma \subseteq AP$, a *process on* Γ is a tuple $\mathcal{S} = \langle S, s^0, t, L\rangle$, where S is a set of states, $s^0 \in S$ is the initial state, $t : S \times \Sigma \to S$ is a partial function called the *transition function* and $L : S \to 2^\Gamma$ labels states by propositions.

A process $\mathcal{S}$ is *finite* if S is finite and it is *complete* if $t(s, a)$ is defined everywhere.

Processes are normally subject to a natural abstraction according to their execution tree isomorphism class. Two processes with isomorphic execution trees are called *bisimilar*.

As announced, processes are composed in a synchronous manner, as to diminish their behavior: The *(synchronous) product* of $\mathcal{S}_1 = \langle S_1, s_1^0, t_1, L_1\rangle$ on Γ_1 and $\mathcal{S}_2 = \langle S_2, s_2^0, t_2, L_2\rangle$ on Γ_2 (with disjoint Γ_1 and Γ_2) is $\mathcal{S}_1 \times \mathcal{S}_2 = \langle S_1 \times S_2, (s_1^0, s_2^0), t, L\rangle$ on $\Gamma_1 \cup \Gamma_2$ where:

(1) $t((s_1, s_2), a) = (s_1', s_2')$ whenever $s_1' = t_1(s_1, a)$ and $s_2' = t_2(s_2, a)$), and
(2) $L(s_1, s_2) = L_1(s_1) \cup L_2(s_2)$.

Definition 3. **(Labeling Processes)** Given $p \in AP$, a *p-labeling process* is a complete process on $\{p\}$. We let Lab_p be the set of p-labeling processes, and we use $\mathcal{E}$, $\mathcal{E}'$ for typical elements. A *labeling of $\mathcal{S}$ by p*, or a *p-labeling of $\mathcal{S}$*, is a product $\mathcal{S} \times \mathcal{E}$, where $\mathcal{E} \in Lab_p$.

The logic can now be interpreted. Given a process $\mathcal{S}$, a formula α denotes a subset of the states, those which "satisfy" it. The semantics is given by induction over α, hence the need to interpret variable formulas: the standard manner consists in setting a valuation $val : Var \to \mathcal{P}(S)$ which defines the subset of states denoted by the formulas $X \in Var$. The pure mu-calculus formulas semantics is standard.

Definition 4. **(Semantics of $\mathrm{Q}L_\mu$)** Given a process $\mathcal{S} = \langle S, s^0, t, L\rangle$ and $val : Var \to \mathcal{P}(S)$, the interpretation of a $\mathrm{Q}L_\mu$-formula α is

$$[\![\, \alpha \,]\!]_{\mathcal{S}}^{[val]} \subseteq S$$

defined inductively by:

$$\begin{array}{l}
[\![\, \top \,]\!]_{\mathcal{S}}^{[val]} = S, \qquad [\![\, p \,]\!]_{\mathcal{S}}^{[val]} = \{s \in S \,|\, p \in L(s)\}, \\
[\![\, X \,]\!]_{\mathcal{S}}^{[val]} = val(X),\ [\![\, \neg\alpha \,]\!]_{\mathcal{S}}^{[val]} = S \setminus [\![\, \alpha \,]\!]_{\mathcal{S}}^{[val]}, \\
[\![\, \alpha \vee \alpha' \,]\!]_{\mathcal{S}}^{[val]} = [\![\, \alpha \,]\!]_{\mathcal{S}}^{[val]} \cup [\![\, \alpha' \,]\!]_{\mathcal{S}}^{[val]}, \\
[\![\, \langle a \rangle\alpha \,]\!]_{\mathcal{S}}^{[val]} = \{s \in S \,|\, \exists s', t(s,a) = s', s' \in [\![\, \alpha \,]\!]_{\mathcal{S}}^{[val]}\}, \\
[\![\, \mu X.\alpha(X) \,]\!]_{\mathcal{S}}^{[val]} = \bigcap\{V \subseteq S \,|\, [\![\, \alpha \,]\!]_{\mathcal{S}}^{[val(V/X)]} \subseteq V\},
\end{array}$$

