

# Quotient-based Control Synthesis for Partially Observed Non-Deterministic Plants with Mu-Calculus Specifications

Samik Basu and Ratnesh Kumar, Fellow IEEE

**Abstract**—We study the control of a nondeterministic plant subject to a specification expressed in the propositional  $\mu$ -calculus under a partial observability of events. We define a function to *quotient* the specification against the plant resulting in a “quotiented formula” with the property that a supervisor enforcing the desired specification exists if and only if the quotiented formula is satisfiable, and a model witnessing the satisfiability can be used as a supervisor. The quotiented formula belongs to an extended  $\mu$ -calculus, which we call  $\mathcal{O}$ - $\mu$ -calculus, where the extension is needed to express the observability constraint that cannot be expressed in the logic of  $\mu$ -calculus. We present the syntax and semantics of  $\mathcal{O}$ - $\mu$ -calculus and present a tableau-based satisfiability solving algorithm that also discovers a model for the quotiented formula when one exists.

## I. INTRODUCTION

Supervisory control problem for discrete event systems involves computing a supervisor (if one exists) such that when composed with a given plant, the composed system conforms to a desired specification. The problem of control of a deterministic plant subject to a  $\mu$ -calculus specification was first addressed by Arnold, Vincent, and Walukiewicz in [3], which led to subsequent extensions in [2], [6].

In this paper, extending our past work [4], we address the same problem with the generalization to include (a) *nondeterminism* in the plant model with (b) *state-based* uncontrollability and unobservability constraints. Uncontrollability constraint defines events that cannot be disabled, whereas unobservability constraint defines events that are unobservable, and event-classes that are indistinguishable. The similarities between the approaches are that we both employ the technique of “quotienting”, and reduce the control problem to a satisfiability problem. The differences are as follow. Firstly, quotienting is performed at the level of  $\mu$ -calculus formulas in our setting and we absorb the controllability and observability requirements as part of quotienting. This lets us handle the plant nondeterminism and *state-based* controllability and observability requirements in a simple manner. On the other hand, in [3], [2], [6]  $\mu$ -calculus quotienting is performed at the level of their alternating tree automata representations and controllability and observability requirements are imposed separately on the quotiented formula. As a result, these techniques are only applicable for deterministic plants and can only consider controllability and observability requirements that are independent of plant states. Secondly, we incorporate the observability requirement using a simple extension of the  $\mu$ -calculus, referred to as  $\mathcal{O}$ - $\mu$ -calculus. The

The research was supported in part by the National Science Foundation under the grants NSF-CCF-0702758, NSF-CNS-0709217, NSF-ECS-0218207, NSF-ECS-0244732, NSF-EPNES-0323379, NSF-ECS-0424048, and NSF-ECS-0601570.

Samik Basu and Ratnesh Kumar are with Iowa State University, Dept. of Computer Science and Electrical & Computer Eng., respectively. Email: {sbasu, rkumar}@iastate.edu.

extension includes a new “observational” formula expression. Analogously, [3] introduced “loop- $\mu$ -calculus” extension to capture unobservable events; further extensions  $\mu$ -calculus (CONV-modality) and alternating tree automata were later proposed in [2] and [6] for capturing indistinguishable events. [12], [11] also presented an automata-theoretic quotienting approach for supervisor synthesis for enforcing  $\mu$ -calculus specifications. An extension of  $\mu$ -calculus, called *quantified  $\mu$ -calculus*, to express the existence of a supervisor as part of the specification logic was introduced. We avoid capturing the supervisor existence as part of the specification logic and delegate that to the satisfiability of the quotiented property. Furthermore, as with [3], [2], [6], the work in [11] is limited to deterministic plants.

Our quotienting procedure is closely related to the ones described in [1], [10], [5], where quotienting of equational  $\mu$ -calculus formula expressions against labeled transition systems and CCS processes is described for model checking systems with regular structures, real-time systems and parameterized systems respectively. Our quotienting definition is tailored for control application taking into consideration the controllability and observability requirements, and also any nondeterminism in the plant model.

The satisfiability of the quotiented formula is used for determining supervisor existence, and a model satisfying the quotiented formula serves as a supervisor. Typically satisfiability checking techniques rely on reducing the satisfiability problem to emptiness problem of alternating tree automata [7] or identifying a winning strategy in parity games [13], [3]. The authors in [8] developed an equivalent disjunctive form of a  $\mu$ -calculus formula for which satisfiability is easily verified. Drawing inspiration from [8], we developed our own tableau-based technique for satisfiability-checking and model-discovery of  $\mathcal{O}$ - $\mu$ -calculus formulas.

## II. PRELIMINARIES

**$\mu$ -calculus Specifications.** The  $\mu$ -calculus [9] is an expressive temporal logic with explicit greatest and least fixed point operators. Properties in temporal logics LTL, CTL, and CTL\* can be encoded as  $\mu$ -calculus formulas. The syntax of  $\mu$ -calculus formulas is defined over a domain  $(AP, \mathcal{X}, A)$  consisting of a set of atomic propositions  $AP$ , a set of formula variables  $\mathcal{X}$ , and a set of actions  $A$ , and is given by the grammar:

$$\phi \rightarrow \text{tt} \mid \text{ff} \mid p \mid X \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a \rangle \phi \mid [a] \phi \mid \sigma X. \phi.$$

Here,  $p \in AP, X \in \mathcal{X}, a \in A$ ; and  $\text{tt}$  and  $\text{ff}$  denote the formulas that are always “True” and “False” respectively;  $\langle a \rangle$  and  $[a]$  denote the “diamond” and “box” operators representing “exists  $a$ -successor” and “all  $a$ -successor” modalities respectively. We also use the notation  $\langle \! \langle \! \rangle \! \rangle$  to denote either  $\langle \! \rangle$  or  $[ \! ]$ .  $\sigma X. \phi$  represents a fixed point formula expression

