# Control of Non-Deterministic Systems With $\mu$-Calculus Specifications Using Quotienting

Samik Basu and Ratnesh Kumar, *Fellow, IEEE*

***Abstract*—The supervisory control problem for discrete event system (DES) under control involves identifying the supervisor, if one exists, which, when synchronously composed with the DES, results in a system that conforms to the control specification. In this context, we consider a non-deterministic DES under complete observation and control specification expressed in action-based propositional $\mu$-calculus. The key to our solution is the process of quotienting the control specification against the plan resulting in a new $\mu$-calculus formula such that a model for the formula is the supervisor. Thus the task of control synthesis is reduced a problem of $\mu$-calculus satisfiability. In contrast to the existing $\mu$-calculus quotienting-based techniques that are developed in deterministic setting, our quotienting rules can handle nondeterminism in the plant models. Another distinguishing feature of our technique is that while existing techniques use a separate $\mu$-calculus formula to describe the controllability constraint (that uncontrollable events of plants are never disabled by a supervisor), we absorb this constraint as part of quotienting which allows us to directly capture more general *state-dependent* controllability constraints. Finally, we develop a tableau-based technique for verifying satisfiability of quotiented formula and model generation. The runtime for the technique is exponential in terms of the size of the plan and the control specification. A better complexity result that is polynomial to plant size and exponential to specification size is obtained when the controllability property is state-independent. A prototype implementation in a tabled logic programming language as well as some experimental results are presented.**

***Index Terms*—Discrete event systems (DES), non-deterministic plant, $\mu$-calculus, supervisory control.**

## I. Introduction

SUPERVISORY control problem was introduced in [1], [2] using deterministic automata representations of the plan and the control-specification. Since then several works focussed on generalization resulting from non-determinism in plants and from expressing control specification using temporal logics and bisimulation equivalences.

In the paper, we consider a DES supervisory control problem where a non-deterministic plant and specification are described as labeled transition systems and modal $\mu$-calculus respectively. The central tenet of our technique is to develop a quotienting based technique to decide the existence of supervisor and generate the same if one exists. The quotienting technique can be described as follows. Given a plant $P$ and the specification $\varphi$ of the controlled plant, quotienting operation generates a new specification $\psi = (\varphi / P)$ (in the same logic as $\varphi$) describing the obligation on supervisor $C$ such that $P$ when controlled by $C$ satisfies $\varphi$. A supervisor $C$ exists only and only when $\psi = (\varphi / P)$ is satisfiable, and a model witnessing the satisfiability is one such $C$. The quotienting operation is defined on the basis of composition definition of $P$ and $C$ and the semantics of logic in which $\varphi$ is defined (modal $\mu$-calculus in our case). It also takes into consideration the possible non-determinism in $P$ and controllability constraint of the possible supervisors (for example its inability to control/disable any uncontrollable actions of the plant).

The DES control problem subject to $\mu$-calculus specification was examined in [3], where the problem was considered in the setting of control of a deterministic plant. The authors also allowed time-varying uncontrollable actions and "projection-type" partial observation function. The work was later extended by considering indistinguishable actions in [4], [5]. We allow nondeterminism in the plant model and more general state-based uncontrollable events under complete observability of events. While at its core our technique also relies on reducing the problem of supervisor synthesis to that of model generation for a satisfiable formula, there are several significant differences; the **key distinguishing aspects** are enumerated as follows:

1) We perform quotienting at the level of $\mu$-calculus formulas. On the other hand, [3]–[5] computes the alternating tree automata representation of $\mu$-calculus and apply quotienting on the tree automata.

2) Quotienting lets us handle not only *state-dependent controllability* requirements but also nondeterministic plants in a straightforward manner. In contrast, [3]–[5] impose a controllability constraint as a separate $\mu$-calculus formula, which is *state-independent*.

3) In [3]–[5], a plant model is assumed deterministic, and further the controllability constraint used assumes a supervisor to be deterministic, obviating the need for the $\mu$-calculus framework. Our work is based on our prior conference publication [6] and allows nondeterminism in plant as well as controller models. Also, while [3]–[5] allow a partial observability of events, this is not adequate to capture nondeterminism: Partial observation identifies only the actions

Manuscript received September 23, 2020; revised December 10, 2020; accepted January 10, 2021. The work was supported in part by the National Science Foundation (NSF-ECCS-1509420, NSF-CSSI-2004766). Recommended by Associate Editor MengChu Zhou. *(Corresponding author: Ratnesh Kumar.)*

Citation: S. Basu and R. Kumar, "Control of non-deterministic systems with $\mu$ -calculus specifications using quotienting," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 5, pp. 953–970, May 2021.

S. Basu is with the Department of Computer Science, Iowa State University, Iowa 50011-3060 USA (e-mail: sbasu@iastate.edu).

R. Kumar is with the Department of Electrical and Computer Engineering, Iowa State University, Iowa 50011-3060 USA (e-mail: rkumar@iastate.edu).

Digital Object Identifier 10.1109/JAS.2021.1003964

from the point of observation, whereas nondeterminism identifies actions as well as control and specification from the point of observation. Reference [7] proposed extension to their prior work [3] to incorporate nondeterminism in both plant and controller models; in [8] we proposed extension to our prior work [6] to incorporate partial observability.

References [9], [10] also considered automata-theoretic quotienting. However, as opposed to satisfiability checking for supervisor synthesis, the authors coupled the existence of supervisor with the specification logic expressed in *quantified $\mu$-calculus*. Furthermore, as with [3], the work in [10] is limited to deterministic plants.

Our quotienting procedure is closely related to the one described in partial model checker [11]. The work presents quotienting operation on equational $\mu$-calculus against labeled transition systems. The main aim is to show the applicability of quotienting for model checking systems with regular structures (e.g., ring topology). This work is generalized in [12] where quotienting is defined for $\mu$-calculus formulas against arbitrary CCS processes. The technique is coupled with limit computation over sequence of $\mu$-calculus formulas to develop a method for model checking parametrized systems. We present a quotienting operation for labeled transition representing a non-deterministic plant model where labels capture plan-events and certain events are classified as controllable.

The result of quotienting operation is a new $\mu$-calculus formula such that its satisfiability proves the existence of a supervisor and the satisfiable model is one such supervisor. A number of notable work have presented different techniques for satisfiability checking for $\mu$-calculus by verifying alternating tree automata emptiness [13], by identifying winning strategy in parity games [3], [14], [15], or verifying satisfiability of equivalent disjunctive $\mu$-calculus formula. In contrast, we use a tableau-based method for satisfiability checking and model discovery. Central to our tableau is the maintenance of a *history* set which ensures that least fixed point sub-formulas are captured by finite-path in the satisfiable model while greatest fixed point sub-formulas are captured by cycles in the model.

The contributions of this work is summarized as follows:

1) We present a quotienting technique for control synthesis, where the desired property is expressed in $\mu$-calculus, and a plant model is expressed as finite state machine. The proposed quotienting technique methodically translates the desired property expressed in $\mu$-calculus into obligation for a controller, that is also expressed in $\mu$-calculus. This allows us to deal with non-determinism in plant model and state-dependent controllability directly using the quotienting.

2) We have generalized existing quotienting technique used in the context of partial/compositional model checking. Unlike model checking, where all events are of the same type, in our case, events can be classified as controllable vs. uncontrollable, and which can vary from state to state.

3) We present a tableau-based technique for generating satisfiable model (which, in our case, is the model of a controller) for $\mu$-calculus formula.

4) We present a preliminary implementation to show the viability of our technique.

The rest of the paper is organized as follows. In Section II, we discuss the relevance of our work in the context of controller synthesis work that uses techniques other than quotienting. Section III gives a brief overview of the modal $\mu$-calculus (Section III-B), followed by the description of the control problem (Section III-C). Section IV presents a simple example that is used for illustrating our approach. In Section V we present our technique of quotienting $\mu$-calculus specification with respect to a plant model to obtain a quotiented formula representing the obligations of a supervisor. We then develop a methodology to check for the satisfiability of a quotiented formula and identify a supervisor model when possible (Section VI). Section VII describes our prototype implementation. Preliminary experimental results are discussed in Section VIII. We conclude the paper in Section IX.

## II. Other Related Work

The control problem in domain of nondeterministic plant and specification is studied in [16]–[20]. The authors in [20] show how to transform their control problem in nondeterministic setting to one in deterministic setting with an added partial observability. Control of plants modeled using nondeterministic state machines for language specification is also studied in [21], [22]. All the work used deterministic supervisors.

The use of a nondeterministic supervisors for specification represented using language model was explored in [23], [24]. The problem of nondeterministic control was formalized in [25]. The authors focused on control under partial observation for language specification and introduced the notion of achievability (a property weaker than controllability and observability combined). Nondeterministic supervisors have also been used in works allowing nondeterminism in specification. Such specifications are able to impose both sequencing and branching constraints and are modeled using branching-time temporal logic such as CTL$^*$ and $\mu$-calculus, or using bisimulation or simulation equivalence type requirements. In [26] a nondeterministic specification was specified in the temporal logic of CTL$^*$, generalizing the work reported in [27] which used CTL to express specification. Other work related to control subject to temporal logic based specification include [3], [28]–[31].

Bisimulation relation has been used as a technique for supervisory control of deterministic systems subject to language specifications in [32]–[36] the controllability and observability are characterized as a bisimulation type relation. Reference [37] studied the bisimilarity control for a deterministic specification, treating all events controllable. Reference [38] studied bisimilarity control for a partial specification (defined over an “ external event set”) under several restrictions: Deterministic plant, all events controllable, and all events treated indistinguishable from the specification perspective be either all enabled or all disabled at a state. Reference [39] studied the bisimilarity control for again deterministic plants subject to a possibly nondeterministic partial specification, thereby relaxing some of the assumptions of [38].

The most general bisimulation equivalence control problem was finally studied in [40], in which both the plant as well as the specification are nondeterministic. (The same authors also studied the special case of deterministic control in [41], and provided additional comments in [42].) In [43], the author presented a new bisimulation based control synthesis technique with an improved runtime complexity. The extension to allow partial observation of events was reported by the same authors in [44], [45], and the simulation equivalence control under the above generalized framework was addressed by the same authors in [46]. [47] discussed synthesis of maximally permissive controller in the context of simulation equivalence.

In our technique, as we focus on satisfiability of properties expressed in $\mu$-calculus, it is equivalent to synthesis problem where the controlled plant is bisimulation equivalent to the desired behavior. Furthermore, our quotienting based technique does not guarantee the generation of maximally permissive controller. However, note that the behavior of the controller (being synthesized) is expressed in $\mu$-calculus formula resulting from quotienting. In other words, all feasible controller behavior, including the maximally permissive one, is captured by the $\mu$-calculus formula. As part of future work, it would be interesting to investigate and extend our tableau-based model generation technique to generate a maximally permissive controller, and also explore the application of BDD representation (as for example in [48]) for computational improvement.

## III. Preliminaries

### A. Labeled Transition System

The dynamic behavior of system is typically expressed using transition system, where states in the system correspond to configurations of the system, while the directed edges/transitions between configurations describe the evolution of the system. In our case, we augment each transition with label to capture the event/action-name that identifies the evolution due to the transition. Formally, a *labeled transition system* (LTS) $M$ is $(S, A, T, AP, L)$, where $S$ is the set of states, $T \subseteq S \times A \times S$ is the set of transitions labeled by actions in $A$ and $L : S \to 2^{AP}$ is the labeling function which maps states to sets of propositions. If a state is "labeled" by a set of propositions, we say that the propositions are valid or true in that state; all other propositions are false in that state. The truth-values of the propositions describe the states.

### B. Propositional $\mu$-Calculus Specifications

The $\mu$-calculus [49], [50] uses explicit least and greatest fixed points to express temporal ordering of events and states. The set of properties, thus, induced is strictly larger than the one expressible in temporal logics such as LTL, CTL. The syntax of $\mu$-calculus formulas involves propositional constants ($\mathtt{tt}$, $\mathtt{ff}$), atomic propositions $AP$, modal actions $A$ with modalities ([] and $DIAM$), boolean connectives ($\neg$, $\vee$, $\wedge$), fixed point variables $X \in \mathcal{X}$ and expressions.