and $[\![\exists p.\alpha]\!]_{\mathcal{S}}^{[val]}$ is the set of states $s \in S$, for which there exists $\mathcal{E} = \langle E, \varepsilon^0, t', L'\rangle \in Lab_p$, such that

$$(s, \varepsilon^0) \in [\![\, \alpha \,]\!]_{\mathcal{S}\times\mathcal{E}}^{[val']}$$

where $val'(X) \subseteq S \times E$ is $val(X) \times E$, in order to be sound.

Since the interpretation of a sentence α is independent of the valuation val, we simply write $[\![\, \alpha \,]\!]_{\mathcal{S}}$, and we use $\mathcal{S} \models \alpha$ whenever the initial state of $\mathcal{S}$ belongs to $[\![\, \alpha \,]\!]_{\mathcal{S}}$. To sum up, the assertions "$\mathcal{S} \models \alpha_1 \wedge \alpha_2$", "$\mathcal{S} \models \neg\alpha$", ... respectively mean "$\mathcal{S} \models \alpha_2$ and $\mathcal{S} \models \alpha_2$", "not $\mathcal{S} \models \alpha$" (also written "$\mathcal{S} \not\models \alpha$"),... Finally, "$\mathcal{S} \models \exists p.\alpha$" expresses that "there exists a p-labeling process $\mathcal{E}$ such that $\mathcal{S} \times \mathcal{E} \models \alpha$".

Apart from formulas of the form $\exists p.\alpha$, the semantics is the classical mu-calculus definition.

Because the mu-calculus is bisimulation invariant (formulas have the same truth value when interpreted over bisimilar processes), so is the logic $\mathrm{Q}L_\mu$ since the synchronous product which the interpretation of $\exists p.\alpha$ relies on preserves the execution tree isomorphisms.

In the rest of this section we establish how to adjust $\mathrm{Q}L_\mu$ formulas according to a fixed proposition when the latter is used to prune the model. We first formalize what the pruning is.

Definition 5. **(Pruning)** Given any process $\mathcal{S} = \langle S, s^0, t, L\rangle$ on Γ and $p \in \Gamma$, the *p-pruning of $\mathcal{S}$* is $\mathcal{S}_{(\to p)} = \langle S, s^0, t', L'\rangle$ on $\Gamma \setminus \{p\}$, where:

(1) for all $s \in S$ and $a \in \Sigma$, $t'(s,a) = t(s,a)$ if $p \in L(t(s,a))$, otherwise $t'(s,a)$ is undefined, and
(2) $L'(s) = L(s) \setminus \{p\}$.

Definition 6. **(Adjustment)** For any sentence $\alpha \in \mathrm{Q}L_\mu$ and for any $p \in AP$, the *p-adjustment of α* is $\alpha * p \in \mathrm{Q}L_\mu$, inductively defined by (by convention, $*$ has highest priority) :

$$\begin{array}{l}
\top * p = \top, \quad p' * p = p', \quad X * p = X, \\
(\langle a \rangle\alpha) * p = \langle a \rangle(p \wedge \alpha * p),\ (\neg\alpha) * p = \neg\alpha * p, \\
(\alpha \vee \alpha') * p = \alpha * p \vee \alpha' * p,\ \ (\mu X.\alpha) * p = \mu X.\alpha * p, \\
(\exists p.\alpha) * p = \exists p.\alpha * p.
\end{array}$$

Definition 5 and 6 together lead to the following fundamental result:

Proposition 7. Given a sentence $\alpha \in \mathrm{Q}L_\mu$, $\Gamma \subseteq AP$, $p \notin \Gamma$, a process $\mathcal{S}$ on Γ and $\mathcal{E} \in Lab_p$, we have:

$$\mathcal{S} \times (\mathcal{E}_{(\to p)}) \models \alpha \text{ if and only if } \mathcal{S} \times \mathcal{E} \models \alpha * p.$$