1.  $\llbracket \text{tt} \rrbracket_e = Q$     2.  $\llbracket \text{ff} \rrbracket_e = \emptyset$
3.  $\llbracket p \rrbracket_e = \{q \mid p \in L(q)\}$     4.  $\llbracket X \rrbracket_e = e(X)$
5.  $\llbracket \varphi_1 \wedge \varphi_2 \rrbracket_e = \llbracket \varphi_1 \rrbracket_e \cap \llbracket \varphi_2 \rrbracket_e$     6.  $\llbracket \varphi_1 \vee \varphi_2 \rrbracket_e = \llbracket \varphi_1 \rrbracket_e \cup \llbracket \varphi_2 \rrbracket_e$
7.  $\llbracket \langle a \rangle \varphi \rrbracket_e = \{q \mid \exists q' \xrightarrow{a} q' \wedge q' \in \llbracket \varphi \rrbracket_e\}$
8.  $\llbracket [a] \varphi \rrbracket_e = \{q \mid \forall q' \xrightarrow{a} q' \Rightarrow q' \in \llbracket \varphi \rrbracket_e\}$
9.  $\llbracket \mu X. \varphi \rrbracket_e = \cap \{ \hat{Q} \mid \llbracket \varphi \rrbracket_{e[X \mapsto \hat{Q}]} \subseteq \hat{Q} \}$
10.  $\llbracket \nu X. \varphi \rrbracket_e = \cup \{ \hat{Q} \mid \hat{Q} \subseteq \llbracket \varphi \rrbracket_{e[X \mapsto \hat{Q}]} \}$

Fig. 1. Semantics of  $\mu$ -calculus formula

and in this case  $X$  is said to be *bound* in the fixed point formula expression. Here  $\sigma \in \{\mu, \nu\}$  and  $\mu$  and  $\nu$  denote the least and greatest fixed point operations respectively. The set of variables in a formula that are not bound are called *free*. The set of all  $\mu$ -calculus formulas defined over the domain  $(AP, \mathcal{X}, A)$  is denoted  $\Phi[AP, \mathcal{X}, A]$ . For a formula  $\varphi \in \Phi[AP, \mathcal{X}, A]$ ,  $FV(\varphi)$  denotes its set of free-variables,  $Sub(\varphi)$  denotes its set of sub-formulas,  $|\varphi|$ , called the length of  $\varphi$ , denotes the number of boolean and modal operators in  $\varphi$ , and  $ad(\varphi)$ , called the alternation depth of  $\varphi$ , denotes the number of nesting between  $\mu$  and  $\nu$  in  $\varphi$ . We also use  $nd(\varphi)$  to denote the nesting depth, i.e., the number of nesting of fixed point expressions in  $\varphi$ . For example, for  $\varphi \equiv \nu X. ([a]X \wedge \mu Y. (q \vee \langle b \rangle Y))$ ,  $nd(\varphi) = 2$ .

The semantics of a  $\mu$ -calculus formula  $\varphi$  is a set of states of a labeled transition system (LTS)  $M$  where the formula holds under a given environment  $e$ , and is denoted by  $\llbracket \varphi \rrbracket_e^M$ . Here the LTS  $M$  is a 5-tuple,  $(Q, A, T, AP, L)$  where  $Q$  is the set of states,  $T \subseteq Q \times A \times Q$  is the set of transitions labeled by actions in  $A$  and  $L : Q \rightarrow 2^{AP}$  is the labeling function which maps states to sets of propositions.  $e$  is a map of the form,  $e : \mathcal{X} \rightarrow 2^Q$  that associates with each of the formula variables a set of states and is used for defining the semantics of the variables. The LTS is typically understood from the context and so instead of writing  $\llbracket \varphi \rrbracket_e^M$ , we only write  $\llbracket \varphi \rrbracket_e$ , which is recursively defined in Fig. 1. In Fig. 1,  $e[X \mapsto \hat{Q}]$  denotes the environment  $e$  with the substitution that associates the state set  $\hat{Q} \subseteq Q$  with the variable  $X \in \mathcal{X}$ . In other words for  $Y \in \mathcal{X}$ ,  $e[X \mapsto \hat{Q}](Y)$  equals  $\hat{Q}$  if  $Y = X$ , and  $e(Y)$  otherwise.

We model a discrete event system to be controlled, called a plant, as well as a supervisor as LTSs with certain initial or start states, i.e., “initialized LTSs”. An LTS  $M$  with initial states  $Q_0 \subseteq Q$  is said to satisfy a formula  $\varphi$  if there exists an environment  $e$  such that  $Q_0 \subseteq \llbracket \varphi \rrbracket_e$ .

**Supervisory Control.** An uncontrolled discrete event plant  $P$  is modeled as an initialized LTS,  $P = (S_P, A, \delta_P, AP, L_P, S_{0,P})$ , where  $S_P$  is the finite set of states and  $A$  is the finite set of actions or events. At each state  $s \in S_P$ , the actions set is partitioned into a set of controllable actions  $A_c(s)$ , and a set of uncontrollable actions  $A_u(s)$ . Furthermore, at each state  $s \in S_P$ , an observation function  $O_s : A \rightarrow B \cup \{\epsilon\}$  defines the observed values for the actions when the plant state is  $s$ . If  $O_s(a) = O_s(b)$  for some  $a, b \in A$ , then the actions  $a$  and  $b$  are indistinguishable to a supervisor when the plant is at state  $s$ . Furthermore, if  $O_s(a) = \epsilon$  for some  $a \in A$ , then the action  $a$  is unobservable to a supervisor when the plant is at the state  $s$ . The set of transitions is denoted by  $\delta_P \subseteq S_P \times A \times S_P$ ;  $(s, a, s') \in \delta_P$  is said to be uncontrollable if  $a \in A_u(s)$ , and otherwise it is said to be controllable.  $AP$  is a finite set of atomic

propositions and  $L_P : S_P \rightarrow 2^{AP}$  is a labeling function. Finally,  $S_{0,P} \subseteq S_P$  is the set of initial states.

A supervisor  $C = (S_C, A, \delta_C, AP, L_C, S_{0,C})$ , is defined as another initialized LTS. Note that  $P$  and  $C$  share the sets of actions ( $A$ ) and atomic propositions ( $AP$ ). The controlled plant is obtained by the strict synchronous composition of  $P$  and  $C$ , denoted by  $P||C$ , which is defined as:  $(S_{PC}, A, \delta_{P||C}, AP, L_{P||C}, S_{0,PC})$ , where  $S_{PC} = S_P \times S_C$  is the state set;  $A$  and  $AP$  are the same sets as given in  $P$ ;  $\delta_{P||C} \subseteq S_{PC} \times A \times S_{PC}$  is the set of transitions of  $P||C$  and is given by,  $\{((s, q), \sigma, (s', q')) \mid (s, \sigma, s') \in \delta_P \wedge (q, \sigma, q') \in \delta_C\}$ ;  $L_{P||C} : S_{PC} \rightarrow 2^{AP}$  is the labeling function for  $P||C$ , which is defined as  $L_{P||C}(s, c) := L_P(s) \cap L_C(c)$ , and  $S_{0,PC} = S_{0,P} \times S_{0,C}$  denotes the set of initial states of  $P||C$ . In the rest of the paper, we will use  $s, s', s'', \dots$  to represent states in the plant,  $q, q', q'', \dots$  to represent states of supervisor and  $(s, a, s') \in \delta_P$  (or  $(q, a, q') \in \delta_C$ ) will be written as  $s \xrightarrow{a} s'$  (or  $q \xrightarrow{a} q'$ ).