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

In the above, $\langle a \rangle$ is referred to as diamond modality over action $a$; informally, it expresses the existential quantification $a$ successor. On the other hand, $[a]$ is referred to as box modality over action $a$; informally, it expresses the universal quantification of all $a$ successor. The fixed point formula expression $\sigma X.\phi$ includes the type of fixed point $\sigma \in \{\mu, \nu\}$ ($\mu$: least fixed point operator and $\nu$: greatest fixed point operator), the variable $X$ bound by the operator $\sigma$, and $\phi$ the formula describing the definition of the fixed point expression. In any formula, a variable not bound by any fixed point operator is called free variable. 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 $\varphi$, we will use the following notational convenience: $FV(\varphi)$ denotes its set of free-variables, $Sub(\varphi)$ denotes its set of sub-formulas, $|\varphi|$, called length of $\varphi$, denotes the number of boolean and modal operators in $\varphi$, and $ad(\varphi)$, called alternation depth of $\varphi$, denotes the number of nesting between $\mu$ and $\nu$ in $\varphi$ [51]. $ad(\varphi)$ is recursively defined as follows:

1) $ad(\mathtt{tt}) = ad(\mathtt{ff}) = ad(X) = 0$
2) $ad(\varphi \wedge \psi) = ad(\varphi \vee \psi) = \mathtt{max}\{ad(\varphi), ad(\psi)\}$
3) $ad([a]\varphi) = ad(\langle a \rangle \varphi) = ad(\varphi)$
4) $ad(\sigma X.\varphi) = \mathtt{max}\begin{pmatrix} \{1, ad(\varphi)\} \cup \\ \{ad(\sigma Y.\psi) + 1 \mid (\sigma Y.\psi \in Sub(\varphi)) \\ \wedge (X\ \mathtt{free\ in}\ \psi) \\ \wedge (\mathtt{fp}(X) \neq \mathtt{fp}(Y))\} \end{pmatrix}$

where for $\sigma X.\varphi$, $\mathtt{fp}(X) = \sigma$. We use $nd(\varphi)$ to denote the nesting depth, i.e., the number of nestings of fixed point expressions in $\varphi$. The definition is identical to that for $ad(\varphi)$ except when $\varphi = \sigma X.\varphi$.

$$nd(\sigma X.\varphi) = \mathtt{max}\begin{pmatrix} \{1, nd(\varphi)\} \cup \{nd(\sigma_Y Y.\varphi_Y) + 1 \mid \\ (\sigma_Y.Y.\varphi_Y \in Sub(\varphi))\} \end{pmatrix}. \quad (1)$$

Using the modal operators and explicit fixed points, $\mu$-calculus formula can express complex temporal ordering of actions and sequence of propositions. The semantics of a formula $\varphi$ is a set of states in LTS such that the evolution from these states as described in the LTS conform to (model) the ordering of actions and propositions specified by $\varphi$. We say that the states that belong to the semantics of $\varphi$ satisfy $\varphi$. Due to the presence of explicit fixed points and variables bound by the fixed points, the semantics function depends on the mappings of these variables to sets of states. This mapping is referred to as the environment $e : \mathcal{X} \to 2^S$. The semantics of $\mu$-calculus formula $\varphi$ in the context of a given environment $e$, and is denoted by $[[\varphi]]_e^M$. The LTS $M$ is typically understood from the context and so instead of writing $[[\varphi]]_e^M$, we only write $[[\varphi]]_e$, which is recursively defined in Fig. 1. In Fig. 1, $e[X \mapsto \hat{S}]$ denotes the environment $e$ with the substitution that associates the state set $\hat{S} \subseteq S$ with the variable $X \in \mathcal{X}$. In other words for $Y \in \mathcal{X}$, $e[X \mapsto \hat{S}](Y)$ equals $\hat{S}$ if $Y = X$, and $e(Y)$ otherwise. The environment is key to expressing the semantics of $\mu$-calculus, which uses explicit least and greatest fixed points.

The formulas $\mathtt{tt}$ and $\mathtt{ff}$ hold at all and no states, respectively. The semantics of propositions and of variables is

$$\begin{array}{rrcl}
1. & [\![\mathtt{tt}]\!]_e & = & S \\
2. & [\![\mathtt{ff}]\!]_e & = & \emptyset \\
3. & [\![p]\!]_e & = & \{s \mid p \in L(s)\} \\
4. & [\![X]\!]_e & = & e(X) \\
5. & [\![\varphi_1 \wedge \varphi_2]\!]_e & = & [\![\varphi_1]\!]_e \cap [\![\varphi_2]\!]_e \\
6. & [\![\varphi_1 \vee \varphi_2]\!]_e & = & [\![\varphi_1]\!]_e \cup [\![\varphi_2]\!]_e \\
7. & [\![\langle a\rangle\varphi]\!]_e & = & \{s \mid \exists s \xrightarrow{a} s' \wedge s' \in [\![\varphi]\!]_e\} \\
8. & [\![[a]\varphi]\!]_e & = & \{s \mid \forall s \xrightarrow{a} s' \Rightarrow s' \in [\![\varphi]\!]_e\} \\
9. & [\![\mu X.\varphi]\!]_e & = & f^{|S|}_{X,\varphi,e}(\emptyset), \\
 & \multicolumn{3}{l}{\text{where } \forall \hat{S} \subseteq S : f_{X,\varphi,e}(\hat{S}) := [\![\varphi]\!]_{e[X\mapsto \hat{S}]}} \\
10. & [\![\nu X.\varphi]\!]_e & = & f^{|S|}_{X,\varphi,e}(S), \\
 & \multicolumn{3}{l}{\text{where } \forall \hat{S} \subseteq S : f_{X,\varphi,e}(\hat{S}) := [\![\varphi]\!]_{e[X\mapsto \hat{S}]}}
\end{array}$$

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

determined using the labeling function and the environment respectively. The semantics of conjunctive and disjunctive formulas is given as the intersection and the union of the semantics of the sub-formulas, respectively. $\langle a\rangle\varphi$ holds at states which have at least one of its $a$-successor satisfying $\varphi$. Similarly, $[a]\varphi$ holds at a state if all its $a$-successor satisfy $\varphi$. The semantics of a fixed point formulas $\mu X.\varphi$ and $\nu X.\varphi$ are defined using the function $f_{X,\varphi,e} : 2^S \to 2^S$ that maps $\hat{S} \subseteq S$ to $[\![\varphi]\!]_{e[X\mapsto\hat{S}]}$. Observe that $f_{X,\varphi,e} : 2^S \to 2^S$ is monotonic over the complete lattice $(2^S, \subseteq)$. Initially the environment maps all variables bound by greatest fixed points to set of all states in $M$; similarly, the environment maps all variables bound by least fixed points to empty set (Rules 9 and 10). The mappings are iteratively refined using the semantics of the definition of the fixed point expression. The mapping of greatest fixed point variables at iteration $i$ is a subset of its mapping at iteration $i-1$; while the mapping of least fixed point variables at iteration $i$ is a superset of its mapping at the iteration $i-1$. The iterative refinement terminates when the mappings for all variables do not change in two successive iterations. The termination is guaranteed by Tarski-Knaster theorem [52]; in particular, any fixed point is reached in at most $|S|$ applications of $f_{X,\varphi,e}(\cdot)$.

We use LTSs to represent the discrete event systems (DES) expressing the plant model and the supervisor expressing the controller. Some subset of these LTSs are referred to as the initial or start states capturing the initial configurations of the plant and the supervisor. We will refer to these LTSs as *initialized LTSs*. An initialized LTS is satisfy a property $\varphi$ expressed in $\mu$-calculus if and only all initial states of the LTS belong to the semantics of $\varphi$.

## C. 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 $S_P, A, \delta_P$, $AP$, and $L_P$ have the usual semantics. We use $S_{0,P} \subseteq S_P$ to denote the set of start or initial states of the plant. Note that transitions from states can be controllable or uncontrollable. Hence, for each state $s$, the actions on the outgoing transitions are partitioned into two groups: the controllable action set $A_c(s)$, and the uncontrollable action set $A_u(s)$.

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

$$\{((s,q),\sigma,(s',q')) \mid (s,\sigma,s') \in \delta_P \wedge (q,\sigma,q') \in \delta_C\}.$$

$L_{P||C} : S_{PC} \to 2^{AP}$ 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 $s, s', s'', \ldots$ to represent states in the plant, $q, q', q'', \ldots$ 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 $q \xrightarrow{a} q'$).

The synchronous composition induces the control imposed by the supervisor on the plant. For instance, a composed state $(s,q)$ has an outgoing transition on an action only when the both the supervisor and the plan has the outgoing transition on the same action; otherwise, the transition is absent. That is, if a supervisor at the state $q$ wants to disallow a transition with action $a$ from the state $s$, then it simply does not have any evolution on action $a$. However, note that a supervisor cannot disallow any the uncontrollable actions in plant. In other words, if $a$ is an uncontrollable action from state $s$ and the supervisor state composed with $s$ is $q$, then $q$ is required to have an evolution on the action $a$. This requirement is referred to as the *control compatibility*.

## IV. Illustrative Example

We illustrate the salient features of our technique using a simple but representative problem involving controlling the moves of a cat and mouse in a maze Fig. 2(a) . The maze consists of several numbered rooms, which are connected by passage-ways/doorways—some accessible by the mouse (denoted by $m_i$) and some others accessible by the cat (denoted by $c_i$). All doorways have directionality and all, except $c_7$ are controllable. The objective is to generate a controller which appropriately controls (closes) the controllable doorways such that the cat and mouse (initially placed in rooms 2 and 4, respectively) never occupies the same room. The possible (unrestricted) movements of the cat and the mouse (Fig. 3) can be obtained by the asynchronous composition of the movements of the cat with the movements of the mouse (Fig. 2(b)), modeled as labeled transition systems. The nodes in the transition system denotes the rooms in which the entity resides and the directed-labeled edges denotes the movement of the entity from the source node/room to a destinate node/room via the doorway represented by the labels. The movements of the cat and the mouse forms the plant model, which in this example is non-deterministic as the cat can non-deterministically choose to move from room 0 to either room 1 or 3 via doorway $c_1$. As noted before, the supervisor is required to control the movement of the cat and the mouse such that they do not occupy the same room simultaneously. This requirement can be expressed in $\mu$-calculus as $\nu X.(p \wedge [-]X)$, where $p$ represents a proposition which is true only when the cat and the mouse are *not* in the same room.

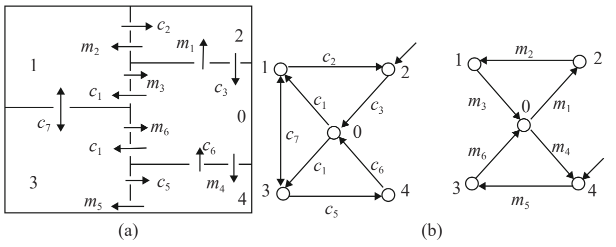

Fig. 2. (a) Cat-Mouse maze; (b) Cat-Mouse models.

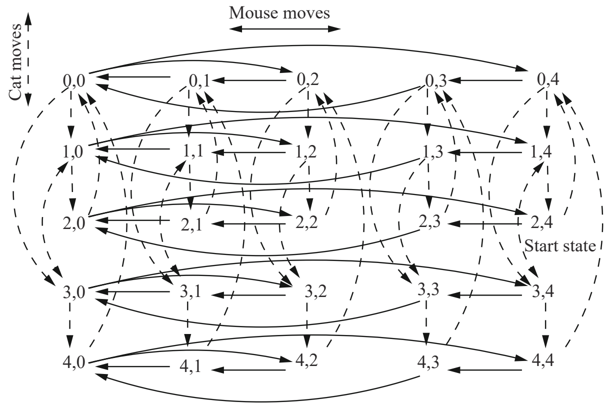

Fig. 3. Plant model for Cat-Mouse example in Fig. 2(b).

We use a short-hand notation $[-]$ to represent *any* action. The greatest fixed point formula represents the states where $p$ holds and this continues to remain true after any action.

The above can be viewed as *safety* requirement of the supervisor objective. The requirement can be further augmented to include *liveness* requirement that the cat and mouse is always able to return to their start state. This can be expressed using alternating fixed point formula: $\nu Y.(\mu Z.(q \vee \langle - \rangle Z) \wedge p \wedge [-]Y)$, where $p$ has the usual meaning and $q$ represents a proposition which is true only when the cat and the mouse are at their respective start states (i.e., the cat is in room 2 and the mouse is in room 4). The greatest fixed point formula over the variable $Y$ has a nested least fixed point formula over the variable $Z$. The least fixed point formula represents the states which can eventually reach the start state while the outer greatest fixed point formula ensures that the cat and the mouse are never in the same room.

## V. Quotienting $\mu$-Calculus Specifications

We present here a formal description of the problem at hand. Given a DES $P$ and a desired controlled behavior $\varphi^\circ$ of $P$ expressed in $\mu$-calculus, the problem is to

$$\exists C: \; P \| C \models \varphi^\circ$$

where $C$ denotes a control-comptabile supervisor, $P\|C$ denotes the composition of the plant with the supervisor, and $\models$ denote the relationship where the left-hand side is a satisfiable model for the property in the right-hand side. In other words, does there exist some supervisor under the presence of which the plan satisfies the desired behavior.

We reduce this problem as follows. The obligation on $P\|C$ is to satisfy $\varphi^\circ$. We transform that obligation to an obligation on on $C$; we refer to this new obligation as $\varphi^{\div}$. The transformation is such that

$$\exists C: \; P\|C \models \varphi^\circ \;\Leftrightarrow\; C \models \varphi^{\div}.$$

That is, satisfiability of the desired behavior by a controlled plan is reduced to the satisfiability of a new behavior by the supervisor alone. In other words, the plant is controllable (supervisor exists) to conform to desired behavior if and only if the transformed formula $\varphi^{\div}$ is satisfiable. This not only addresses the problem of whether a supervisor exists but also presents a roadway to generate a supervisor, if one exists. The satisfiable model for $\varphi^{\div}$ is one such supervisor.

To ensure the correctness and viability of this approach, we need to develop a technique for generating $\varphi^{\div}$ in the context $P$ and $\varphi^\circ$. This specification transformation is referred to as *quotienting* and is similar in flavor to that in [11], [12], [53] where it is applied for efficient model checking of synchronous systems with replicated sub-systems, hybrid systems, and infinite-state parametrized systems. Informally, quotienting identifies the "parts" of the specifications are satisfied by the component of a composition and leaves behind other parts of the specification that need to be satisfied by the rest of the composition. In our current setting, $\varphi^{\text{div}}$ is obtained by quotienting $\varphi^\circ$ against the component $P$ in the composition $P\|C$.

The result of quotienting is a new $\mu$-calculus formula expressing the temporal obligations of a supervisor controller. As this formula captures such obligations in the context of the states of the plant whose actions are being controlled, it is immediate that the fixed point variables in the resultant formula are parameterized by the state-information. More precisely, for every fixed point variable $X \in \mathcal{X}$ in the original formula, quotienting can generate a new fixed point variable of the form $X_{(s,k)}$, where $s \in S_P$ and $k \in \mathbb{N}$ (natural numbers). The necessity of $k$ steps from the fact that the same formula may be quotiented multiple times at that same plant state in different contexts; this will be elaborated in the discussion below. The set of these new variables is denoted as $\mathcal{X}_{(S_P \times \mathbb{N})}$. There is another type of fixed point variables induced by quotienting operation—these are necessary to capture control compatibility requirements. These are greatest fixed point variables of the form $Z_s$, where $s \in S_P$; the set of such variables is denoted by $\mathcal{Z}$. For the formula $\varphi^\circ$, we use $FV^\circ, Sub^\circ$, and $nd^\circ$ to represent free variables, subformulas and the nesting depth, respectively.