PROOF. Since, the processes $\mathcal{S} \times (\mathcal{E}_{(\to p)})$ and $(\mathcal{S} \times \mathcal{E})_{(\to p)}$ are isomorphic, they must satisfy the same formulas in each isomorphic states: $[\![\, \alpha \,]\!]_{\mathcal{S}\times(\mathcal{E}_{(\to p)})} = [\![\, \alpha \,]\!]_{(\mathcal{S}\times\mathcal{E})_{(\to p)}}$. An easy proof by induction on α gives the following more general result to conclude: $[\![\, \alpha \,]\!]_{(\mathcal{S}\times\mathcal{E})_{(\to p)}} = [\![\, \alpha * p \,]\!]_{\mathcal{S}\times\mathcal{E}}$.

As shown by Proposition 7, objects of type $\mathcal{E}_{(\to p)}$ indicate what controllers should be like. The soundness of the coming logical specifications can be argued by using two kinds of binary relations between processes. The first kinds are equivalence relations: given a process $\mathcal{S}$, two p-labeling processes $\mathcal{E}_1$ and $\mathcal{E}_2$ will be equivalent if they cause the same p-labeling on $\mathcal{S}$. They will be used in the next section.

Definition 8. **(The $\equiv_{\mathcal{S}}$-equivalences)** Given a process $\mathcal{S}$ and two c-labeling processes $\mathcal{E}_1$ and $\mathcal{E}_2$, *$\mathcal{E}_1$ and $\mathcal{E}_2$ induce the same c-labeling on $\mathcal{S}$*, written $\mathcal{E}_1 \equiv_{\mathcal{S}} \mathcal{E}_2$, whenever $\mathcal{S}\times\mathcal{E}_1$ and $\mathcal{S}\times\mathcal{E}_2$ are bisimilar.

The next binary relation between processes we consider here is the preorder of *simulation*; its main use further on is to compare more or less permissive behaviors.

Definition 9. **(Simulation)** Given two processes $\mathcal{S}_1 = \langle S_1, s_1^0, t_1, L_1\rangle$ and $\mathcal{S}_2 = \langle S_2, s_2^0, t_2, L_2\rangle$, a *simulation of* $\mathcal{S}_1$ *by* $\mathcal{S}_2$ is a binary relation $R \subseteq S_1 \times S_2$ such that $(s_1^0, s_2^0) \in R$ and, for any $(s_1, s_2) \in R$ and for any $a \in \Sigma$, we have: if $t_1(s_1, a)$ is defined, then $t_2(s_2, a)$ is defined and $(t(s_1, a), t_2(s_2, a)) \in R$.
We write $\mathcal{S}_1 \preceq \mathcal{S}_2$ whenever there exists a simulation of $\mathcal{S}_1$ by $\mathcal{S}_2$. Basically, $\mathcal{S}_1 \preceq \mathcal{S}_2$ tells that $\mathcal{S}_1$ has more executions than $\mathcal{S}_2$.

One easily checks that $\preceq$ is a preorder. Note that if two processes $\mathcal{S}_1$ and $\mathcal{S}_2$ are bisimilar, then $\mathcal{S}_1 \preceq \mathcal{S}_2$ and $\mathcal{S}_2 \preceq \mathcal{S}_1$; the relation $\preceq$ ignoring the atomic propositions, the reciprocal does not hold in general. We let $\mathcal{S}_1 \prec \mathcal{S}_2$ mean $\mathcal{S}_1 \preceq \mathcal{S}_2$ and not $\mathcal{S}_2 \preceq \mathcal{S}_1$.

We now establish two useful technical lemmas.

Lemma 10. For any process $\mathcal{S}$, we have: $\mathcal{S} \preceq \mathcal{S}_{(\to c)}$ if and only if $\mathcal{S} \models [\,]\mathtt{INV}\,(p)$.