Due to the presence of uncontrollable actions and observation functions, the supervisor is required to satisfy the following requirements: (a) *Control Compatibility*: when controlling the transitions defined at a plant state  $s$ , the supervisor must not disable any uncontrollable transitions defined at  $s$ ; and (b) *Observation Compatibility*: when controlling the transitions at a plant state  $s$ , the supervisor state-updates on enabled actions that are indistinguishable at  $s$  must be identical, and the supervisor state must not change on the execution of enabled actions that are unobservable at  $s$ .

**$\mu$ -calculus with observational formulas:  $\mathcal{O}$ - $\mu$ -calculus.** In order to capture the observation compatibility requirement as part of the quotiented formula, the traditional  $\mu$ -calculus syntax and semantics needs to be extended. The set of extended formulas, denoted by  $\Phi^{\mathcal{O}}[AP, \mathcal{X}, A]$ , includes expressions of the form  $\mathcal{O}_s$  representing observability requirement when the plant is at state  $s$ . The semantics of  $\mathcal{O}_s$  is set of states of an LTS model  $M = (Q, A, T, AP, L)$  that satisfy the formula and is defined as follows.

$$\llbracket \mathcal{O}_s \rrbracket_e = \left\{ \begin{array}{l} q \mid (\forall q \xrightarrow{a} q_a, q \xrightarrow{b} q_b : [O_s(a) = O_s(b) \Rightarrow q_a = q_b]) \\ \wedge (\forall q \xrightarrow{a} q_a : [O_s(a) = \epsilon \Rightarrow q_a = q]) \end{array} \right\} \quad (1)$$

We refer to  $\mathcal{O}_s$  as an *observational* formula.

### III. QUOTIENTING $\mu$ -CALCULUS SPECIFICATIONS

Given a discrete event plant  $P$ , and a  $\mu$ -calculus specification  $\varphi^\circ$ , the control problem is to determine the existence of a control and observation compatible supervisor  $C$  enforcing the specification, and to find one when it exists. We view the control problem as a specification transformation problem, which transforms the obligation  $\varphi^\circ$  on the controlled plant  $P||C$  to an obligation  $\varphi^\dagger$ , a  $\mathcal{O}$ - $\mu$ -calculus formula, on the supervisor  $C$ . Satisfiability of  $\varphi^\dagger$  ensures the existence of a supervisor  $C$ , while a model for a satisfiable  $\varphi^\dagger$  represents a desired supervisor. This specification transformation is referred to as *quotienting* and is similar in flavor to that in [1], [10], [5] where it is applied for efficient model checking of synchronous systems with replicated sub-systems, real-time systems and infinite-state parameterized systems respectively.

We use  $FV^\circ$ ,  $Sub^\circ$ ,  $nd^\circ$  to denote the set of free variables, the set of sub-formulas, and the nesting depth respectively, of  $\varphi^\circ$ . The quotienting operation involves introducing new

$$\begin{aligned}
1. \quad (\text{tt}/_T s) &= \begin{cases} \nu Z_s. \left( \bigwedge_{\substack{a \in A_u(s) \\ a \in A_c(s)}} \langle a \rangle (\text{tt}/_{T \cup \{Z_s\}} s') \right) \\ \text{if } Z_s \notin T \\ Z_s \text{ otherwise} \end{cases} \\
2. \quad (\text{ff}/_T s) &= \text{ff}. \quad 3. (p/_T s) = \begin{cases} (\text{tt}/_T s) \wedge p & \text{if } p \in L_P(s) \\ \text{ff} & \text{otherwise} \end{cases} \\
4. \quad (\varphi_1 \wedge \varphi_2/_T s) &= (\varphi_1/_T s) \wedge (\varphi_2/_T s). \\
5. \quad (\varphi_1 \vee \varphi_2/_T s) &= (\varphi_1/_T s) \vee (\varphi_2/_T s). \\
6. \quad (\langle a \rangle \varphi/_T s) &= (\text{tt}/_T s) \wedge \begin{cases} \langle a \rangle (\bigvee_{s':s \xrightarrow{a} s'} (\varphi/_T s')) \\ \text{if } \exists s' : s \xrightarrow{a} s' \\ \text{ff} \text{ otherwise} \end{cases} \\
7. \quad ([a] \varphi/_T s) &= (\text{tt}/_T s) \wedge \begin{cases} [a] (\bigwedge_{s':s \xrightarrow{a} s'} (\varphi/_T s')) \\ \text{if } \exists s' : s \xrightarrow{a} s' \\ \text{tt} \text{ otherwise} \end{cases} \\
8. \quad (\sigma X. \varphi_x/_T s) &= \begin{cases} \sigma X_{(s,k+1)}. (\varphi_x/_T [X_{(s,k)}/_{X_{(s,k+1)}}] s) \\ \text{if } X_{(s,k)} \in T \\ \sigma X_{(s,1)}. (\varphi_x/_T [X_{(s,1)}] s) \text{ otherwise} \end{cases} \\
9. \quad (X/_T s) &= \begin{cases} X_{(s,1)} & \text{if } X \in FV^\circ \\ X_{(s,k)} & \text{otherwise if } X_{(s,k)} \in T \\ (\sigma X. \varphi_x/_T s) & \text{otherwise where } \sigma X. \varphi_x \in \text{Sub}^0 \end{cases}
\end{aligned}$$