Finally, 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). For each $s \in S_P$, and $T \in \mathcal{T}$, we define a quotienting function $/_T(s)$ : $\Phi[AP, \mathcal{X}, A] \to \Phi[AP, \mathcal{X}_{(S_P, \mathbb{N})} \cup \mathcal{Z}, A]$ as shown in Fig. 4.

*Discussion:* Given a plant state $s$ and the property $\varphi$ to be enforced at that state, the quotienting operation, $(\varphi /_T s)$, generates a formula $\psi$. Any supervisor state (e.g., $q$) that satisfies $\psi$ is guaranteed to ensure that controlled plan state $(sq)$ satisfies $\varphi$. In the following, we present the rules for quotienting with justification of why each rule is valid.

As true is satisfied in any plant state $s$, Rule 1 in Fig. 4

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

Fig. 4. Quotienting rules.

captures that any supervisor state $q$ composed with $s$ necessarily satisfies the control compatibility requirement. That is, for any uncontrollable action $a \in A_u(s)$ from $s$ to $s'$, there is "matching" action from $q$ and the destination state (e.g., $q'$) satisfies the control compatibility requirement in the context of the state $s'$. This is represented by the conjunction of $\langle a\rangle$ modal formula ($a \in A_u(s)$). On the other hand, actions that are not in $A_u(s)$ may or may not be allowed by the supervisor. This is captured by the box-modality formula. Recall that the box-modality can be satisfied in the absence of the modal action. Finally, note that the result of quotienting is a greatest fixed point formula over a new variable $Z_s$. The tag set records that the formula $\mathtt{tt}$ is quotiented against $s$. The quotienting operation is recursive; the recursion terminates when $\mathtt{tt}$ is quotiented against $s$ more than one time, and the result is equal to the corresponding fixed point variable $Z_s$ as recorded in the tag set.

The formula $\mathtt{ff}$ is a contradiction and the hence the quotienting of $\mathtt{ff}$ against any plant state results in a formula $\mathtt{ff}$ indicating the non-existence of any supervisor state as well (Rule 2). Rule 3 states that the controlled plant state $(s,q)$ satisfies the atomic proposition $p$ if and only if $p$ is be satisfied by both $s$ and $q$; additionally, the supervisor state conforms to contral compatibility requirement ($(\mathtt{tt}{/_T}\, s)$). In Rule 4, quotienting of conjunctive formula is described as the conjunction of quotienting of the conjuncts. Rule 5 describes a similar rule for the disjunctive formula.

Rule 6 deals with diamond modality formula. A supervised plant state $(s,q)$ satisfies $\langle a\rangle\varphi$ if and only if both $s$ and $q$ have at least one $a$-successor leading $s'$ and $q'$, respectively such that the $(s',q')$ satisfies $\varphi$. In other words, if $s$ has multiple $a$ successors of the form $s'$, then the supervisor controlling the behavior of $s$ must also have an $a$ successor which conforms to the formula $\vee'_s(\varphi{/_T}\, s')$. If $s$ does not have any $a$-successor, then the formula cannot be enforced at $s$ by any supervisor (indicated by the result of quotient being $\mathtt{ff}$). The quotiented formula uses $(\mathtt{tt}{/_T}\, s)$ to ensure the control compatibility requirement is satisfied. The dual of Rule 6 is expressed in Rule 7.

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 nesting of fixed point formulas, 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, and we provide a value for such a bound (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 $X_{(s,j)}$ captures three features:

1) the variable $X$ that is bound by the fixed point in the formula being quotiented;

2) the state $s$ that is used to quotient the formula;

3) the number of times the formula is quotiented by $s$. This value is incremented appropriately by keeping track of the previous count in the tag set $T$.

Consider a least fixed point formula $\mu X.\varphi$ with no fixed point nesting. Its semantics is computed using the function $f_{X,\varphi,e}$ starting from bottom of the subset lattice $\emptyset$. It can be directly proved that if $(s_1,q_1) \in f_{X,\varphi,e}(\emptyset)$, i.e, $(s_1,q_1) \in [\![\varphi]\!]_{e[X\mapsto\emptyset]}$, then $q_1 \in [\![\varphi]\!]_{e[(X/s)\mapsto\emptyset]}$ for all $s$. Next suppose $(s_k,q_k)$ is present in the result of $f^{k-1}_{X,\varphi,e}(\emptyset)$ and is not in the result of computation of $f^{k-1}_{X,\varphi,e}(\emptyset)$. Let, $(s_{k-1},q_{k-1}) \in f^{k-1}_{X,\varphi,e}(\emptyset)$ leads to the inclusion $(s_k,q_k) \in f^{k}_{X,\varphi,e}(\emptyset)$, i.e., $(s_{k-1},q_{k-1}) \in e[X \mapsto f^{k-1}_{X,\varphi,e}(\emptyset)](X) \Rightarrow (s_k,q_k) \in [\![\varphi]\!]_{e[X\mapsto f^{k-1}_{X,\varphi,e}(\emptyset)]}$. From the quotienting rules for the non-fixed point formula expressions, when $\varphi$ is quotiented against $s_k$, $X$ is be quotiented against $s_{k-1}$

$$\begin{aligned} & (s_{k-1},q_{k-1}) \in e[X \mapsto f^{k-1}_{X,\varphi,e}(\emptyset)](X) \\ \Rightarrow\ & q_{k-1} \in \\ & e[(X/s_{k-1}) \mapsto \{q_{k-1} \mid (s_{k-1},q_{k-1}) \in \\ & e[X \mapsto f^{k-1}_{X,\varphi,e}(\emptyset)](X)\}](X/s_{k-1}) \\ \Rightarrow\ & q_k \in \\ & [\![(\varphi/s_k)]\!]_{e[(X/s_{k-1})\mapsto\{q_{k-1} \mid (s_{k-1},q_{k-1})\in e[X\mapsto f^{k-1}_{X,\varphi,e}(\emptyset)](X)\}]} \\ \Rightarrow\ & q_k \in (f/s_k)^k_{X,\varphi,e}(\emptyset) \end{aligned}$$

where $(f/s_k)^k_{X,\varphi,e}(\emptyset)$ is equal to

$$[\![(\varphi/s_k)]\!]_{e[(X/s_{k-1})\mapsto\{q_{k-1} \mid (s_{k-1},q_{k-1})\in e[X\mapsto f^{k-1}_{X,\varphi,e}(\emptyset)](X)\}]}.$$

As $f_{X,\varphi,e}$ is monotonic, the function $(f/s)_{X,\varphi,e}$ is also monotonic for all $s$. Therefore, quotienting a least (greatest) fixed point formula results in another least (greatest) fixed point formula. As such, the quotienting Rules 8 and 9 do not

Plant model: $S_P = \{s, s'\}$ and $s \xrightarrow{b} s, s \xrightarrow{a} s', s' \xrightarrow{b} s$. $A_u(s) = A_u(s') = \emptyset$.
Control specification: $\nu X.[b](\mu Y.([a]X \wedge [b]Y))$

$$
\begin{aligned}
&(\nu X.[b](\mu Y.([a]X \wedge [b]Y))/_{\emptyset} s)\\
&= \nu X_{(s,1)}.((tt/_{\{X_{(s,1)}\}} s) \wedge [b]((\underline{\mu Y.([a]X \wedge [b]Y)}/_{\{X_{(s,1)}\}} s))) \text{ Rules 8, 7}\\
&= \nu X_{(s,1)}.((tt/_{\{X_{(s,1)}\}} s) \wedge [b](\mu Y_{(s,1)}.([a](\underline{X}/_{\{X_{(s,1)},Y_{(s,1)}\}} s') \wedge \ldots))) \text{ Rules 8, 7}\\
&= \nu X_{(s,1)}.((tt/_{\{X_{(s,1)}\}} s) \wedge [b](\mu Y_{(s,1)}.([a](\underline{\nu X.[b](\mu Y.([a]X \wedge [b]Y))}/_{\{X_{(s,1)},Y_{(s,1)}\}} s') \wedge \ldots))) \text{ Rule 9}\\
&= \nu X_{(s,1)}.((tt/_{\{X_{(s,1)}\}} s) \wedge [b](\mu Y_{(s,1)}.([a]\nu X_{(s',1)}.[b]((\underline{\mu Y.([a]X \wedge [b]Y)}/_{\{X_{(s,1)},X_{(s',1)},Y_{(s,1)}\}} s)) \wedge \ldots))) \text{ Rules 8, 7}\\
&= \nu X_{(s,1)}.((tt/_{\{X_{(s,1)}\}} s) \wedge [b](\mu Y_{(s,1)}.([a]\nu X_{(s',1)}.[b](\mu Y_{(s,2)}.([a](\underline{X}/_{\{X_{(s,1)},X_{(s',1)},Y_{(s,2)}\}} s') \wedge \ldots)) \wedge \ldots))) \text{ Rules 8, 7}\\
&= \nu X_{(s,1)}.((tt/_{\{X_{(s,1)}\}} s) \wedge [b](\mu Y_{(s,1)}.([a]\nu X_{(s',1)}.[b](\mu Y_{(s,2)}.([a]X_{(s',1)} \wedge \ldots)) \wedge \ldots))) \text{ Rule 9}
\end{aligned}
$$

Fig. 5. Snippet showing application of Rules 7–9 of Fig. 4.

alter the fixed point nature of the formula being quotiented.

Next, let us consider the case where a formula variable is unfolded to its definition due to quotienting (Rule 9, case 3). Suppose for a fixed point formula $\sigma X.\varphi$, $(s,q) \in [[\varphi]]_{e[X \mapsto \hat{S}]}$ because $(s',q') \in \hat{S}$. Then from quotienting rules, when $\sigma X.\varphi$ is quotiented against $s$ for the first time, it generates a fixed point variable $X_{(s,1)}$ (Rule 8, case 2) and leads to the quotienting of $X$ in $\varphi$ against $s'$. If $s'$ equals $s$, then the quotienting process terminates with result $X_{(s,1)}$ (Rule 9, case 2); otherwise, the result is the quotienting of $\sigma X.\varphi$ against $s'$ (Rule 9, case 3). This follows from the fact that if $(s',q')$ is added in the semantics of $\sigma X.\varphi$ during the $i$th iteration of $f_{X,\varphi,e}$, then $(s,q)$ is added in the $j$-th iteration with $i < j$. This requires unfolding of the formula variable $X$ to $\sigma X.\varphi$ which leads to the quotienting of $\sigma X.\varphi$ against $s'$.

Next, let us consider the repeated quotienting of a fixed point formula against the same state. Let $\sigma_x X.\varphi_x$ be a fixed point formula where $\sigma_y Y.\varphi_y$ is a subformula of $\varphi_x$ and $X$ is a subformula of $\varphi_y$. Suppose $(s,q)$ belongs to the semantics of $\sigma_x X.\varphi_x$ because $(s',q')$ belongs to the semantics of $\sigma_y Y.\varphi_y$ and the latter holds because $(s'',q'')$ belongs to the semantics of $X$. This implies that iterative computation of fixed point of $\sigma_x X.\varphi_x$ includes $(s'',q'')$ before it includes $(s,q)$ and the computation depends on the fixed point computation of $\sigma_y Y.\varphi_y$ (as the latter is an inner fixed point formula). I.e., quotienting $\varphi_x$ against $s$ will result in quotienting of $\sigma_y Y.\varphi_y$ against $s'$ which in turn will result in quotienting of $X$ against $s''$. If $s''$ is different from $s$ then, following quotienting Rule 9, case 3, $s''$ will quotient $\sigma_x X.\varphi_x$. Furthermore, if the inclusion of $(s'',q'')$ also depends on $(s',q')$ satisfying $\sigma_y Y.\varphi_y$ then $\sigma_y Y.\varphi_y$ will again be quotiented against $s'$. In other words, the same state $s'$ will be used to quotient the same formula $\sigma_y Y.\varphi_y$ multiple times (as multiple iterations in the computation of semantics of $\sigma_x X.\varphi_x$ uses the semantic-computation of $\sigma_y Y.\varphi_y$). To keep track of these, new fixed point variables are generated when quotienting a fixed point formula multiple times against the same state (Rule 8, case 1).

Finally note that quotienting a free variable results in a new free variable (Rule 9, case 1). This completes the discussion about the various quotienting rules.

*Remark 1:* The first quotienting rule captures the fact that supervisor should allow all uncontrollable actions and can disallow some/all controllable actions. This is state-dependent controllability constraint. A simple modification to this rule can accommodate a generalized constraint $\psi_s$ that a supervisor state must satisfy when the plant is in state $s$; the modification being $(\mathtt{tt}/_T s) = \psi_s$.

*Example 1:* Fig. 5 shows the application of the above rules on a given plant model and a control specification. The plant model consists of two states $s$ and $s'$ where $s$ is the start state and three transitions: a self-loop at $s$ on action $b$, a transition from $s$ to $s'$ on action $a$ and a transition from $s'$ to $s$ on action $b$. The specification, described as an alternating fixed point $\mu$-calculus formula, states that after every $b$ action, there are infinitely many $(\geq 0)$ $a$ actions separated by finitely many $(\geq 0)$ $b$ actions. Quotienting of the formula against the plant state $s$ with an empty tag-set is obtained following the Rules 7 and 8. This results in a greatest fixed point formula expression over the variable $X_{(s,1)}$. The first conjunct in the definition of $X_{(s,1)}$ is equal to the result of quotienting $\mathtt{tt}$ against the state $s$ with the tag set $\{X_{(s,1)}\}$; while the second conjunct corresponds to the case where the supervisor is left with the obligation to satisfy a box-modality on action $b$ following which the formula resulting from the quotienting $\mu Y.([a]X \wedge [b]Y)$ against $s$ with tag set $\{X_{(s,1)}\}$ must be satisfied.