PROOF. Let S and S' be respectively the set of states of $\mathcal{S}$ and $\mathcal{S}_{(\to p)}$. Hence $S' \subseteq S$. Actually one proves that $\mathcal{S} \preceq \mathcal{S}_{(\to p)}$ by check the simulation $\{(s, s) \in S' \times S'\}$.

Lemma 11. For any p-labeling process $\mathcal{E}$: $\mathcal{S} \preceq \mathcal{S} \times \mathcal{E}_{(\to p)}$ if and only if $\mathcal{S} \times \mathcal{E} \models [\,]\mathtt{INV}\,(p)$.

PROOF. By Lemma 10 for $\mathcal{S} \times \mathcal{E}$, $\mathcal{S} \times \mathcal{E} \models [\,]\mathtt{INV}\,(p)$ if and only if $\mathcal{S} \times \mathcal{E} \preceq (\mathcal{S} \times \mathcal{E})_{(\to p)}$. Since $\mathcal{E}$ is complete, $\mathcal{S} \preceq \mathcal{S} \times \mathcal{E}$ and because $(\mathcal{S} \times \mathcal{E})_{(\to p)}$ is bisimilar to $\mathcal{S} \times \mathcal{E}_{(\to p)}$, we obtain $\mathcal{S} \preceq \mathcal{S} \times \mathcal{E} \preceq (\mathcal{S} \times \mathcal{E})_{(\to p)} \preceq \mathcal{S} \times \mathcal{E}_{(\to p)}$. By transitivity, $\mathcal{S} \preceq \mathcal{S} \times \mathcal{E}_{(\to p)}$. The reciprocal is easy.

3. BASIC CONTROLLERS

The current section shows how basic controllers can be specified in the logic $\mathrm{Q}L_\mu$. The correctness of the specifications is also given.

¿From now on, we assume given $\Gamma \subseteq AP$ and a process $\mathcal{P}$ on Γ which represents a plant: usually, the set of events Σ divides between *uncontrollable* events $\Sigma_{uc} \subseteq \Sigma$ on the one hand and *controllable* events, by complementary, on the other hand.

We suppose that the reader is familiar with the notion of *controller of* $\mathcal{P}$ *for some control objective*, as proposed in the reference work of (Ramadge and Wonham, 1989). In our setting, since the control objectives are mu-calculus, and even quantified mu-calculus, definable properties, we agree that *a controller of* $\mathcal{P}$ *for* $\alpha \in \mathrm{Q}L_\mu$ is a process $\mathcal{C}$ with the following:

(1) at any time, it allows all uncontrollable events:
$\mathcal{C} \models \mathtt{Admi} \stackrel{\text{def}}{=} \mathtt{INV}\,(\bigwedge_{u\in\Sigma_{uc}} \stackrel{u}{\to})$, and
(2) it achieves the objectives inasmuch as:
$\mathcal{P} \times \mathcal{C} \models \alpha$.

In the following, we fix a fresh atomic proposition c and we let the following mu-calculus formula:

$$\mathtt{Controller}(c) \stackrel{\text{def}}{=} c \wedge \mathtt{INV}\,((\neg c) \Rightarrow \mathtt{INV}\,(\neg c)) \wedge \mathtt{INV}\,(c \Rightarrow \bigwedge_{u\in\Sigma_{uc}} [u]c)$$

$\mathtt{Controller}(c)$ is interpreted on a c-labeling of $\mathcal{P}$, where c represents reachable states under control: a c-labeling of $\mathcal{P}$ satisfies $\mathtt{Controller}(c)$ whenever, under control,

(1) the initial state is reachable, since c-labeled,
(2) successor states of unreachable states are neither reachable, and
(3) the successors of reachable states via uncontrollable events remain reachable.