Fig. 2. Quotienting Rules

variables, one for each element in  $\mathcal{X} \times S_P \times \mathbb{N}$  ( $\mathbb{N}$  is the set of integers), which for  $X \in \mathcal{X}$ ,  $s \in S_P$ , and  $k \in \mathbb{N}$  is denoted as  $X_{(s,k)}$ . The set  $\mathcal{X} \times S_P \times \mathbb{N}$  itself is denoted as  $\mathcal{X}_{(S_P \times \mathbb{N})}$ . In order to keep track of control and observation compatibility requirements, quotienting also introduces new greatest fixed point variables of the form  $Z_s$ , one for each  $s \in S_P$ . The set of all such variables is denoted by  $\mathcal{Z}$ . We use  $\mathcal{T}$ , referred to as “tags”, to denote  $2^{\mathcal{X}_{(S_P \times \mathbb{N})} \cup \mathcal{Z}}$ . The tag set  $T \in \mathcal{T}$  maintains a certain history that is needed for quotienting a fixed-point formula or a fixed-point variable (to be explained in more detail below). Finally, in order to take into consideration the observation compatibility requirement, the quotienting results in  $\mathcal{O}$ - $\mu$ -calculus formulas. For each  $s \in S$ , and  $T \in \mathcal{T}$ , the quotienting function is a map,  $/_T s : \Phi[AP, \mathcal{X}, A] \rightarrow \Phi^{\mathcal{O}}[AP, \mathcal{X}_{(S_P, \mathbb{N})} \cup \mathcal{Z}, A]$  (see Fig. 2).

**Discussion.** The quotienting operation,  $(\varphi/_T s)$ , generates a formula  $\psi$  such that a controlled plant state  $(s, q)$  satisfies  $\varphi$  if and only if the supervisor state  $q$  satisfies  $\psi$ .

Rule 1 in Fig. 2 states that, regardless of the controlled plant state  $(s, q)$  (the propositional constant  $\text{tt}$  is satisfied by every state  $(s, q)$ ), the supervisor state  $q$  should satisfy control and observation compatibility. This is represented using a greatest fixed point formula over  $Z_s$  which requires that for all uncontrollable transitions  $s \xrightarrow{a} s'$  there exists a transition  $q \xrightarrow{a} q'$  such that  $q'$  is itself control and observation compatible with respect to actions in  $A_u(s')$ . Further the controllable actions in  $A_c(s)$  may or may not be permitted (implied by the box-modality). Also note that the supervisor must conform to the observability requirement when plant is at state  $s$  (denoted by  $O_s$ ), i.e., all actions that are indistinguishable ( $O_s(a) = O_s(b)$ ) must have identical successors while unobservable actions ( $O_s(a) = \epsilon$ ) must not update the supervisor state. Finally, in order to be able to terminate the recursive quotienting, the tag set keeps track of whether or not  $\text{tt}$  has already been quotiented against the

state  $s$ . If yes, the recursive quotienting terminates with the result equal to the corresponding fixed-point variable  $Z_s$ .

Rule 2 states that the controlled plant state  $(s, q)$  satisfies  $\text{ff}$  if and only if the supervisor state  $q$  satisfies  $\text{ff}$ . This is a tautology. Rule 3 states that for the controlled plant state  $(s, q)$  to satisfy the atomic proposition  $p$ ,  $p$  must be satisfied by  $s$  (otherwise no  $q$  state exists) and the supervisor state  $q$  must also satisfy  $p$ . In addition, the supervisor state should satisfy the control compatibility and observation requirements ( $(\text{tt}/_T s)$ ). Rules 4 and 5 states that quotienting commutes with conjunction and disjunction.

Rule 6 corresponds to the modal formula  $\langle a \rangle \varphi$ . A controlled plant state  $(s, q)$  satisfies  $\langle a \rangle \varphi$  if and only if there exists a successor state  $(s', q')$  on action “ $a$ ” such that  $(s', q')$  satisfies  $\varphi$ ; or equivalently, there exists a successor  $(s', q')$  such that  $q'$  satisfies  $(\varphi/_T s')$ . Proceeding further,  $(s, q)$  satisfies  $\langle a \rangle \varphi$  if and only if there exists  $a$ -successor  $s'$  and  $q$  satisfies the formula  $\langle a \rangle \bigvee_{s':s \xrightarrow{a} s'} (\varphi/_T s')$ . Note that the quotiented formula includes the extra conjunct  $(\text{tt}/_T s)$  to account for the control and observation compatibility requirements. Rule 7 is a dual of Rule 6 and can be understood similarly.

Rules 8 and 9 are used for quotienting fixed point formula and fixed-point variable respectively. Due to (i) the multiplicity of the plant states, (ii) the multiplicity of the variables that a certain fixed-point formula embeds, and (iii) the fact that quotienting is performed by recursively descending the parse-tree of the sub-formulas, the quotienting of a fixed-point formula can occur in association with different states and multiple times with each state. The tag set keeps track for each fixed-point formula and for each state, the number of times the fixed-point formula has been quotiented (with respect to the state). The count is incremented by one each time such a quotienting is performed. We argue that the count remains bounded. As a result the size of tag set itself remains bounded (see Theorem 1).

Rule 8 states that the quotient of a fixed point formula is the fixed point of the quotient formula, where the fixed point variable  $X_{(s,j)}$  appearing in the quotiented formula is a function of (a) the variable  $X$  of the formula being quotiented, (b) the state  $s$  against which the quotient is being performed, and (c) the number of times,  $j$ , such quotient has been performed in past. Note that each repeated quotienting operation on the same fixed point formula against the same state increments the count of quotienting and the corresponding variable is stored in the tag set  $T$ .

Rule 9 states that the quotient of a fixed point variable is the same as the quotient of the corresponding fixed point formula when that variable has not been quotiented in the past as indicated by the tag set; otherwise the result is the corresponding fixed point variable obtained from the tag set. The latter case causes the termination of the recursive computation. Note that, if the variable being quotiented is a free variable, the quotienting operations generates a new free variable as the result.

**Remark.** The first quotienting rule handles the state-dependent controllability and observation constraint. It is possible to modify this rule to accommodate any general constraint  $\psi_s$  that a supervisor state must satisfy when the plant is in state  $s$  by simply defining Rule 1 as  $(\text{tt}/_T s) = \psi_s$ .

We have the following theorems establishing the termi-

nation of the quotienting of a fixed-point formula and the correctness of the reduction of the control problem to the satisfiability of the quotiented formula.

*Theorem 1:* Given  $P = (S_P, A, \delta_P, AP, L_P, S_{0,P})$  and a control specification formula  $\varphi^\circ$ , the maximum number of times a fixed point expression  $\sigma Y.\varphi_y$ , a sub-formula of  $\varphi^\circ$ , is quotiented by any state  $s \in S_P$  is  $O(|S_P|^{nd^\circ})$ .