For brevity, the subsequent steps show the quotienting operation on certain sub-formulas; the sub-formulas being quotiented are underlined. For example, the next step shows the quotienting of $\mu Y.([a]X \wedge [b]Y)$. A new least fixed point formula variable $Y_{(s,1)}$ is generated and its definition follows from the quotienting Rules 7 and 8. Quotienting $[a]X$ against $s$ results in $[a]$ modal obligation for the supervisor (note $s \xrightarrow{a} s'$); the obligation for all $a$-successors being the formula generated by quotienting $X$ with $s'$. In the next step owing to the fact that $X$ has not been quotiented against $s'$ yet (Rule 9, case 3), $X$ is expanded with its definition. The quotienting operation is continued and $\mu Y([a]X \wedge [b]Y)$ is quotiented again against $s$. At this point, the tag set contains $Y_{(s,1)}$, and so the new fixed point variable generated is $Y_{(s,2)}$ and the tag set is appropriately updated (Rule 8, case 2). Finally, the variable $X$ inside the definition of $Y$ is quotiented for the second time against $s'$ and the result is $X_{(s',1)}$ (Rule 9, case 2).

*Example 2:* Fig. 6 shows the quotienting of the control specification of Section IV against two of the states of the cat-mouse plant model. The figure also presents quotienting of $\mathtt{tt}$ against a plant state. The plant states are of the form $s_{ij}$ representing the state when the cat is in room $i$ and the mouse is in room $j$. Proposition $p$ is true in the states $s_{ij}$ with $i \neq j$.

$$
\begin{aligned}
&(a)\ (\nu X.(p \wedge [-]X)/_{\emptyset} s_{24})\\
&= \nu X_{s_{24},1}.((p/_{\{X_{s_{24},1}\}} s_{24}) \wedge ([-]X/_{\{X_{s_{24},1}\}} s_{24}))\\
&= \nu X_{s_{24},1}.(p \wedge [c_3](tt/_{\{X_{s_{24},1}\}} s_{04}) \wedge [m_5](tt/_{\{X_{s_{24},1}\}} s_{23}) \wedge\ [c_3](X/_{\{X_{s_{24},1}\}} s_{04}) \wedge [m_5](X/_{\{X_{s_{24},1}\}} s_{23}))\\
&= \nu X_{s_{24},1}.(p \wedge [c_3](tt/_{\{X_{s_{24},1}\}} s_{04}) \wedge [m_5](tt/_{\{X_{s_{24},1}\}} s_{23}) \wedge [c_3](\nu X.(p \wedge [-]X)/_{\{X_{s_{24},1}\}} s_{04})\\
&\qquad \wedge [m_5](\nu X.(p \wedge [-]X)/_{\{X_{s_{24},1}\}} s_{23}))\ )
\end{aligned}
$$

$$
\begin{aligned}
&(b)\ (\nu X.(p \wedge [-]X)/_{T} s_{04})\\
&= \nu X_{s_{04},1}.((p/_{T\cup\{X_{s_{04},1}\}} s_{04}) \wedge ([-]X/_{T\cup\{X_{s_{04},1}\}} s_{04}))\\
&= \nu X_{s_{04},1}.(p \wedge [c_1]((tt/_{T\cup\{X_{s_{04},1}\}} s_{14}) \wedge (tt/_{T\cup\{X_{s_{04},1}\}} s_{34})) \wedge\ [c_1]((X/_{T\cup\{X_{s_{04},1}\}} s_{14}) \wedge (X/_{T\cup\{X_{s_{04},1}\}} s_{34}))\\
&\qquad \wedge [m_5](tt/_{T\cup\{X_{s_{04},1}\}} s_{03}) \wedge [m_5](X/_{T\cup\{X_{s_{04},1}\}} s_{03}))
\end{aligned}
$$

$$
\begin{aligned}
&(c)\ (tt/_{T} s_{14})\\
&= \nu Z_{s_{14}}.(\langle c_7\rangle(tt/_{T\cup\{Z_{s_{14}}\}} s_{34}) \wedge [c_2](tt/_{T\cup\{Z_{s_{14}}\}} s_{24}) \wedge [m_5](tt/_{T\cup\{Z_{s_{14}}\}} s_{13}))\\
&= \nu Z_{s_{14}}.(\langle c_7\rangle \nu Z_{s_{34}}.(\langle c_7\rangle(tt/_{T\cup\{Z_{s_{14}},Z_{s_{34}}\}} s_{14})) \wedge\ [c_2](tt/_{T\cup\{Z_{s_{14}}\}} s_{24}) \wedge [m_5](tt/_{T\cup\{Z_{s_{14}}\}} s_{13}))\\
&= \nu Z_{s_{14}}.(\langle c_7\rangle \nu Z_{s_{34}}.(\langle c_7\rangle Z_{s_{14}}) \wedge\ [c_2](tt/_{T\cup\{Z_{s_{14}}\}} s_{24}) \wedge [m_5](tt/_{T\cup\{Z_{s_{14}}\}} s_{13}))
\end{aligned}
$$

Fig. 6. $\nu X.(p \wedge [-]X)$ quotiented against (a) $s_{24}$ and (b) $s_{04}$ (with non-determinism on $c_1$); (c) $tt$ quotiented against $s_{14}$.

Fig. 7 shows the recursions of the quotienting operations. For brevity, we have omitted the tag-sets and the formula expressions for the control-compatibility. The node enclosed within a square box denotes the termination of quotienting, for the reason that the definition of $X$ has already been quotiented against $s_{34}$ (Rule 9, case 2).

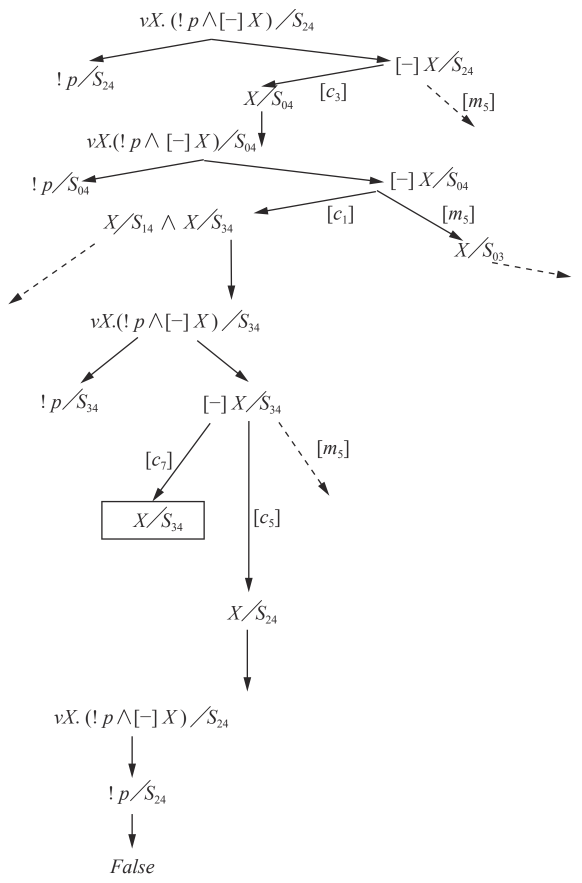

Fig. 7. Quotienting recursion snapshot in Cat-Mouse model.

We have the following theorems establishing the termination 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:* The theorem identifies an upper bound for $k$ appearing in a fixed point variable, $X_{(s,k)}$ of the quotiented formula. 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 $f(n)$ (induction hypothesis). 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, $\forall i \ge 1. f(i) = |S_P|^i$. ■

*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 plant state $(s, q)$ satisfies $\varphi^\circ$ if and only if the supervisor state $q$ satisfies $(\varphi^\circ /_{\emptyset} s)$.

*Proof:* The proof proceeds by translating a $\mu$-calculus formula $\varphi$ into its corresponding equational form and presenting the semantics of equational $\mu$-calculus formulas using [51]. See Appendix A for details. ■

## VI. Satisfiability Checking and Model Discovery

In this section, we focus on verifying the satisfiability of $\mu$-

calculus formula $\varphi^\diamond \in \Phi[AP, \mathcal{X}, A]$. If the formula is satisfiable, we also develop a model witnessing the satisfiability. This technique will be used to generate/identify the supervisor.

*Preliminaries:* Recall that, the $\mu$-calculus formula expresses temporal ordering using explicit greatest and least fixed points, and these fixed point sub-formulas can have arbitrary nesting. We assign identifiers to each fixed point variable based on their binding and nesting depth.

$$id(X) := \begin{cases} 2 \times ad(\sigma X.\varphi_x) & \text{if } \sigma = \nu \\ 2 \times ad(\sigma X.\varphi_x) - 1 & \text{otherwise.} \end{cases}$$

The $id$ of greatest fixed point variables are even, while the $id$ of least fixed point variables are odd. The $id$ of the variable bound by the outer-most fixed point is the largest.

## A. Tableau-Based Approach

We present a set of implications, which if valid, establishes not only the satisfiability of the $\mu$-calculus formula but also helps identify a satisfiable model. These implications form a tableau written as

$$\frac{A}{A_1\ A_2\ A_3 \cdots A_n}$$

i.e., $A_1 \wedge A_2 \wedge \ldots A_n \Rightarrow A$. In other words, in order to prove the validity of $A$, we need to verify the validility of $A_1, A_2, \ldots, A_n$. Given a obligation to prove some claim $A$, the tableau induces a proof-tree, where the nodes in the tree represents the obligations and sub-obligations (nemerator and denominators of the tableau rule) and edges represent the dependency (conjunctive). A proof tree successfully validates a claim (at its root) if all its leaf nodes are valid.

In our setting, each tableau rule is of the form

$$\frac{C^0_{\mathcal{H}^0}\ \ M^0}{C^1_{\mathcal{H}^1}\ \ M^1\ \ C^2_{\mathcal{H}^2}\ M^2\ \ldots\ C^n_{\mathcal{H}^n}\ M^n}.$$

Here, for all $i \in [0, n]$ "$C^i_{\mathcal{H}^i}\ \ M^i$" is referred to as a node of the tableau; In particular, "$C^0_{\mathcal{H}^0}\ \ M^0$" is the numerator node of a tableau rule while "$C^i_{\mathcal{H}^i}\ \ M^i$" $(1 \le i \le n)$ pairs are referred to as the denominator nodes.

The $C'$s is a set of elements of the form $(\varphi, \overrightarrow{X}, \overrightarrow{N})$, where $\varphi$ is a $\mu$-calculus formula, $\overrightarrow{X} \in \mathcal{X}^*$ represents a sequence of fixed point variables and $\overrightarrow{N} \in \mathbb{N}^*$ is a sequence of integers. Each set $C$ is annotated with a history $\mathcal{H}$, which keeps track of the association of formula set with model-states $(\{(C^j, s^j)\})$.

As noted before, the validity of the formula-set $C^0$ in the numerator node requires the validity of all the formula-sets $C^i$'s in the denominator nodes. In particular, if $C^0$ contains the set $\{(\varphi_{0i}, \overrightarrow{X}_{0i}, \overrightarrow{N}_{0i}) \mid i \in [1, k]\}$, then the validation objective is to determine whether the conjunction of $\mu$-calculus formula $\bigwedge_{i \in [1,k]} \varphi_{0i}$ is satisfiable. While this validity is confirmed in a bottom-up fashion by generating obligations for the denominator from the obligations of the numerator, the models witnessing the validity (if validation is successful) are generated (partially) in a top-down fashion. These associated models $M'$s are expressed using a prefix notation to allow for such partial representation. There are two constant models $M_{\mathtt{tt}}$ is simply a single state (representing a true-model) while $M_{\mathtt{ff}}$ represents a model with no state (a false-model). In general, a model expression takes the form $s * [\bigwedge_i (a_i : M_i)]$, meaning $M$ is rooted at state $s$ possessing for each $i$ an $a_i$-transition to the root of model $M_i$. In case $M_i = M_{\mathtt{ff}}$ for some $i$, then $s * [\bigwedge_i (a_i : M_i)]$ is equivalent to $M_{\mathtt{ff}}$. Fig. 8 presents the tableau rules.

1. $\dfrac{[\{(\mathtt{tt}, \overrightarrow{X}, \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M}{C_{\mathcal{H}}\quad M}$ 2. $\dfrac{[\{(\mathtt{ff}, \overrightarrow{X}, \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M{=}M_{\mathtt{ff}}}{\bullet}$

3. $\dfrac{[\{(p, \overrightarrow{X}, \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M{=}s{*}B}{C_{\mathcal{H}}\quad M{=}s{*}B}$ where $p \in L(s)$

4. $\dfrac{\{\}_{\mathcal{H}}\quad M{=}M_{\mathtt{tt}}}{\bullet}$ 5. $\dfrac{[\{(\varphi \wedge \psi, \overrightarrow{X}, \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M}{[\{(\varphi, \overrightarrow{X}, \overrightarrow{N}), (\psi, \overrightarrow{X}, \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M}$

6. $\dfrac{[\{(\varphi \vee \psi, \overrightarrow{X}, \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M}{[\{(\varphi, \overrightarrow{X}, \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M}$ 7. $\dfrac{[\{(\varphi \vee \psi, \overrightarrow{X}, \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M}{[\{(\psi, \overrightarrow{X}, \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M}$

8. $\dfrac{[\{(\sigma X.\varphi, \overrightarrow{X}, \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M}{[\{\varphi, \overrightarrow{X}, \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M}$

9. $\dfrac{[\{(X, \overrightarrow{X}, \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M}{C_{\mathcal{H}}\quad M}$ where $X \in FV^\diamond$

10. $\dfrac{[\{(X, \overrightarrow{X}, \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M}{[\{(\sigma X.\varphi, (X.\overrightarrow{X}), \overrightarrow{N})\} \cup C]_{\mathcal{H}}\quad M}$ $\sigma X.\varphi \in Sub^\diamond$

11. $\dfrac{[([a]\varphi, \overrightarrow{X}, \overrightarrow{N}) \cup C]_{\mathcal{H}}\ M}{[(([a]\varphi \wedge \langle a \rangle \mathtt{tt}), \overrightarrow{X}, \overrightarrow{N}) \cup C]_{\mathcal{H}}\ M}$