Proposition 12. For any $\mathcal{E} \in Lab_c$ s.t. $\mathcal{P} \times \mathcal{E}$ satisfies $\mathtt{Controller}(c)$, there exists $\mathcal{E}' \in Lab_c$ which defines a control, in the sense that $\mathcal{E}'_{(\to^c)} \models \mathtt{Admi}$, and s.t. $\mathcal{E}' \equiv_{\mathcal{P}} \mathcal{E}$.

PROOF. Let $\mathcal{E} = \langle E, e^0, t, L\rangle \in Lab_c$ and let us assume that $\mathcal{P} \times \mathcal{E} \models \mathtt{Controller}(c)$. Define $\mathcal{E}' \in Lab_c$ by $\mathcal{E}' = \langle E, e^0, t', L\rangle$ where

$$t'(e,a) = \begin{cases} e & \text{if } (a \in \Sigma_{uc}) \text{ and } (c \in L(e)) \\ & \quad\text{and } (c \notin L(t(e,a))) \\ t(e,a) & \text{otherwise} \end{cases}$$

By construction, $\mathcal{E}' \models c \wedge \mathtt{INV}\,(c \Rightarrow \bigwedge_{u\in\Sigma_{uc}}[u]c)$, hence $\mathcal{E}'_{(\to^c)} \models \mathtt{INV}\,(\bigwedge_{u\in\Sigma_{uc}} \stackrel{u}{\to})$. The fact that $t'(e,a)$ differs from $t(e,a)$ only when $a \in \Sigma_{uc}$, $c \in L(e)$ and $c \notin L(t(e,a))$, and since by assumption, $\mathcal{P} \times \mathcal{E} \models \mathtt{INV}\,(\bigwedge_{u\in\Sigma_{uc}} c \Rightarrow [u]c)$, we conclude that $\mathcal{E}' \equiv_{\mathcal{P}} \mathcal{E}$.

Propositions 7 and 12 together lead to Theorem 13 below, provided we agree upon the following:

$$\mathtt{Solution}(c, \alpha) \stackrel{\text{def}}{=} \alpha * c \wedge \mathtt{Controller}(c)$$

Theorem 13. For any $\alpha \in \mathrm{Q}L_\mu$,

there exists a controller of $\mathcal{P}$ for α
if and only if

$$\mathcal{P} \models \exists c.\mathtt{Solution}(c, \alpha) \qquad (1)$$

PROOF. The specification above is correct as we show now: assume there is an controller $\mathcal{C}$ for α. We will define $\mathcal{E} \in Lab_c$ such that $\mathcal{E}_{(\to^c)}$ and $\mathcal{C}$ are isomorphic: we assign c to each state of $\mathcal{C}$ and we complete the resulting process with a sink state. Then $\mathcal{P} \times \mathcal{E}$ satisfies $c \wedge \mathtt{INV}((\neg c) \Rightarrow \mathtt{INV}(\neg c))$ and since $\mathcal{C}$ is a controller, $\mathcal{P}\times\mathcal{E}$ satisfies $\mathtt{INV}(c \Rightarrow \bigwedge_{u\in\Sigma_{uc}}[u]c)$. Since $\mathcal{C}$ is a controller for α on $\mathcal{P}$ and since $\mathcal{E}_{(\to^c)}$ and $\mathcal{C}$ are isomorphic, by Proposition 7, $\mathcal{P} \times \mathcal{E}$ satisfies $\alpha{*}c$.

Assume now that $\mathcal{P}$ satisfies the formula (1). Then there exists $\mathcal{E} \in Lab_c$ such that $\mathcal{P} \times \mathcal{E}$ satisfies $(\alpha * c)$ and $\mathtt{Controller}(c)$. Since $\mathcal{P} \times \mathcal{E} \models \mathtt{Controller}(c)$, there exists (Proposition 12) a labeling process $\mathcal{E}'$ such that $\mathcal{E}' \equiv_{\mathcal{P}} \mathcal{E}$ and $\mathcal{E}'_{(\to c)}$ is an controller. Moreover, $\mathcal{E}' \equiv_{\mathcal{P}} \mathcal{E}$ and $\mathcal{P} \times \mathcal{E} \models \alpha{*}c$ imply $\mathcal{P} \times \mathcal{E}' \models \alpha{*}c$. Then, take $\mathcal{E}'_{(\to c)}$ to conclude (by Proposition 7).