*Proof:* For a formula  $\varphi$  with  $nd(\varphi) = 1$ , the above theorem can be proved immediately. Assume that for  $\varphi$  with  $nd(\varphi) = n$ , the maximum number of times a fixed point expression is quotiented by a state is  $f(n)$ . We add an outer fixed point formula  $\sigma X.\varphi_x$  expression such that  $\varphi$  is a sub-formula in  $\varphi_x$  and  $nd(\sigma X.\varphi_x) = n + 1$ . If  $\sigma X.\varphi_x$  is quotiented once, then fixed point expressions in its sub-formula  $\varphi$  with  $nd(\varphi) = n$  can be quotiented  $f(n)$  times by a state (from induction hypothesis). Since  $X$  is the outermost fixed point variable, it can be quotiented  $|S_P|$  times. Proceeding further, fixed point expressions in its sub-formula  $\varphi$  can be quotiented  $f(n) \times |S_P|$  times by a state, i.e.,  $f(n+1) = f(n) \times |S_P|$ . Therefore,  $\forall i \geq 1. f(i) = |S_P|^i$ .  $\square$

*Theorem 2:* Consider a plant  $P = (S_P, A, \delta_P, AP, L_P, S_{0,P})$  and a control specification  $\varphi^\circ$ . Then for any supervisor  $C = (S_C, A, \delta_C, AP, L_C, S_{0,C})$ , a controlled state  $(s, q)$  satisfies  $\varphi^\circ$  iff the supervisor state  $q$  satisfies  $(\varphi^\circ /_\emptyset s)$ .

*Proof:* Follows from the discussion of the quotienting rules and Theorem 1.  $\square$

#### IV. SATISFIABILITY CHECKING AND MODEL DISCOVERY

We have reduced the problem of control to one of  $\mathcal{O}-\mu$ -calculus formula satisfiability. We now present a technique for checking the satisfiability of formula  $\varphi^\circ \in \Phi^\mathcal{O}[AP, \mathcal{X}, A]$  and discovering a model witnessing the satisfiability.

*Preliminaries.* To distinguish between greatest and least fixed point variables at various alternation depths, we assign an identifier,  $id$  to each variable.  $id(X) := 2 \times ad(\sigma X.\varphi_x)$  if  $\sigma = \nu$ ; otherwise  $id(X) := 2 \times ad(\sigma X.\varphi_x) - 1$ . Note that the  $id$  of a fixed point variable is an odd number if and only if it is of the least fixed point nature, and is the largest for the outer most fixed point variable.

**Tableau-based Approach.** We use a set of rules to construct a tableau from which satisfiability of a  $\mathcal{O}-\mu$ -calculus formula can be established and an appropriate model can be discovered. Each tableau rule is of the form:  $\frac{C_{\mathcal{H}^1}^0 \ M^1 \ C_{\mathcal{H}^2}^0 \ M^2 \ \dots \ C_{\mathcal{H}^n}^0 \ M^n}{C_{\mathcal{H}^1}^1 \ M^1 \ C_{\mathcal{H}^2}^2 \ M^2 \ \dots \ C_{\mathcal{H}^n}^n \ M^n}$  where each  $C$  is a set with elements of the form  $(\varphi, \vec{X}, \vec{N})$ . Here  $\varphi \in \Phi^\mathcal{O}[AP, \mathcal{X}, A]$  is a formula expression,  $\vec{X} \in \mathcal{X}^*$  is a sequence of fixed point variables, and  $\vec{N} \in \mathbb{N}^*$  is a sequence of integers. The history annotation  $\mathcal{H}$  is of the form  $\{(C^j, s^j)\}$ , where each  $s^j$  is a state-symbol.

For  $i > 0$ , each  $C^i, \mathcal{H}^i$  is determined as a function of  $C^0$  and  $\mathcal{H}^0$ , whereas  $M^0$  is defined as a function of  $M^1, M^2, \dots, M^n, C^0$  and  $\mathcal{H}^0$ . Further each  $M^i$  is a model expression which satisfies the conjunction of the formula expressions in  $C^i$ . There are two constant model expressions:  $M_{\text{tt}}$  is simply a single state (representing a true-model) while  $M_{\text{ff}}$  represents a model with no state (a false-model). Additionally  $M$  can take the form,  $s * [\bigwedge_i (a_i : M_i)]$ , meaning that  $M$  is rooted at a state  $s$ , possessing for each  $i$  an  $a_i$ -transition to the root of the model  $M_i$ . In case  $M_i = M_{\text{ff}}$  for some  $i$ , then  $s * [\bigwedge_i (a_i : M_i)]$  is equivalent to  $M_{\text{ff}}$ .

Each pair “ $C_{\mathcal{H}^i}^i \ M^i$ ” is referred to as a node of the tableau. We also say that the pair “ $C_{\mathcal{H}^0}^0 \ M^0$ ” is the numerator node of a tableau rule while “ $C_{\mathcal{H}^i}^i \ M^i$ ” ( $1 \leq i \leq n$ ) pairs are referred to as the denominator nodes. Fig. 3 presents the tableau rules.

*Discussion.* Given a formula  $\varphi^\circ$  whose satisfiability needs to be determined, a tableau-tree (or simply a tableau) rooted at the node “ $\{(\varphi^\circ, \epsilon, \epsilon)\}_\emptyset \ M$ ” is iteratively created by firing a tableau-rule whose numerator matches a node of the existing tableau-tree, and in which case successor tableau-tree nodes corresponding to the denominator are created. Note no successor is created for a tableau-rule whose denominator is empty (denoted as a “•”). When the iteration creating the tableau-tree terminates the model expressions of the leaf nodes possess definite valuations, which are used to associate definite valuations with the model expressions of all other nodes, including the root. At this point  $\varphi^\circ$  is satisfiable if and only if the model expression  $M$  of the root node is not  $M_{\text{ff}}$  (see Theorem 3).

Rule 1 states that  $M$  satisfies a conjunction of formula expressions one of which is  $\text{tt}$  if and only if it satisfies the conjunction without the conjunct  $\text{tt}$ . On the other hand Rule 2 states that  $M$  satisfies a conjunction of formula expressions one of which is  $\text{ff}$  if and only if  $M$  equals  $M_{\text{ff}}$ . Rule 3 states that  $M$  satisfies a conjunction of formula expressions one of which is an atomic proposition  $p$  if and only if  $M$  is rooted at a state  $s$  labeled  $p$  ( $p \in L(s)$ ) and also  $M$  satisfies the conjunction without the conjunct  $p$ . The fact that  $M$  is rooted at  $s$  has been represented as  $M = s * B$ , where  $B$  is of the form  $\bigwedge_i (a_i : M_i)$ .