12. $\dfrac{[([a]\varphi, \overrightarrow{X}, \overrightarrow{N}) \cup C]_{\mathcal{H}}\ M}{[C - \{([a]\varphi_i, \overrightarrow{X_i}, \overrightarrow{N_i}) \mid ([a]\varphi_i, \overrightarrow{X_i}, \overrightarrow{N_i}) \in C\}]_{\mathcal{H}}\ M}$

if $C = \{(\psi, \overrightarrow{X_k}, \overrightarrow{N_k}) \mid \psi \in \{\langle\![a_i]\!\rangle \psi_i\}\}$,
$\nexists(C' = \{([a]\varphi, \overrightarrow{X'}, \overrightarrow{N'})\} \cup C, s) \in \mathcal{H}$ and
$\nexists(\langle a \rangle \varphi_j, \overrightarrow{X_j}, \overrightarrow{N_j}) \in C$

13. $\dfrac{C_{\mathcal{H}}\ \ M}{C^{a,1}_{\mathcal{H}'}\quad M^{a,1} \ldots\ C^{a,n}_{\mathcal{H}'}\quad M^{a,n}}$

if $C = \{(\langle\![a_i]\!\rangle \varphi_i, \overrightarrow{X}_i, \overrightarrow{N}_i)\}$, $\nexists(C' = \{(\langle\![a_i]\!\rangle \varphi_i, \overrightarrow{X'_i}, \overrightarrow{N'_i})\}, s) \in \mathcal{H}$
and $\forall([a]\varphi_i, \overrightarrow{X_i}, \overrightarrow{N_i}) \in C. \exists(\langle a \rangle \varphi_j, \overrightarrow{X_j}, \overrightarrow{N_j}) \in C$, in which case,

$M := s_C * \bigwedge_{a,j} a : M^{a,j}$
$C^{a,j} := \{(\varphi_i, \overrightarrow{X}_i, i.\overrightarrow{N}_i) \mid ([a]\varphi_i, \overrightarrow{X}_i, \overrightarrow{N}_i) \in C\} \cup \{(\varphi_j, \overrightarrow{X}_j, j.\overrightarrow{N}_j)\}$,
such that $(\langle a \rangle \varphi_j, \overrightarrow{X}_j, \overrightarrow{N}_j) \in C$
$\mathcal{H}' := \mathcal{H} \cup \{(C^{\langle\![\,]\!\rangle}, s_C)\}$,
where $C^{\langle\![\,]\!\rangle} := \{(\langle\![a]\!\rangle \varphi_i, \overrightarrow{X}_i, i.\overrightarrow{N}_i) \mid (\langle\![a]\!\rangle \varphi_i, \overrightarrow{X}_i, \overrightarrow{N}_i) \in C\}$

14. $\dfrac{C_{\mathcal{H}}\quad M}{\bullet}$

if $C = \{(\langle\![a_i]\!\rangle \varphi_i, \overrightarrow{X}_i, \overrightarrow{N}_i)\}, \exists(C' = \{(\langle\![a_i]\!\rangle \varphi_i, \overrightarrow{X'_i}, \overrightarrow{N'_i})\}, s) \in \mathcal{H}$
in which case,

$M := \begin{cases} M_{\mathtt{ff}} & \text{if } \mathtt{lfp}(C, C') \\ s & \text{otherwise} \end{cases}$ , where

$\mathtt{lfp}(C, C')$ is a Boolean expression that holds iff
$\exists i_0, i_1, \ldots, i_n = i_0 : (\forall j \in [0, n-1] : \overrightarrow{N'_{i_j}} \in \mathtt{suff}(\overrightarrow{N_{i_{j+1}}})$,
and $\mathtt{max}\{id(X) \mid X \in \overrightarrow{X_{i_{j+1}}} / \overrightarrow{X'_{i_j}}, j \in [0, n-1]\}$ is odd

Fig. 8. Tableau for satisfiability and model discovery for $\varphi^\diamond$.

*Discussion:* If the objective is to verify the satisfiability of $\varphi^\diamond$, a tableau-tree is constructed rooted at the node "$\{(\varphi^\diamond, \epsilon, \epsilon)\}_\emptyset\ M$". The children of the root are identified by "firing" the tableau-rule whose numerator matches with the root. This firing may result in multiple new nodes in the tableau tree, and each of these new nodes may lead to new nodes by firing new tableau-rules. Note no successor is created for a tableau-rule whose denominator is empty (denoted as a "$\bullet$"). When a tableau-node does not have any children (leaf-node), the model associated with that tableau-node is assigned a concrete value; and this results in a path in the model expression starting from the root of the tableau to the leaf-node. Once the iterative tableau-node creation terminates along all branches of the tableau tree, the root node gets associated with a concrete model. At this point, the formula at the root, $\varphi^\diamond$, is satisfiable if and only if the model expression $M$ of the root node is not $M_{\texttt{ff}}$ (see Theorem 3).

Note that, the set $C$ correspond to conjunction of formula. As a result Rule 1 states that if one of the conjuncts is $\texttt{tt}$, then obligation is to satisfy the rest of the conjuncts. On the other hand, Rule 2 states that only a false model $M_{\texttt{ff}}$ can witness the satisfiability of a conjunctive formula where one of the conjuncts is $\texttt{ff}$.

Rule 3 imposes the obligation that $p \in L(s)$ for the model rooted at state $s$ to ensure the satisfiability of a conjunct which is a proposition $p$. The denominator considers the satisfiability of the rest of the formula. 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 captures the scenario where there is no obligation for proving satisfiability and therefore, the true model (or any model) can be used as a witness.

Rules 5–7 deal with conjunctive and disjunctive formula. Note that Rules 6 and 7 result in two different tableau-tree depending on the consideration of left- or right-conjunct.

Rule 8 states that $M$ satisfies a fixed point formula $\sigma X.\varphi$ if and only if it satisfies $\varphi$. Rule 9 corresponds the scenario where one of the conjuncts is a free variable. As any suitable mapping of free variables to sets of states (for non-false models) can be generated, the denominator of the tableau rule simply discards the free variable. In typical scenario, formulas will not involve free variables (all variables will be bound by some fixed point expression).

Rule 10 is similar but applies to a conjunction of formula expressions one of which is a fixed-point bound variable. In this case, the variable is replaced by its binding expression in the denominator and the variable $X$ is pre-pended to the sequence $\overrightarrow{X}$ of the corresponding tuple. This is to keep track of the sequence of fixed-point variables visited thus far.

Rules 11–14 apply when all the formula expressions in the numerator $C$ are modal formula expressions. 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. Recall that, there are two ways to satisfy a box-modal formula $[a]\varphi$. Either every $a$ transition leads to destinations that satisfy $\varphi$, or there is no $a$ transition. Accordingly, Rule 11 captures the first scenario; and Rule 12 considers the second scenario. Note that, in Rule 12 all box-modal formula on the same box-modal actions are removed from the formula set in the denominator.

Rule 13 is applied where every box-modal action has a corresponding diamond modal obligation. It states that for any action $a$, a set of formula expression of the form $\{\langle\![a]\!\rangle\varphi\}$ 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. Accordingly for each action $a$, a denominator node aggregates all the box-obligations and one diamond obligation (see definition of $C^{a,j}$ in Rule 13 of Fig. 8 ). Finally the history is augmented to record i) $C$, modified to include the ancestor node tag by pre-pending $\overrightarrow{N_i}$ 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 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).

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 $\texttt{lfp}(C, C')$ 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 $\texttt{lfp}$ evaluates to true, then the model $M$ is set to $M_{\texttt{ff}}$; otherwise it is set equal to the state $s$ corresponding to the element $C'$ in $\mathcal{H}$.

*Example 3:* Table I presents the snapshot of the tableau for identifying a model satisfying the formula: $\nu X.\mu Y.([a]X \wedge [b]Y)$. The formula expressions at node A1 in the tableau are the same as the formula expressions at node A0. The formulas at A1 originated from the formula $[a]X$ at A0 (see history set at A1). The outermost fixed point variable expanded between these tableau-nodes is $X$, the greatest fixed point variable. As the such the model state at node A1 is the same as the model state at node A0, namely state $s$.

TABLE I
SNAPSHOT OF A TABLEAU

| | | |
|---|---|---|
| | Given formula equations | |
| | $\nu X.\mu Y.([a]X \wedge [b]Y) \quad id(X) := 4, id(Y) := 1$ | |
| | $\{\nu X.\mu Y.([a]X \wedge [b]Y), \epsilon, \epsilon\}_\emptyset \;\; M$ | *Rule 8* |
| | $\{([a]X \wedge [b]Y), \epsilon, \epsilon\}_\emptyset \;\; M$ | *Rule 5* |
| | $\{([a]X, \epsilon, \epsilon), ([b]Y, \epsilon, \epsilon)\}_\emptyset \;\; M$ | *Rules 11 and 12* |
| **A0:** | $\{([a]X, \epsilon, \epsilon), (\langle a\rangle\texttt{tt}, \epsilon, \epsilon)\}_\emptyset \;\; M := s * a : M_1$ | *Rule 13* |
| | $\{(X, \epsilon, 1), (\texttt{tt}, \epsilon, 2)\}_{\{(\{([a]X,\epsilon,1),(\langle a\rangle\texttt{tt},\epsilon,2)\},s)\}} \;\; M_1$ | *Rules 1 and 10* |
| | $\{(\mu Y.([a]X \wedge [b]Y), X, 1)\}_{\{(\{([a]X,\epsilon,1),(\langle a\rangle\texttt{tt},\epsilon,2)\},s)\}} \;\; M_1$ | *Rule 8* |
| | $\{([a]X \wedge [b]Y, X, 1)\}_{\{(\{([a]X,\epsilon,1),(\langle a\rangle\texttt{tt},\epsilon,2)\},s)\}} \;\; M_1$ | *Rule 5* |
| | $\{([a]X, X, 1)([b]Y, X, 1)\}_{\{(\{([a]X,\epsilon,1),(\langle a\rangle\texttt{tt},\epsilon,2)\},s)\}} \;\; M_1$ | *Rules 11 and 12* |
| **A1:** | $\{([a]X, X, 1), (\langle a\rangle\texttt{tt}, X, 1)\}_{\{(\{([a]X,\epsilon,1),(\langle a\rangle\texttt{tt},\epsilon,2)\},s)\}} \;\; M_1 := s$ | *Rule 14* |
| | • | |

*Theorem 3:* Given a $\mu$-calculus formula $\varphi^\diamond$, it is satisfiable if and only if there exists a tableau with root node "$\{(\varphi^\diamond, \epsilon, \epsilon)\}_\emptyset \;\; M$", such that $M$ is assigned to a non false-model.

*Proof:* The completeness of our satisfiability checking follows from the fact there is one tableau rule for each syntactic construct of a $\mu$-calculus formula. Then to show the soundness of our satisfiability check it suffices to show the soundness of each of the tableau rules. The soundness of the tableau rules follows from the semantics of the $\mu$-calculus formula (Fig. 1). The soundness of the Rules 1–13 can be realized directly from the discussions given preceding the theorem statement.

The soundness of Rule 14 is more involved as it depends on the semantics of fixed points. Consider first the semantics of a least fixed point formula $\mu X.\varphi$, which is the smallest state-set satisfying $\varphi$. A least fixed point formula $\mu X.\varphi$ is satisfiable if and only if $[[\varphi]]_{e[X\mapsto\emptyset]}$ is non-empty. In other words, $\mu X.\varphi$ is satisfiable if and only if the model for $\varphi$ is a non false-model when the model corresponding to $X$ inside $\varphi$ is $M_{\mathtt{ff}}$. Rule 14 in the tableau captures precisely this fact and states that if the outermost formula variable expanded in a tableau starting and ending in the same tableau node is a least fixed point variable, then model corresponding to the later node is a false-model. A dual property holds for $\nu X.\varphi$, namely the formula is unsatisfiable if and only if $[[\varphi]]_{e[X\mapsto S]}$ is empty.

The main difference between the models of least and greatest fixed point formulas is that in case of former, the model must have a finite path satisfying the least fixed point formula while in the latter the model may contain a loop to satisfy the greatest fixed point formula. In case of greatest fixed point, Rule 14 identifies this looping structure. The loop starts and ends at the tableau-node satisfying the formula expressions that led to repetition of the tableau nodes (repetition resulted from expansion of outermost greatest fixed point variable). ■

## B. 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 $\varphi^\div$ 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 $O(max_a d^a) \le O(d)$. 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 constraint is state-independent, Rule 1 can be simplified as

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

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

We next consider the complexity of satisfiability checking and model discovery for the quotiented formula $\varphi^\div$, which considers at each of its nesting level, all possible subsets of the subformulae of $\varphi^\div$. At each nesting level the number of possible subsets of the subformulae $\varphi^\div$ 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 $O(|S_P|^{nd^\circ}\times 2^{(1+d)\times|\varphi^\circ|+|A|})$ when the controllability is state-independent. Note that this is polynomial in the number of plant states $S_P$.

## VII. Implementation

A prototype implementation for performing the quotient operation and for checking satisfiability and identifying a supervisor model has been realized in XSB, a tabled logic programming language [54]. The tabling feature in XSB is used to avoid repeated subcomputation in addition to computing the least model of normal logic programs Predicates or relations are defined as logical rules in XSB in the following manner:

$$\mathtt{Goal} : -\mathtt{SubGoal}_1, \mathtt{SubGoal}_2, ..., \mathtt{SubGoal}_N.$$

The predicate `Goal` evaluates to true only when each of the subgoals in the right-hand side of ":-" evaluates to true. In essence, the above logical rule represents

$$\mathtt{SubGoal}_1 \wedge \mathtt{SubGoal}_2 \wedge \ldots \mathtt{SubGoal}_N \Rightarrow \mathtt{Goal}.$$

A rule with empty right-hand side is referred to as a *fact*.

*XSB Encoding of DES Problem:* Models in supervisory control problem, represented as labeled transition systems, are encoded by rules and facts in XSB:

1) `ctrans(S, A, T)`, denotes a transition of a component in a plant model from a state `S` to a state `T` via an action `A`,

2) `trans(S, A, T)`, denotes a transition from a plant state `S` to a plant state `T` via an action `A`,

3) `startstate(S)`, denotes the fact that `S` is a start state of the considered plant model,

4) `label(S, P)`, denotes the labeling of a plant state `S` with a proposition `P`.