4. MAXIMALLY PERMISSIVE CONTROLLERS

We first give the definition for *Maximal Permissiveness* in a fully natural way. Next we specify it in $\mathrm{Q}L_\mu$, with the arguments for its correctness.

Definition 14. **(Maximally Permissive Controller)** Given a formula $\alpha \in \mathrm{Q}L_\mu$, we say that the controller $\mathcal{C}$ of $\mathcal{P}$ for α is *maximally permissive* if for any other controller $\mathcal{C}'$ of $\mathcal{P}$ for α, we do not have $\mathcal{S} \times \mathcal{C} \prec \mathcal{S} \times \mathcal{C}'$.

Essentially, since the labelings of the plant can be compared regarding the size of the remaining subtree after pruning, so can the underlying controllers, hence our capability to express optimality in the logic: given any two fresh atomic propositions c and c', we define the formulas $c \sqsubseteq c'$ and $c \sqsubset c'$ by:

$$c \sqsubseteq c' \stackrel{\text{def}}{=} ([\,]\mathtt{INV}(c')){*}c$$

$$c \sqsubset c' \stackrel{\text{def}}{=} c \sqsubseteq c' \wedge \neg(c' \sqsubseteq c)$$

Formula $c \sqsubseteq c'$ contains a c-adjustment to express that c' holds all along c-labeled executions.

Proposition 15. For all $\mathcal{E}, \mathcal{E}' \in Lab_c$, we have:

$$\mathcal{P} \times \mathcal{E}_{(\to c)} \preceq \mathcal{P} \times \mathcal{E}'_{(\to c')} \text{ iff } \mathcal{P} \times \mathcal{E} \times \mathcal{E}' \models c \sqsubseteq c'$$

PROOF. Assume that $\mathcal{P}\times\mathcal{E}\times\mathcal{E}'$ satisfies $c \sqsubseteq c'$. By Proposition 7, this is equivalent to say that $\mathcal{P} \times \mathcal{E}' \times \mathcal{E}_{(\to c)}$ satisfies $[\,]\mathtt{INV}(c')$. Applying now Lemma 11 for $\mathcal{P} \times \mathcal{E}_{(\to c)}$ leads to $\mathcal{P} \times \mathcal{E}_{(\to c)} \preceq \mathcal{P} \times \mathcal{E}_{(\to c)} \times \mathcal{E}'_{(\to c')}$. Because clearly $\mathcal{P} \times \mathcal{E}_{(\to c)} \times \mathcal{E}'_{(\to c')} \preceq \mathcal{P} \times \mathcal{E}'_{(\to c')}$, we conclude by transitivity. Reciprocally, assume $\mathcal{P} \times \mathcal{E}_{(\to c)} \preceq \mathcal{P} \times \mathcal{E}'_{(\to c')}$; by composing both terms with $\mathcal{E}_{(\to c)}$, we obtain $\mathcal{P} \times \mathcal{E}_{(\to c)} \preceq \mathcal{P} \times \mathcal{E}'_{(\to c')} \times \mathcal{E}_{(\to c)}$. Now similar arguments to previous ones can be put forward to conclude.