Rule 4 states that the true-model (or equivalently any model) satisfies a conjunction of *empty* set of formula expressions. Rule 5 states that  $M$  satisfies a conjunctive formula if and only if it satisfies each of the conjuncts, whereas Rules 6 and 7 state that  $M$  satisfies a disjunctive formula if and only if it satisfies one of the disjuncts. Rule 8 states that  $M$  satisfies a fixed point formula  $\sigma X.\varphi$  if and only if it satisfies  $\varphi$ .

Rule 9 states that  $M$  satisfies a conjunction of formula expressions one of which is a free variable  $X \in FV^\circ$  if and only if it satisfies the conjunction without the conjunct  $X$ . This is so because the satisfiability of a free variable can always be guaranteed by selecting an appropriate environment (see Fig. 1). Rule 10 is similar but applies to a conjunction of formula expressions one of which is a bound fixed-point variable. In this case  $M$  satisfies the conjunction if and only if it satisfies the conjunction with the fixed-point variable replaced by its fixed point formula expression. At the same time the fixed-point variable  $X$  is pre-pended to the sequence  $\vec{X}$  of the corresponding tuple. This is to keep track of the sequence of fixed-point variables visited thus far.

Rules 11, 12, 13 and 14 apply when all the formula expressions in the numerator  $C$  are modal formula expressions and observational formulas. Rule 14 applies when the modal formula expressions in the numerator  $C$  are also present in an element  $C'$  of the history set (and otherwise the Rule 11 or 12 or 13 is applied). Rules 11 and 12 are applied when there exists a box-modal formula on an action  $a$  with no diamond-modal formula on the same action (and otherwise Rule 13 is applied). Together Rules 11 and 12 state that

$$\begin{array}{l}
\boxed{1} \frac{\{(\tau\tau, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M}{C_{\mathcal{H}} M} \quad \boxed{2} \frac{\{(\text{ff}, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M := M_{\text{ff}}}{\bullet} \quad \boxed{3} \frac{\{(p, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M := s*B}{C_{\mathcal{H}} M := s*B} \text{ where } p \in L(s) \\
\boxed{4} \frac{\{ \}_{\mathcal{H}} M := M_{\tau\tau}}{\bullet} \quad \boxed{5} \frac{\{(\varphi \wedge \psi, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M}{\{(\varphi, \vec{X}, \vec{N}), (\psi, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M} \quad \boxed{6} \frac{\{(\varphi \vee \psi, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M}{\{(\varphi, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M} \quad \boxed{7} \frac{\{(\varphi \vee \psi, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M}{\{(\psi, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M} \\
\boxed{8} \frac{\{(\sigma X, \varphi, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M}{\{\varphi, \vec{X}, \vec{N}\} \cup C \}_{\mathcal{H}} M} \quad \boxed{9} \frac{\{(X, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M}{C_{\mathcal{H}} M} \text{ where } X \in FV^\diamond \\
\boxed{10} \frac{\{(X, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M}{\{(\sigma X, \varphi, (X, \vec{X}), \vec{N})\} \cup C \}_{\mathcal{H}} M} \quad \sigma X, \varphi \in \text{Sub}^\diamond \\
\boxed{11} \frac{\{([a]\varphi, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M}{\{([a]\varphi \wedge (a)\tau\tau, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M} \quad \boxed{12} \frac{\{([a]\varphi, \vec{X}, \vec{N})\} \cup C \}_{\mathcal{H}} M}{[C - \{([a]\varphi_i, \vec{X}_i, \vec{N}_i) \mid ([a]\varphi_i, \vec{X}_i, \vec{N}_i) \in C\}]_{\mathcal{H}} M} \\
\text{if } C = \{(\psi_i, \vec{X}_i, \vec{N}_i) \mid \psi_i \in \{[a]\varphi_j, \mathcal{O}_k\}\}, \quad \exists C' = \{(\psi_i, \vec{X}'_i, \vec{N}'_i), s\} \in \mathcal{H}, \text{ and } \exists [a]\varphi_j, \vec{X}_j, \vec{N}_j \in C : ((a)\varphi_j, \vec{X}_j, \vec{N}_j) \notin C \\
\boxed{13} \frac{C_{\mathcal{H}} M}{C_{\mathcal{H}'}^{a,1} M^{a,1} \dots C_{\mathcal{H}'}^{a,n} M^{a,n}} \\
\text{if } C = \{(\psi_i, \vec{X}_i, \vec{N}_i) \mid \psi_i \in \{[a]\varphi_j, \mathcal{O}_k\}\}, \quad \exists C' = \{(\psi_i, \vec{X}'_i, \vec{N}'_i), s\} \in \mathcal{H}, \text{ and } \forall [a]\varphi_j, \vec{X}_j, \vec{N}_j \in C : \exists ((a)\varphi_j, \vec{X}_j, \vec{N}_j) \in C \\
\text{in which case, letting } O := Cl\{O_i \mid (O_i, \vec{X}_i, \vec{N}_i) \in C\}, \text{ we have} \\
M := s_C * \bigwedge_{a,j} \left\{ \begin{array}{ll} \bigwedge_{b:O(b)=O(a), ([b]\varphi_i, \vec{X}_i, \vec{N}_i) \in C} [b: M^{a,j}] & \text{if } O(a) \neq \epsilon \\ \bigwedge_{b:O(b)=O(a), ([b]\varphi_i, \vec{X}_i, \vec{N}_i) \in C} [b: M^{a,j} (\equiv M)] & \text{otherwise} \end{array} \right\} \\
\forall j : ((a)\varphi_j, \vec{X}_j, \vec{N}_j) \in C : C^{a,j} := \begin{cases} \{(\varphi_i, \vec{X}_i, i, \vec{N}_i) \mid O(b) = O(a) \wedge ([b]\varphi_i, \vec{X}_i, \vec{N}_i) \in C\} \cup \{(\varphi_j, \vec{X}_j, j, \vec{N}_j)\} & \text{if } O(a) \neq \epsilon \\ \{(\varphi_i, \vec{X}_i, i, \vec{N}_i) \mid ([b]\varphi_i, \vec{X}_i, \vec{N}_i) \in C \text{ and } O(b) = O(a)\} \cup C & \text{otherwise} \end{cases} \\
\mathcal{H}' := \mathcal{H} \cup \{(\tilde{C}, s_C)\}, \text{ where } \tilde{C} := \{(\psi_i, \vec{X}_i, i, \vec{N}_i) \mid (\psi_i, \vec{X}_i, \vec{N}_i) \in C\} \\
\boxed{14} \frac{C_{\mathcal{H}} M}{\bullet} \text{ if } C = \{(\psi_i, \vec{X}_i, \vec{N}_i) \mid \psi_i \in \{[a]\varphi_j, \mathcal{O}_k\}\}, \quad \exists C' = \{(\psi_i, \vec{X}'_i, \vec{N}'_i), s\} \in \mathcal{H} \\
\text{in which case, } M := \begin{cases} M_{\text{ff}} & \text{if } \text{lfp}(C, C') \\ s & \text{otherwise} \end{cases}, \text{ where } \text{lfp}(C, C') \text{ is a Boolean expression that holds iff} \\
\exists i_0, i_1, \dots, i_n = i_0 : \forall j \in [0, n-1] : \vec{N}'_{i_j} \in \text{succ}(\vec{N}_{i_{j+1}}), \max\{id(X) \mid X \in \vec{X}_{i_{j+1}} / \vec{X}'_{i_j}, j \in [0, n-1]\} \text{ is odd}
\end{array}$$