5) `uncontrollable(S, AList)`, states that the actions in `AList` are not controllable at a state `S`.

The encoding of the models for the cat-mouse example (Fig. 2) is shown in Appendix B.

The plant model is defined by the product of its components, i.e., a transition in plant model corresponds to a transition in one of the participating components. A state in a plant model is represented as a list of component states. Appendix B shows the encoding of the transitions of the plant model derived from the encoding of the transitions in its components. The predicate `pickone` selects a component state `S1` from the plant state `S`. For a transition on an action `A`, the destination plant-state `T` is reconstructed form the destination component-state `T1` using the predicate `putback`. The start state of the plant is encoded as a fact that initially the cat and the mouse are in rooms 2 and 4 of the maze respectively.

Formula definition is represented by a term `fDef` with three arguments representing the fixed point variable, fixed point operator and the body of the definition. For example, the formula $\nu X.(p \wedge [-]X)$ is represented as

```
fDef(x,nu,and(prop(p),fBox(_,form(x)))).
```

In the above, the terms `and` and `fBox` are used to capture the boolean connective $\wedge$ and box modal operator $[\,]$ respectively. Similar terms (`or` and `fDiam`) are used for capturing the dual operators. The term `prop` is used to denote the atomic propositions. Finally, "`_`" in `fBox(_,form(x))` represents *any* action.

*XSB Encoding of Quotienting Rules:* Quotienting rules of the form $(\varphi /_{T} s) = \psi$ are encoded as logical rules using `quot(S, Tag, Phi, Psi)`, where `S` represents the state $s$, `Tag` is $T$, and `Phi` and `Psi` are input and output formula expressions $\varphi$ and $\psi$ respectively.

The following shows a snapshot of compiling and executing the quotienting program in XSB:

```
| ?- [quot].
[quot loaded]
[catmouse.P dynamically loaded,
cpu time used: 0.0040 seconds]
yes
| ?- startstate(S),
   fDef(x, Sigma, XDef),
   quot(S, [], fDef(X, Sigma, XDef), QRes).
S = [cat(2),mouse(4)]
Sigma = nu
XDef = and(neg(prop(p)), fBox(_,form(x)))
QRes = fDef(x(1,x([cat(2),mouse(4)],1))).
```

In the above, `quot` is the main program file which contains the quotienting rules and the directives to include the plant model file (e.g., `catmouse.P` containing the model for the plant). The result of quotienting is obtained via grounding of variable `QRes` which holds the valuation of the outermost fixed point variable of the quotient. The actual formula expressions are asserted as facts and can be viewed using "`listing(fDef)`".

*Encoding Tableau-Rules For Satisfiability Checking and Model Discovery:* The tableau-rules for satisfiability checking and model discovery are encoded as logical relations in XSB. Specifically, for the tableau-rule of the form $\frac{\alpha}{\beta_1,\beta_2,\ldots,\beta_n}$, the encoding is

$$\alpha : -\beta_1,\beta_2,\ldots,\beta_n.$$

The predicate `genmodel(SetOfFormula,History,Model)` represents the tableau node $C_{\mathcal{H}}\ M$ in Fig. 8 , where `SetOfFormula` represents $C$, and `History` and `Model` represent the associated history set $\mathcal{H}$ and the model $M$ respectively.

For details of implementation and tool documentation see http://www.cs.iastate.edu/~sbasu/control-quot.

## A. Formula Simplification

Formula generated via quotienting can be prohibitively large. We use a number of simplification rules following the semantics of the $\mu$-calculus formula expressions, that reduces the length of the generated formula. The simplification rules are as follows:

$$\varphi \wedge \texttt{ff} = \texttt{ff} \quad \varphi \vee \texttt{tt} = \texttt{tt} \quad \langle a \rangle \texttt{ff} = \texttt{ff}$$
$$\mu X.(X \vee \varphi) = \mu X.\varphi \quad \mu X.(X \wedge \varphi) = \texttt{ff}$$
$$\nu X.(X \vee \varphi) = \texttt{tt} \quad \nu X.(X \wedge \varphi) = \nu X.\varphi.$$

These simplification rules are applied on-the-fly, e.g., $((\varphi \wedge \psi) /_{T} s)$ first computes the $(\varphi /_{T} s)$ and if the result is `ff`, $(\psi /_{T} s)$ is not computed.

## B. Model Simplification

The models generated using tableau rules can be simplified by merging bisimilar states in the model. Bisimulation equivalence [55] states that two states $s_1$ and $s_2$ are equivalent if they are related by the largest bisimilarity relation $\mathcal{R}$ defined as follows:

$$s_1 \mathcal{R} s_2 \Rightarrow (\forall s_1 \xrightarrow{a} t_1 : \exists s_2 \xrightarrow{a} t_2 : t_1 \mathcal{R} t_2) \wedge s_2 \mathcal{R} s_1.$$

We use the above relation to identify equivalent states in the identified model and simplify the model to contain a single state from each equivalence class.

## VIII. Experimental Result

We revisit the cat-mouse example from Section IV. The specification formula $\nu X.(p \wedge [-]X)$ is quotiented against the plant model and the tableau-based model discovery algorithm is applied on the quotiented formula to obtain a candidate supervisor. The supervisor model we obtained is presented in Fig. 9(a). In the figure, the states in the supervisor represent the corresponding rooms in which the cat and the mouse are present. Note that, when the cat and the mouse are in rooms 0 and 4 respectively, the supervisor can allow the cat-move $c_1$. As this is a non-deterministic transition for the cat, the successor supervisor state designates that the cat can be either in room 1 or 3.

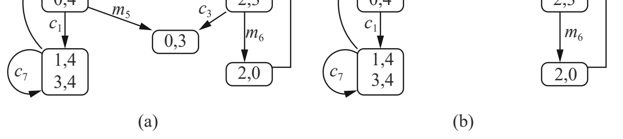

Fig. 9. Supervisors for plant in Fig. 2 for specifications (a) $\nu X.(p \wedge [-]X)$ and (b) $\nu Y.(\mu Z.(q \vee \langle - \rangle Z) \wedge p \wedge [-]Y)$.

For finding a supervisor for the more general specification $\nu Y.(\mu Z.(q \vee \langle - \rangle Z) \wedge p \wedge [-]Y)$, we quotiented the formula against the plant model and using the tableau-based model discovery algorithm to obtain a supervisor (Fig. 9(b)), which can be seen to be less permissive than the previous one, as expected. Since the specification demands that the controlled plant must ensure that the start state is always reachable, the supervisor disallows the transitions $m_5$ and $c_3$ from the states $(0,4)$ and $(2,3)$ respectively; these transitions lead the controlled plant to the *deadlocked* state $(0,3)$ from where the start state $(2,4)$ is unreachable.

Fig. 10(a) presents the deterministic plant obtained by renaming the cat-move from 0 to 3 by $c_4$. The supervisor corresponding to the specification $\nu X.(p \wedge [-]X)$ is presented in Fig. 10(b) . In this case, the supervisor can distinguish between the two states, one where the cat and the mouse are in rooms 1 and 4 respectively and the other where the cat and the mouse are in rooms 3 and 4 respectively. In contrast to the nondeterministic case where these states were reached on the single cat-move $c_1$, the determinization results in the reachability of the states via two distinct cat-moves $c_1$ and $c_4$.

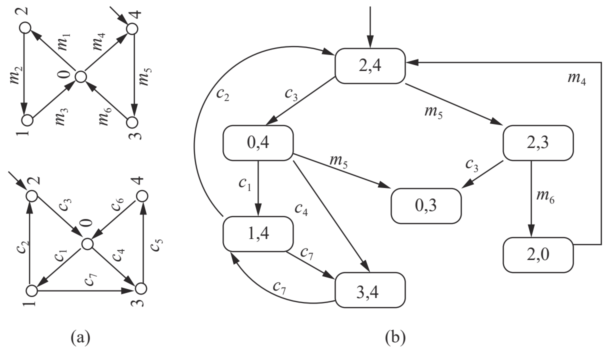

Fig. 10. (a) Deterministic cat and mouse models and (b) Supervisor for specification $\nu X.(p \wedge [-]X)$.

Table II summarizes the effect of applying the simplification rules (see Section VII) in both quotienting and model discovery modules of our implementation for both the nondeterministic and deterministic models. For example, in the absence of simplification, for the control specification $\nu X.(p \wedge [-]X)$, quotienting generates a formula expression consisting of 341 fixed point sub-formulas, while the simplification reduces it to one consisting of only 10. Tableau-based model discovery with no simplification identifies a model containing 26 transitions for the simplified quotiented formula (possessing 10 fixed point formula expressions). Bisimulation equivalence reduces the number of transitions to 9. The entries "–" represent the case where the execution is terminated after quotienting generated more than 3000 sub-formulas.

## IX. Conclusion

We presented a technique for supervisory control of nondeterministic discrete event plants under complete observation of events subject to specification expressed in the propositional $\mu$-calculus. Central to our method is a direct-quotienting of the $\mu$-calculus specification against the plant model. A control-compatible supervisor exists if and only if the quotiented formula is satisfiable, and further a model witnessing the satisfiability can be used as a supervisor. We also developed a sound and complete tableau-based methodology for satisfiability checking and model discovery of $\mu$-calculus formulas. Our technique works for nondeterministic plant models and can generate supervisors that are also nondeterministic. The complexity of verification and synthesis is single exponential in the size of the plant as well as the specification. A prior work on control for bisimilarity [40] has a complexity that is doubly exponential in the size of the plant and the specification.

Some of the future avenues for research include incorporating the notion of partial observability of actions into quotienting.

## Appendix A
## Proof for Theorem 2

*a) Equational $\mu$-calculus:* Equational system of $\mu$-calculus consists of a set of equations of the form $X =_\sigma \phi$ where $X$ belongs to the set of the fixed point variables and $\phi$ belongs to the set of basic formulas defined by the following syntax:

$$\phi \to \mathtt{tt} \mid \mathtt{ff} \mid p \mid X \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a \rangle \phi \mid [a]\phi.$$

We will use $X_i =_{\sigma_i} \varphi_i$ to denote the $i$-th equation in the equational $\mu$-calculus formula and $nd(X_i =_{\sigma_i} \varphi_i)$ to denote the nesting depth of the formula.

*b) Translation:* Given a $\mu$-calculus formula $\varphi$, its corresponding equational form is obtained by applying a translation function $\mathtt{Tr}$ as shown in Fig. 11 . For example, $\mathtt{Tr}(\ \nu X.(p \wedge [-]X \wedge \mu Y.(q \vee \langle - \rangle Y))\ ) = \left\{X =_\nu p \wedge [-]X \wedge Y,\ Y =_\mu q \vee \langle - \rangle Y\right\}$ and $nd(X =_\nu p \wedge [-]X \wedge Y) = 2$, $nd(Y =_\mu q \vee \langle - \rangle Y) = 1$.

*c) Semantics:* The semantics of $i$-th $\mu$-calculus formula equation $X_i =_{\sigma_i} \varphi_i$ is defined using the (greatest/least) fixed point of the function $F_{X_i,e} : 2^S \to 2^S$ where $S$ is the set of states in the LTS model.

$$F_{X_i,e}(S_i) = [\![\varphi_i]\!]_{e[X_i \mapsto S_i][X_j \mapsto F^N_{X_j,e[X_i \mapsto S_i]}(S^0_j)]}\ \forall X_j :$$
$$nd(X_j =_{\sigma_j} \varphi_j) = nd(X_i =_{\sigma_i} \varphi_i) - 1 \quad (3)$$

where $e : \mathcal{X} \to 2^S$, $N = |S|$ and $F^N_{X_j,e[X_i \mapsto S_i]}(S^0_j)$ is the fixed point of the function $F_{X_j,e[X_i \mapsto S_i]}$ and for all $1 \le k \le n$,

$$S^0_k = \begin{cases} \emptyset & \text{if } X_k =_{\sigma_k} \varphi_k \wedge \sigma_k = \mu \\ S & \text{otherwise.} \end{cases} \quad (4)$$

*d) From equational form to non-equational $\mu$-calculus:* Re-translating the equational formula back to its *normal* form starts from the formula equation with the highest nesting depth (i.e., the outermost fixed point formula). Every equation of the form $X =_\sigma \varphi$ is re-translated to $\sigma X.\varphi'$ where $\varphi'$ is the result of re-translating $\varphi$ such that for every $Y \in Sub(\varphi)$ if $nd(Y =_{\sigma_y} \psi) < nd(X =_\sigma \varphi)$ then $Y$ is replaced by the result of re-translating $Y =_{\sigma_y} \psi$; otherwise $Y$ remains unaltered.

$$\mathtt{RTr}(X =_\sigma \varphi) := \sigma X.\mathtt{RFTr}(\varphi, nd_X) \quad \text{where } nd_X = nd(X =_\sigma \varphi)$$
$$\mathtt{RFTr}(\mathtt{tt}, nd) := \mathtt{tt}$$
$$\mathtt{RFTr}(\mathtt{ff}, nd) := \mathtt{ff}$$
$$\mathtt{RFTr}(p, nd) := p$$
$$\mathtt{RFTr}(\varphi_1 \wedge \varphi_2, nd) := \mathtt{RFTr}(\varphi_1, nd) \wedge \mathtt{RFTr}(\varphi_2, nd)$$