Theorem 16. For any $\alpha \in \mathrm{Q}L_\mu$, there exists a maximally permissive controller of $\mathcal{P}$ for α if and only if

$$\begin{aligned}\mathcal{P} \models \exists c.\ &\mathtt{Solution}(c,\alpha)\ \wedge \\ &\forall c'.[\, c \sqsubset c' \Rightarrow \neg\mathtt{Solution}(c',\alpha)\,]\end{aligned} \quad (2)$$

PROOF. $\Leftarrow$) By Theorem 13 and because $\mathcal{P} \models \exists c.\mathtt{Solution}(c,\alpha)$, there exists $\mathcal{E} \in Lab_c$ s.t. $\mathcal{E}_{(\to c)}$ is a controller on $\mathcal{P}$ for α and

$$\mathcal{P} \times \mathcal{E} \models \forall c. c \sqsubset c' \Rightarrow \neg\mathtt{Solution}(c',\alpha) \quad (3)$$

Equation 3 with Proposition 15 means for all process $\mathcal{E}' \in Lab_{c'}$, if $\mathcal{P} \times \mathcal{E}_{(\to c)} \prec \mathcal{P} \times \mathcal{E}'_{(\to c')}$ then $\mathcal{P} \times \mathcal{E} \times \mathcal{E}' \not\models \mathtt{Solution}(c',\alpha)$. Now, because $\mathcal{E}$ is complete and since c does not occur in $\mathtt{Solution}(c',\alpha)$, we can assert: $\mathcal{P} \times \mathcal{E}' \not\models \mathtt{Solution}(c',\alpha)$. By Theorem 13, $\mathcal{E}'_{(\to c')}$ necessarily cannot be a controller of $\mathcal{P}$ for α. By Definition 9, $\mathcal{E}_{(\to c)}$ is a maximally permissive controller of $\mathcal{P}$ for α.

$\Rightarrow$) Assume a maximally permissive controller for α on $\mathcal{P}$, say $\mathcal{C}$. Let us label all states of $\mathcal{C}$ by c, and complete the result to get some $\mathcal{E} \in Lab_c$ s.t. $\mathcal{E}_{(\to c)}$ equals $\mathcal{C}$. By Theorem 13 (and actually its proof arguments), $\mathcal{P} \times \mathcal{E} \models \mathtt{Solution}(c,\alpha)$. Moreover, since $\mathcal{C}$ is maximally permissive, for all $\mathcal{E}' \in Lab_{c'}$, we have $\mathcal{P} \times \mathcal{E} \times \mathcal{E}' \models \mathtt{Solution}(c',\alpha) \Rightarrow \neg(c \sqsubset c')$. Which concludes.

5. EXPRESSIVENESS

We focuse now on Formula (2), where α is a parameter, and we partially answer expresiveness. Recall that Formula (2) represents a new kind of specification which expresses the maximal permissiveness requirement for controllers.

Theorem 17. Let α be formula in L_μ. **There cannot exist** formulas β and γ in L_μ, or even in $\mathrm{Q}L_\mu$, such that for all plant $\mathcal{P}$:

$\mathcal{C}$ is a maximally permissive controller of $\mathcal{P}$ for α
if and only if
$\mathcal{C} \models \gamma$ and $\mathcal{P} \times \mathcal{C} \models \beta$

PROOF. Take $\alpha = \mu X.[\,]X \vee p$, written $\mathbf{AF}p$ in the following, as in the classic logic CTL. $\mathbf{AF}p$ tells that along any execution, the proposition p will eventually hold.

Assume some pair (β,γ) exists. Take $\mathcal{P}$ as in Fig.1, where the two events a and b of Σ are both controllable. Clearly, the single state process with an a-loop is a maximally permissive controller for $\mathbf{AF}p$. By assumption, $\mathcal{C} \models \gamma$ and $\mathcal{P} \times \mathcal{C} \models \beta$.