Fig. 3. Tableau for satisfiability and model discovery for  $\varphi^\diamond$

$[a]\varphi$  can be satisfied at a state if and only if either some outgoing  $a$ -transition exist and all its destinations satisfy  $\varphi$  (Rule 11) or there is no  $a$ -transition (Rule 12). This reduction similar to the one presented in [8] where the authors reduced  $[a]\varphi$  to  $(a \rightarrow \emptyset \wedge a \rightarrow \{\varphi\})$ ;  $a \rightarrow \Psi$  is defined as  $\bigwedge \{ \langle a \rangle \psi \mid \psi \in \Psi \} \wedge [a] \vee \{ \psi \mid \psi \in \Psi \}$ .

Rule 13 is applied when every box-modal action has a corresponding diamond modal obligation. As  $C$  may contain several observational formulas  $\mathcal{O}_i$ , it is necessary to identify a combined observation function  $O$ , called the *closure of*  $\{\mathcal{O}_i\}$  and denoted  $Cl\{\mathcal{O}_i\}$ , such that

- $[O(a) = O(b)] \Leftrightarrow [\exists i, \exists j : (O_i(a) = O_j(b)) \vee (O_i(a) = O_j(c) \wedge O(c) = O(b))]$
- $[O(a) = \epsilon] \Leftrightarrow [\exists i : (O_i(a) = \epsilon) \vee (O(a) = O(b) \wedge O_i(b) = \epsilon)]$

It can be verified that the observational formula  $\mathcal{O}$  corresponding to the observation function  $O$  satisfies:  $\llbracket \mathcal{O} \rrbracket_e = \bigcap_i \llbracket \mathcal{O}_i \rrbracket_e$ . Proceeding further, Rule 13 states that for any action  $a$ , a set of formula expression of the form  $\{[a]\varphi\}$  is satisfiable by a model state if and only if

- Each diamond obligation is satisfied by some  $a$ -successor and each  $a$ -successor satisfies all of the box obligations.
- Indistinguishable events led to the same successor states

and unobservable events result in self-loops. Consequently, when  $a$  and  $b$  are indistinguishable, all  $a$ -successors also satisfy all formulas  $\psi$  such that  $[b]\psi$  is present in  $C$ , whereas when  $a$  is unobservable there is a single  $a$ -successor (reached by a self-loop on  $a$ ) which satisfies all formulas  $\psi$  such that  $\langle b \rangle \psi$  is present in  $C$  and  $b$  is indistinguishable from  $a$ , together with all the formulas in  $C$ .

Accordingly for each action  $a$  with  $O(a) \neq \epsilon$ , a denominator node aggregates all the box-modal obligations over indistinguishable events and one diamond-modal obligation (see definition of  $C^{a,j}$  in Rule 13 of Fig. 3); if  $O(a) = \epsilon$  then the aggregation includes all the diamond- and box-modal obligations on all unobservable actions together with the current state obligations (since the unobservable transitions will have only one successor, the current state).

The history is augmented to record (i)  $C$ , modified to include the ancestor node tag by pre-pending  $\vec{N}_i$  with  $i$ , and (ii) the model state  $s_C$  that satisfies the formula expressions in  $C$ . Such augmentation is performed to record the fact that  $C$  was visited. Note that a  $C$  containing only the modal formula expressions (where each box-formula has a diamond-formula) and observational formula expressions is recorded in the history set. This is because only for such a

$C$ , a transition in the model state occurs (on the associated modal actions).

Finally, for every action  $a$ , the synthesized model  $M$  contains (i) outgoing transitions on  $a$  to the roots of the models  $M^{a,j}$ , (ii) outgoing transitions on  $b$  to the roots of the models  $M^{a,j}$ , if  $b$  and  $a$  are indistinguishable ( $O(b) = O(a)$ ), and (iii) a self-loop on  $a$  at the current state  $s_C$  when  $a$  is unobservable ( $O(a) = \epsilon$ ).

Rule 14 applies when the modal formula expressions in the numerator  $C$  are also present in an element  $C'$  of the history set, implying that such formula expressions are being revisited owing to the expansion of certain fixed point variables (Rule 10). The set of fixed point variables expanded is given by  $\overrightarrow{X_{i_{j+1}}}/\overrightarrow{X_{i_j}}$ ,  $j \in [0, n-1]$ , where the notation “ $\overrightarrow{X_1}/\overrightarrow{X_2}$ ” removes the suffix  $\overrightarrow{X_2}$  from the sequence  $\overrightarrow{X_1}$ . The predicate  $\text{Lfp}(C, C')$  holds if and only if the outermost fixed point variable (one having the largest  $id$ ) expanded is of the least fixed point nature (i.e. its  $id$  is odd). If  $\text{Lfp}$  evaluates to true, then the model  $M$  is set to  $M_{\text{Lfp}}$ ; otherwise it is set equal to the state  $s$  corresponding to the element  $C'$  in  $\mathcal{H}$ .