TABLE II
RESULTS FOR CAT-MOUSE EXAMPLE

| **Specification** | **Plant** | **# Fixed point sub-formula** | | **# Transitions in model** | |
|---|---|---|---|---|---|
| | | *No simplification* | *Simplification* | *No reduction* | *With reduction* |
| $\nu X.(p\wedge[-]X)$ | Nondeterministic | 341 | 10 | 26 | 9 |
| | Deterministic | 338 | 10 | 14 | 11 |
| $\nu Y.(\mu Z.(q\vee\langle-\rangle Z)\wedge p\wedge[-]Y)$ | Nondeterministic | – | 846 | 24 | 7 |
| | Deterministic | – | 846 | 19 | 9 |

$$
\begin{array}{rcl}
\mathtt{Tr}(\sigma X.\varphi) & := & [X =_\sigma \mathtt{FTr}(\varphi)] \\
 & & \text{where } nd(X =_\sigma \varphi) := nd(\sigma X.\varphi) \\
\mathtt{Tr}(\varphi) & := & [X =_\mu \varphi] \\
 & & \text{if } \varphi \text{ is not a fixed point formula,} \\
 & & X \text{ is not in } \varphi \text{ and} \\
 & & nd(X =_\mu \varphi) := nd(\varphi) + 1 \\
\mathtt{FTr}(\mathtt{tt}) & := & \mathtt{tt} \\
\mathtt{FTr}(\mathtt{ff}) & := & \mathtt{ff} \\
\mathtt{FTr}(p) & := & p \\
\mathtt{FTr}(X) & := & X \\
\mathtt{FTr}(\varphi_1\wedge\varphi_2) & := & \mathtt{FTr}(\varphi_1)\wedge\mathtt{FTr}(\varphi_2) \\
\mathtt{FTr}(\varphi_1\vee\varphi_2) & := & \mathtt{FTr}(\varphi_1)\vee\mathtt{FTr}(\varphi_2) \\
\mathtt{FTr}(\langle a\rangle\varphi) & := & \langle a\rangle\mathtt{FTr}(\varphi) \\
\mathtt{FTr}([a]\varphi) & := & [a]\mathtt{FTr}(\varphi) \\
\mathtt{FTr}(\sigma X.\varphi) & := & \mathtt{Tr}(\sigma X.\varphi)
\end{array}
\tag{2}
$$

Fig. 11. Translating $\mu$-calculus to its equational form.

$$\mathtt{RFTr}(\varphi_1\vee\varphi_2,nd) := \mathtt{RFTr}(\varphi_1,nd)\vee\mathtt{RFTr}(\varphi_2,nd)$$

$$\mathtt{RFTr}(\langle a\rangle\varphi,nd) := \langle a\rangle\mathtt{RFTr}(\varphi,nd)$$

$$\mathtt{RFTr}([a]\varphi,nd) := [a]\mathtt{RFTr}(\varphi,nd)$$

$$\mathtt{RFTr}(Y,nd) := \begin{cases} \mathtt{RTr}(Y =_{\sigma_y} \psi) \\ \qquad \text{if } nd(Y =_{\sigma_y} \psi) < nd \\ Y \qquad \text{otherwise.} \end{cases} \tag{5}$$

For example, $\mathtt{RTr}(X =_\nu p\wedge[-]X\wedge Y)$ of the equations $\{X =_\nu p\wedge[-]X\wedge Y,\ Y =_\mu q\vee\langle-\rangle Y\}$ translates the formula equations to $\nu X.(p\wedge[-]X\wedge\mu Y.(q\vee\langle-\rangle Y))$.

*Lemma 1:* For any formula $\sigma X.\varphi$, $f^N_{X,\varphi,e}(S^0_1) = F^N_{X,e}(S^0_1)$ where $\mathtt{Tr}(\sigma X.\varphi) := X =_\sigma \mathtt{FTr}(\varphi)$.

*Proof:* The lemma follows directly from the definition of the functions $f$ and $F$. ■

Going back to the supervisory control problem, we will use $SQ$ to denote subsets of $S_{PC}$ and $Q$ to denote subsets of $S_C$. As before, individual states in $S_{PC}$ will be denoted by $(s,q)$ where $s\in S_P$ and $q\in S_C$. From Lemma 1, the supervised plant $P\|C$ satisfies a formula $\varphi$ if and only if its start states belong to semantics of the top variable $X_1$ of the corresponding equational formula set $E$ of $\varphi$

$$\forall (s_{0,P},s_{0,C})\in S_{0,P}\times S_{0,C} : (s_{0,P},s_{0,C})\in F^N_{X_1,e}(SQ^0_1) \tag{6}$$

where $N = |S_{PC}|$ and $SQ^0_1$ are assigned according to (4).

We introduce a function $G_{X_i,e',k} : (2^{S_C})^{|S_P|} \to (2^{S_C})^{|S_P|}$ where

$$\forall 1\le i\le n = |E|, \forall 1\le j\le m = |S_P|, \forall 1\le k\le m = |S_P| :$$
$$e'(X_i/_k s_j) = \{q \mid (s_j,q)\in e(X_i)\}. \tag{7}$$

The function is defined as follows:

$$
G_{X_i,e',k}(Q_{i1},Q_{i2},\ldots Q_{im}) =
$$
$$
\begin{cases}
[\![\varphi_i/s_1]\!]_{e'} & [(X_i/_k s_1)\mapsto Q_{i1}]\ \ldots\ [(X_i/_k s_m)\mapsto Q_{im}] \\
 & [(X_{i+1}/_1 s_1)\mapsto \\
 & G^N_{X_{i+1},e'',1}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\ldots,Q^0_{i+1\ m})(X_{i+1}/_1 s_1)] \\
 & \ldots \\
 & [(X_{i+1}/_1 s_m)\mapsto \\
 & G^N_{X_{i+1},e'',1}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\ldots,Q^0_{i+1\ m})(X_{i+1}/_1 s_m)], \\
\ldots & \\
[\![\varphi_i/s_m]\!]_{e'} & [(X_i/_k s_1)\mapsto Q_{i1}]\ \ldots\ [(X_i/_k s_m)\mapsto Q_{im}] \\
 & [(X_{i+1}/_1 s_1)\mapsto \\
 & G^N_{X_{i+1},e'',1}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\ldots,Q^0_{i+1\ m})(X_{i+1}/_1 s_1)] \\
 & \ldots \\
 & [(X_{i+1}/_1 s_m)\mapsto \\
 & G^N_{X_{i+1},e'',1}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\ldots,Q^0_{i+1\ m})(X_{i+1}/_1 s_m)].
\end{cases}
\tag{8}
$$

In the above, we use $X_{i+1}$ to denote the fixed point variables with $nd(X_{i+1} =_{\sigma_{i+1}} \varphi_{i+1}) := nd(X_i =_{\sigma_i} \varphi) - 1$, and

$$e'' = e'[(X_i/_k s_1)\mapsto Q_{i1}][(X_i/_k s_2)\mapsto Q_{i2}]\ \ldots$$
$$[(X_i/_k s_m)\mapsto Q_{im}].$$

Recursive computation of $G$ is defined as follows where $1\le k,k_1\le m = |S_P|$:

$$
G^k_{X_i,e',k_1}(Q_{i1},Q_{i2},\ldots Q_{im}) =
$$
$$
\begin{cases}
[\![\varphi_i/s_1]\!]_{e'} & [(X_i/_{k_1} s_1)\mapsto G^{k-1}_{X_i,e',k_1}(Q_{i1},Q_{i2},\ldots Q_{im})(X_i/_{k_1} s_1)] \\
 & \ldots \\
 & [(X_i/_{k_1} s_m)\mapsto G^{k-1}_{X_i,e',k_1}(Q_{i1},Q_{i2},\ldots Q_{im})(X_i/_{k_1} s_m)] \\
 & [(X_{i+1}/_k s_1)\mapsto \\
 & G^N_{X_{i+1},e'',k}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\ldots,Q^0_{i+1\ m})(X_{i+1}/_k s_1)] \\
 & \ldots \\
 & [(X_{i+1}/_k s_m)\mapsto \\
 & G^N_{X_{i+1},e'',k}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\ldots,Q^0_{i+1\ m})(X_{i+1}/_k s_m)], \\
\ldots & \\
[\![\varphi_i/s_m]\!]_{e'} & [(X_i/_{k_1} s_1)\mapsto G^{k-1}_{X_i,e',k_1}(Q_{i1},Q_{i2},\ldots Q_{im})(X_i/_{k_1} s_1)] \\
 & \ldots \\
 & [(X_i/_{k_1} s_m)\mapsto G^{k-1}_{X_i,e',k_1}(Q_{i1},Q_{i2},\ldots Q_{im})(X_i/_{k_1} s_m)] \\
 & [(X_{i+1}/_k s_1)\mapsto \\
 & G^N_{X_{i+1},e'',k}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\ldots,Q^0_{i+1\ m})(X_{i+1}/_k s_1)] \\
 & \ldots \\
 & [(X_{i+1}/_k s_m)\mapsto \\
 & G^N_{X_{i+1},e'',k}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\ldots,Q^0_{i+1\ m})(X_{i+1}/_k s_m)]
\end{cases}
\tag{9}
$$

where

$$\begin{aligned} e'' = \; & e'[(X_i/_{k_1} s_1) \mapsto G^{k-1}_{X_i,e',k_1}(Q_{i1},\dots Q_{im})(X_i/_{k_1} s_1)] \\ & [(X_i/_{k_1} s_2) \mapsto G^{k-1}_{X_i,e',k_1}(Q_{i1},\dots Q_{im})(X_i/_{k_1} s_2)] \\ & \dots \\ & [(X_i/_{k_1} s_m) \mapsto G^{k-1}_{X_i,e',k_1}(Q_{i1},\dots Q_{im})(X_i/_{k_1} s_m)]. \end{aligned}$$

We will write

$$G^k_{X_i,e',k_1}(Q_{i1},Q_{i2},\dots Q_{im})(X_i/_{k_1} s_j) =$$

$$\begin{cases} [[\varphi_i/s_1]]_{e'} & [(X_i/_{k_1} s_j) \mapsto \\ & G^{k-1}_{X_i,e',k_1}(Q_{i1},Q_{i2},\dots Q_{im})(X_i/_{k_1} s_j)] \\ & \dots \\ & [(X_i/_{k_1} s_m) \mapsto \\ & G^{k-1}_{X_i,e',k_1}(Q_{i1},Q_{i2},\dots Q_{im})(X_i/_{k_1} s_m)] \\ & [(X_{i+1}/_k s_1) \mapsto \\ & G^N_{X_{i+1},e'',k}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\dots,Q^0_{i+1\ m})(X_{i+1}/_k s_1)] \\ & \dots \\ & [(X_{i+1}/_k s_m) \mapsto \\ & G^N_{X_{i+1},e'',k}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\dots,Q^0_{i+1\ m})(X_{i+1}/_k s_m)]. \end{cases}$$

*e) Formula with 1-nesting depth:* First consider the case where there is only one equation of the form $X_n =_{\sigma_n} \varphi_n$ and all variables $X_i \neq X_n$ that are present in $\varphi_n$ are free variables whose mapping to states are decided by the environment $e$.

*Lemma 2:* $(s_j,q) \in F_{X_n,e}(S\,Q_n) \Leftrightarrow q \in G_{X_n,e',1}(Q_{n1},Q_{n2},\dots,Q_{nm})(X_n/_1 s_j)$ where $e'$ is defined as in (7) and $\forall 1 \le l \le m : Q_{nl} = \{q \mid (s_l,q) \in S\,Q_n\}$.

*Proof:* From (3), $(s_j,q) \in F_{X_n,e}(S\,Q_n) \Leftrightarrow (s_j,q) \in [[\varphi_n]]_{e[X_n \mapsto S\,Q_n]}$. It is given that $\forall 1 \le l \le m : Q_{nl} = \{q \mid (s_l,q) \in S\,Q_n\}$. We apply the quotienting rules for non-fixed point formulas (see Rules 1–7 in Fig. 4), and compute the semantics of quotienting $X_n$ against $s$ using the relationship between environment mappings $e$ and $e'$. I.e., $e'(X_n/_1 s_j) = \{q \mid (s_j,q) \in e(X_n)\}$. Therefore,

$$\begin{aligned} & (s_j,q) \in F_{X_n,e}(S\,Q_n) \\ \Leftrightarrow\; & (s_j,q) \in [[\varphi_n]]_{e[X_n \mapsto S\,Q_n]} \\ \Leftrightarrow\; & q \in [[\varphi_n/s_j]]_{e'} \left[(X_n/_1 s_1) \mapsto Q_{n1}\right] \\ & \qquad \left[(X_n/_1 s_1) \mapsto Q_{n2}\right] \\ & \qquad \dots \\ & \qquad \left[(X_n/_1 s_m) \mapsto Q_{nm}\right] \\ \Leftrightarrow\; & q \in G_{X_n,e',k}(Q_{n1},Q_{n_2},\dots,Q_{nm})(X_n/_1 s_j). \end{aligned}$$

In the above, result of quotienting $X_n$ against $s_l$ $(1 \le l \le m)$ is denoted as $(X_n/_1 s_l)$. ■

*Lemma 3:* $(s_j,q) \in F^N_{X_n,e}(S\,Q^0_n) \Leftrightarrow q \in G^N_{X_n,e',1}(Q^0_{n1},Q^0_{n2},\dots,Q^0_{nm})(X_n/_1 s_j)$ where $e'$ is defined as in (7) and $\forall 1 \le l \le m : Q^0_{nl} = \{q \mid (s_l,q) \in S\,Q^0_n\}$.

*Proof:* This can be proved by induction using Lemma 2 as the base case.

$$\begin{aligned} & (s_j,q) \in F^2_{X_n,e}(S\,Q_n) \;\Leftrightarrow \\ & (s_j,q) \in [[\varphi_n]]_{e[X_n \mapsto F_{X_n,e}(S\,Q_n)]} \;\Leftrightarrow \\ & q \in [[\varphi_n/s_j]]_{e'}[(X_n/_1 s_1) \mapsto G_{X_n,e',1}(Q_{n1},Q_{n2},\dots,Q_{nm})(X_n/s_1)] \\ & \qquad \dots[(X_n/_1 s_m) \mapsto G_{X_n,e',1}(Q_{n1},Q_{n2},\dots,Q_{nm})(X_n/s_m)] \\ & \Leftrightarrow G^2_{X_n,e',1}(Q_{n1},Q_{n2},\dots,Q_{nm})(X_n/s_j). \end{aligned}$$

■

Observe that, the functions $F$ and $G$ reach their respective fixed point in $N$ recursive computations where $N = |S_{PC}|$.