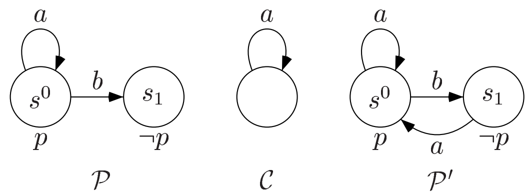

Fig. 1. $\mathcal{P}$, the controller $\mathcal{C}$ and $\mathcal{P}'$

Now define $\mathcal{P}'$ as $\mathcal{P}$ plus an a-transition back to the initial state, as drawn in Figure 1. Because $\mathcal{P} \times \mathcal{C}$ and $\mathcal{P}' \times \mathcal{C}$ are bisimilar, we also have $\mathcal{P}' \times \mathcal{C} \models \beta$. Since by assumption $\mathcal{C} \models \gamma$, $\mathcal{C}$ is a solution of (β, γ) and yet $\mathcal{C}$ is not a maximally permissive controller of $\mathcal{P}'$ for $\mathbf{AF}p$: indeed, $\mathcal{C}$ cuts the transition from s^0 to s_1 of $\mathcal{P}'$, which should be kept. This leads to a contradiction.

As a consequence of Theorem 17, while with the logic $\mathrm{Q}L_\mu$ the maximal permissiveness is expressed in a uniform manner, its statement in the framework of (Arnold *et al.*, 2003) must be designed according to the plant, hence an ad hoc solution.

6. CONCLUSION

The presented work proposes a logical framework for the specification of control problems, which additionally allows to perform synthesis. The key ideas are built on an extension of the propositional mu-calculus using quantified atomic propositions. The resulting is expressive enough to specify maximally permissive controllers for any mu-calculus definable control objective, independently of the plant. Moreover, we showed that other existing logical frameworks, like (Arnold *et al.*, 2003), cannot express maximal permissiveness on its own, but would rather need a system-dependent specification. For the sake of simplicity, we have only proposed here a fully observational setting, but the results still hold in the more general case of partial observation, as explained in (Riedweg and Pinchinat, 2003*b*), as well in a forthcoming paper.

As explained in the paper, the maximal permissiveness property is definitely a strong argument in favor of our approach and demonstrates a fundamental feature of the quantified logic framework. Indeed, contrary to other approaches, it is possible to express properties outside the remaining system after the synchronous product of the plant with the controller. As a consequence, many other kinds of optimality criteria can be considered, such as the unicity of a solution, but also, "contextual" properties: one can specify that deadlocks are forbidden unless they already exist in the plant, priority between events, fairness of the control, forcing events, etc.

REFERENCES

Arnold, A. and D. Niwinski (2001). *Rudiments of mu-calculus*. North-Holland.

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

Emerson, E. A. and C. Lei (1986). Efficient model checking in fragments of the propositional mu-calculus. In: *Proc. 1st IEEE Symp. Logic in Computer Science, Cambridge, Massachussets.*. pp. 267–278.

Golaszewski, C. H. and P. J. Ramadge (1987). Control of discrete event systems with forced events. In: *Proc. of 26th Conf. Decision and Control*. Los Angeles, CA, USA. pp. 247–251.

Kozen (1983). Results on the propositional μ-calculus. *Theoretical Computer Science* **27(3)**, 333–354.

Kupferman, O., P. Madhusudan, P. S. Thiagarajan and M. Y. Vardi (2000). Open systems in reactive environments: Control and synthesis. In: *Proc. 11th Int. Conf. on Concurrency Theory*. Vol. 1877 of *LNCS*. Springer-Verlag. pp. 92–107.

Ramadge and Wonham (1989). The control of discrete event systems. In: *Proc. of the IEEE*. Vol. 77(1). pp. 81–98.

Riedweg (2003). Logiques pour le Contrôle d'Automatismes Discrets. PhD thesis. Université de Rennes I.

Riedweg and Pinchinat (2003*a*). Quantified mu-calculus for control synthesis. In: *Mathematical Foundations of Computer Science*. Vol. 2747 of *LNCS*. pp. 642–651.

Riedweg, S. and S Pinchinat (2003*b*). Quantified loop mu-calculus for control synthesis under partial observation. Technical Report R4949. Irisa.

Thistle and Wonham (1994). Control of of infinite behavior of finite automata. *SIAM Control and Optimization* **32(4)**, 1075–1097.