We have the following theorem attesting to the correctness of the tableau-based satisfiability-checking model-discovery algorithm.

*Theorem 3:* A  $\mathcal{O}$ - $\mu$ -calculus formula  $\varphi^\circ$  is satisfiable iff there exists a tableau with root node “ $\{(\varphi^\circ, \epsilon, \epsilon)\}_\emptyset M$ ”, such that  $M$  is assigned to a non false-model.

*Proof:* Follows from the above discussion.  $\square$

#### A. Complexity

We consider a nondeterministic plant with state set  $S_P$ , maximum branching degree  $d^a$  on any action  $a$  ( $d^a=1$  for a deterministic  $P$ ), maximum branching degree  $d$  over all actions, and a control specification  $\varphi^\circ$ .

The length of the quotiented formula  $\varphi^\ddagger$  can be estimated by first estimating its nesting depth and next estimating the number of boolean and modal operators appearing at each level of the nesting. The nesting depth of the quotiented formula is  $O(|S_P|^{nd^\circ})$  (from Theorem 1). Now to estimate the number of boolean and modal operators at any level of the nesting, we consider the “amplification factor” due to each quotienting rule (with respect to the existing number of boolean and modal operators in  $\varphi^\circ$ , which is  $|\varphi^\circ|$ ), and aggregate them to get the overall amplification. All but Rules 1, 6, and 7 have the unity amplification factor. The amplification factor of Rule 1 is  $O(|S_P| \times d)$  since the number of boolean operators in each greatest fixed point formula is  $O(d)$  and the number of greatest fixed point formula introduced at a nesting level is  $O(|S_P|)$ . Rules 6 and 7 have the amplification factor of  $O(\max_a d^a) \leq O(d)$ . So the overall amplification factor is  $O(1 + (|S_P| \times d) + d)$ . Multiplying this by the number of boolean and modal operators in  $\varphi^\circ$ , i.e.,  $|\varphi^\circ|$ , yields the second estimate as  $O([1 + (|S_P| + 1) \times d] \times |\varphi^\circ|)$ . So the length of the quotiented formula is  $O([1 + (|S_P| + 1) \times d] \times |\varphi^\circ| \times |S_P|^{nd^\circ})$ .

Note that when the controllability and observability requirements are state-independent then Rule 1 can be simplified as:

$$(\text{tt}/_T s) = \begin{cases} \nu Z. (\bigwedge_{a \in A_u} \langle a \rangle Z \bigwedge_{b \in A_c} [b] Z \wedge \mathcal{O}) & \text{if } s \in S_{0,P} \\ \text{tt} & \text{otherwise} \end{cases}$$

where  $\mathcal{O}$  denotes the state-independent observational for-

mula. In this case, the length of the quotiented formula  $\varphi^\ddagger$  becomes  $O([(1 + d + |A|) \times |\varphi^\circ|] \times |S_P|^{nd^\circ})$ . A similar simplification is applicable for any other state-independent controllability and observability constraints.

We next consider the complexity of satisfiability checking and model discovery for the quotiented formula  $\varphi^\ddagger$ , which considers at each of its nesting level, all possible subsets of the subformulae of  $\varphi^\ddagger$ . At each nesting level the number of possible subsets of the subformulae  $\varphi^\ddagger$  examined is  $O(2^{[1 + (|S_P| + 1) \times d] \times |\varphi^\circ|})$ . So the overall complexity is given by  $O(|S_P|^{nd^\circ} \times 2^{[1 + (|S_P| + 1) \times d] \times |\varphi^\circ|})$ . In light of the discussion of the previous paragraph, the complexity simplifies to  $O(|S_P|^{nd^\circ} \times 2^{(1 + d + |A|) \times |\varphi^\circ|})$  when the controllability is state-independent. Note that this is polynomial in the number of plant states  $S_P$ .

A prototype implementation of our technique is available at <http://www.cs.iastate.edu/~sbasu/control-quot>.

## V. CONCLUSION

We presented a quotienting-based approach for supervisory control of nondeterministic discrete event plants under partial observation of events subject to control specifications expressed in the propositional  $\mu$ -calculus. Central to our method is a direct-quotienting of the  $\mu$ -calculus specification against the plant model. The solution required extending the classic  $\mu$ -calculus to include observational formulas, the syntax and semantics of which were defined. A control and observation compatible supervisor exists if and only if the quotiented formula in the extended logic is satisfiable, and further a model witnessing the satisfiability can be used as a supervisor. A sound and complete tableau-based methodology for satisfiability checking and model discovery for a formula in the extended logic was also developed.

## REFERENCES

- [1] H. Andersen. Partial model checking. In *Logic in Computer Science*, 1995.
- [2] A. Arnold, X. Briand, G. Point, and A. Vincent. A generic approach to the control of discrete event systems. In *Conference on Decision and Control*, pages 1–5, 2005.
- [3] A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, pages 7–34, 2003.
- [4] S. Basu and R. Kumar. Quotient based approach to control of nondeterministic discrete-event systems with  $\mu$ -calculus specification. In *Conference on Decision and Control*, San Diego, CA, 2006.
- [5] S. Basu and C. R. Ramakrishnan. Compositional analysis for verification of parameterized systems. *Theoretical Computer Science*, 354(2):211–229, 2006.
- [6] X. Briand. Dynamic control with indistinguishable events. *Discrete Event Dynamic Systems*, 16(3):353–384, 2006.
- [7] E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs. *Siam Journal of Computing*, 29(1):132–158, 1999.
- [8] D. Janin and I. Walukiewicz. Automata for the modal  $\mu$ -calculus and related results. In *International Symposium on Mathematical Foundations of Computer Science*, 1995.
- [9] D. Kozen. Results on the propositional  $\mu$ -calculus. *Theoretical Comput. Sci.*, 1983.
- [10] K. J. Kristoffersen, F. Laroussinie, K. G. Larsen, P. Pettersson, and W. Yi. A compositional proof of a real-time mutual exclusion protocol. In *Joint Conf. Theory and Practice of Software Development*, pages 565–579, 1997.
- [11] S. Pinchinat and S. Riedweg. A decidable class of problems for control under partial observation. *Information Processing Letters*, 95(4):454–465, 2005.
- [12] S. Riedweg and S. Pinchinat. Quantified mu-calculus for control synthesis. In *Mathematical Foundations of Computer Science*, 2003.
- [13] C. Stirling. Games and modal mu-calculus. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 298–312, 1996.