We identify the $\mu$-calculus formula equations, semantics of which is given by $G^N_{X_n,e',1}(Q^0_{n1},Q^0_{n2},\dots,Q^0_{nm})(X_n/_1 s_j)$. The equation set $E_n = \{(X_n/_1 s_l) =_{\sigma_n} \varphi_n/s_l\}$ such that the top variable $(X_n/_1 s_j)$ is defined by the equation $(X_n/_1 s_j) =_{\sigma_n} \varphi_n/s_j$ and for every $(X_n/_1 s_k)$ appearing in the right hand side of any equation, there exists a formula equation defining $(X_n/_1 s_k)$. We apply $\mathtt{Tr}((X_n/_1 s_j) =_{\sigma_n} \varphi_n/s_j)$ and rename all variables of the form $(X_n/_1 s_l)$ by $X_{n,(s_l,1)}$ to obtain the corresponding non-equational formula $\sigma_n X_{n,(s_j,1)}.\psi$ ($\psi$ obtained from $\mathtt{FTr}(\varphi_n/s_j)$ and appropriate renaming). The result is identical to the formula obtained by applying our quotienting rules (Fig. 4) on $((\sigma_n X_n.\varphi_n)/_{\emptyset} s_j)$. Our quotienting will generate a formula $\sigma_n.X_{n,(s_j,1)}.(\varphi_n/_{\{X_{n,(s_j,1)}\}} s_j)$ (Rule 8 case 2). For every, $X_n$ quotiented against $s_k$ ( $k \neq j$) in $(\varphi_n/_{\{X_{n,(s_j,1)}\}} s_j)$, Rule 9 case 3 of our quotienting will be applied which corresponds to expansion of the formula equation $(X_n/_1 s_k) =_{\sigma_n} \varphi_n/s_k$ during the re-translation from equational to non-equational form). For every, $X_n$ quotiented against $s_j$ in $(\varphi_n/_{\{X_{n,(s_j,1)}\}} s_j)$, the result is $X_{n,(s_j,1)}$ (Rule 9 case 2 of our quotienting). Finally, for every, $X_i$ quotiented against any $s_l$ $(i \neq n)$, the result is $X_{i,(s_l,1)}$ (Rule 9 case 1). Therefore, for $\mu$-calculus formula with 1-nesting, $(s,q) \in [[\sigma_n X_n.\varphi_n]]_e \Leftrightarrow q \in [[((\sigma_n X_n.\varphi_n)/_{\emptyset} s)]]_{e'}$.

*f) Formula with 2-nesting depth:* Consider that there are two formula equations $X_{n-1} =_{\sigma_{n-1}} \varphi_{n-1}$ and $X_n =_{\sigma_n} \varphi_n$ and $nd(X_n) = nd(X_{n-1}) - 1$. As before, consider all other variables are free variables. From (3), we have

$$\begin{aligned} & F_{X_{n-1},e}(S\,Q_{n-1}) \\ & \quad = [[\varphi_{n-1}]]_{e[X_{n-1} \mapsto S\,Q_{n-1}][X_n \mapsto F^N_{X_n,e[X_{n-1} \mapsto S\,Q_{n-1}]}(S\,Q^0_n)]}. \end{aligned} \tag{10}$$

From (9),

$$G^k_{X_{n-1},e',1}(Q_{n-1\ 1},Q_{n-1\ 2},\dots Q_{n-1\ m}) =$$

$$\begin{cases} [[\varphi_{n-1}/s_1]]_{e'} & [(X_{n-1}/_1 s_1) \mapsto G^{k-1}_{X_{n-1},e',1}(Q_{i1},Q_{i2},\dots \\ & Q_{im})(X_{n-1}/_1 s_1)] \\ & \dots \\ & [(X_{n-1}/_1 s_m) \mapsto G^{k-1}_{X_{n-1},e',1}(Q_{i1},Q_{i2},\dots \\ & Q_{im})(X_{n-1}/_1 s_m)] \\ & [(X_n/_k s_1) \mapsto G^N_{X_n,e'',k}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\dots \\ & Q^0_{i+1\ m})(X_n/_k s_1)] \\ & \dots \\ & [(X_n/_k s_m) \mapsto G^N_{X_n,e'',k}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\dots \\ & Q^0_{i+1\ m})(X_n/_k s_m)], \\ \dots & \\ [[\varphi_{n-1}/s_m]]_{e'} & [(X_{n-1}/_1 s_1) \mapsto G^{k-1}_{X_{n-1},e',1}(Q_{i1},Q_{i2},\dots \\ & Q_{im})(X_{n-1}/_1 s_1)] \\ & \dots \\ & [(X_{n-1}/_1 s_m) \mapsto G^{k-1}_{X_{n-1},e',1}(Q_{i1},Q_{i2},\dots \\ & Q_{im})(X_{n-1}/_1 s_m)] \\ & [(X_n/_k s_1) \mapsto G^N_{X_n,e'',k}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\dots \\ & Q^0_{i+1\ m})(X_n/_k s_1)] \\ & \dots \\ & [(X_n/_k s_m) \mapsto G^N_{X_n,e'',k}(Q^0_{i+1\ 1},Q^0_{i+1\ 2},\dots \\ & Q^0_{i+1\ m})(X_n/_k s_m)], \end{cases} \tag{11}$$

where

$$\begin{aligned}e'' = e'&[(X_{n-1}/_1 s_1) \mapsto G^{k-1}_{X_{n-1},e',1}(Q_{n-1\ 1},\ldots\\&\qquad Q_{n-1\ m})(X_{n-1}/_1 s_1)]\\&[(X_{n-1}/_1 s_2) \mapsto G^{k-1}_{X_{n-1},e',1}(Q_{n-1\ 1},\ldots,\\&\qquad Q_{n-1\ m})(X_{n-1}/_1 s_2)\\&\ldots\\&[(X_{n-1}/_1 s_m) \mapsto G^{k-1}_{X_{n-1},e',1}(Q_{n-1\ 1},\ldots,\\&\qquad Q_{n-1\ m})(X_{n-1}/_1 s_m)].\end{aligned}$$

*Lemma 4:* $(s_j,q)\in F_{X_{n-1},e}(SQ_{n-1}) \Leftrightarrow q\in G_{X_{n-1},e',1}(Q_{n-1\ 1},\ldots,Q_{n-1\ m})(X_{n-1}/_1 s_j)$ where $e'$ is defined as in (7) and $\forall 1\le l\le m: Q_{n-1\ l}=\{q\mid (s_l,q)\in SQ_{n-1}\}$.

*Proof:* The proof follows from Lemma 3, (10), (11), and Rules 1–7 in Fig. 4 (similar to proof for Lemma 2) as follows:

$$\begin{aligned}&(s_j,q)\in F_{X_{n-1},e}(SQ_{n-1}) \Leftrightarrow\\&(s_j,q)\in [\![\varphi_{n-1}]\!]_{e[X_{n-1}\mapsto SQ_{n-1}][X_n\mapsto F^N_{X_n,e[X_{n-1}\mapsto SQ_{n-1}]}(SQ^0_n)]}\\&\quad\Leftrightarrow q\in [\![\varphi_n/s_j]\!]e'[X_{n-1}/s_1\mapsto Q_{n-1\ 1}]\\&\qquad\qquad [X_{n-1}/s_2\mapsto Q_{n-1\ 2}]\ldots\\&\qquad\qquad [X_{n-1}/s_m\mapsto Q_{n-1\ m}]\\&\qquad\qquad [X_n/s_1\mapsto G^N_{X_n,e'',1}(Q^0_{n1},\ldots,Q^0_{nm})(X_n/s_1)]\ldots\\&\qquad\qquad [X_n/s_m\mapsto G^N_{X_n,e'',1}(Q^0_{n1},\ldots,Q^0_{nm})(X_n/s_m)]\\&\Leftrightarrow q\in G_{X_n,e',1}(Q_{n-1\ 1},Q_{n-1\ 2},\ldots,Q_{n-1\ m})\end{aligned}$$

where

$$\begin{aligned}e'' = e'&[X_{n-1}/s_1\mapsto Q_{n-1\ 1}]\\&[X_{n-1}/s_2\mapsto Q_{n-1\ 2}]\ldots\\&[X_{n-1}/s_m\mapsto Q_{n-1\ m}].\end{aligned}$$

In the above, result of quotienting $X_n$ and $X_{n-1}$ against $s_l$ will be denoted by $(X_n/_1 s_l)$ and $(X_{n-1}/_1 s_l)$ respectively. ■

From the above Lemma, $(s_j,q)\in F_{X_{n-1},e}(SQ^0_{n-1})\Leftrightarrow q\in G_{X_{n-1},e',k}(Q^0_{n-1\ 1},Q^0_{n-1\ 2},\ldots,Q^0_{n-1\ m})(X_{n-1}/_k s_j)$. Proceeding further, we will show that

$$\begin{aligned}F^2_{X_{n-1},e}(SQ^0_{n-1}) = [\![\varphi_{n-1}]\!]_e&[X_{n-1}\mapsto F_{X_{n-1},e}(SQ^0_{n-1})]\\&[X_n\mapsto F^N_{X_n,e[X_{n-1}\mapsto F_{X_{n-1},e}(SQ^0_{n-1})]}(SQ^0_n)].\end{aligned}\tag{12}$$

From Lemma 3, we have $(s_j,q)\in F^N_{X_n,e}(SQ^0_n) \Leftrightarrow q\in G^N_{X_n,e',1}(Q^0_{n1},Q^0_{n2},\ldots,Q^0_{nm})(X_n/_1 s_j)$. We can further infer that

$$\begin{aligned}&(s_j,q)\in F^N_{X_n,e_k}(SQ^0_n) \Leftrightarrow\\&\qquad q\in G^N_{X_n,e'_k,k}(Q^0_{n1},Q^0_{n2},\ldots,Q^0_{nm})(X_n/_k s_j)\end{aligned}\tag{13}$$

where $e_k=e[X_{n-1}\mapsto F^{k-1}_{X_{n-1},e}(SQ^0_{n-1})]$ and $e'_k$ and $e_k$ are related by (7).

Therefore, from the Lemma 4, (12), quotienting operations (Rules 1–7 in Fig. 4) and using the environment mapping $e'$, $(s_j,q)\in F^2_{X_{n-1},e}(SQ^0_{n-1})\Leftrightarrow$

$$\begin{aligned}&q\in [\![\varphi_{n-1}/s_j]\!]_{e'}[(X_{n-1}/_1 s_1)\mapsto\\&\quad G_{X_{n-1},e',1}(\langle Q^0_{n-1\ 1},\ldots,Q^0_{n-1\ m})(X_{n-1}/_1 s_1)]\\&\quad\ldots\\&\quad[(X_{n-1}/_1 s_m)\mapsto\\&\quad G_{X_{n-1},e',1}(\langle Q^0_{n-1\ 1},\ldots,Q^0_{n-1\ m})(X_{n-1}/_1 s_m)]\\&\quad[(X_n/_2 s_1)\mapsto\\&\quad G^N_{X_n,e'',2}(\langle Q^0_{n1},\ldots,Q^0_{n2}\rangle)(X_n/_2 s_1)]\\&\quad\ldots\\&\quad[(X_n/_2 s_m)\mapsto\\&\quad G^N_{X_n,e'',2}(\langle Q^0_{n1},\ldots,Q^0_{n2}\rangle)(X_n/_2 s_m)]\end{aligned}\tag{14}$$

where $e''$ is equivalent to

$$\begin{aligned}&e'[(X_{n-1}/_1 s_1)\mapsto G_{X_{n-1},e',1}(\langle Q^0_{n-1\ 1},\ldots,Q^0_{n-1\ m})(X_{n-1}/_1 s_1)]\\&\ldots\\&[(X_{n-1}/_1 s_m)\mapsto G_{X_{n-1},e',1}(\langle Q^0_{n-1\ 1},\ldots,Q^0_{n-1\ m})(X_{n-1}/_1 s_m)].\end{aligned}$$

Observe that in (12), the semantics of inner fixed point formula is computed twice; the first time in the computation of $F_{X_{n-1},e}(SQ^0_{n-1})$ using the environment mapping $X_{n-1}\mapsto SQ^0_{n-1}$, and the second time under the new environment mapping $X_{n-1}\mapsto F_{X_{n-1},e}(SQ^0_{n-1})$. We keep track of this by using the subscript 2 in the function $G_{X_n,e'',2}$. In the above, we denote quotienting of $X_{n-1}$ and $X_n$ against $s_l$ by $(X_{n-1}/_1 s_l)$ and $(X_{n-1}/_2 s_l)$ respectively. Therefore, using Lemma 3,

$$\begin{aligned}&(s_j,q)\in F^2_{X_{n-1},e}(SQ^0_{n-1})\Leftrightarrow\\&q\in G^2_{X_{n-1},e',1}(Q^0_{n-1\ 1},Q^0_{n-1\ 2},\ldots,Q^0_{n-1\ m})(X_{n-1}/_1 s_j).\end{aligned}\tag{15}$$

From the above, proceeding further we get the following:

*Lemma 5:* $(s_j,q)\in F^N_{X_{n-1},e}(SQ^0_{n-1}) \Leftrightarrow q\in G^N_{X_{n-1},e',1}(Q^0_{n-1\ 1},Q^0_{n-1\ 2},\ldots,Q^0_{n-1\ m})(X_{n-1}/_1 s_j)$ where $e'$ is defined as in (7) and $\forall 1\le l\le m: Q^0_{n-1,l}=\{q\mid (s_l,q)\in SQ^0_{n-1}\}$. ■

We obtain the equation set semantics of which is given by $G^N_{X_{n-1},e',1}(Q^0_{n-1\ 1},Q^0_{n-1\ 2},\ldots,Q^0_{n-1\ m})(X_{n-1}/_1 s_j)$. It will contain the equation for the top-variable $(X_{n-1}/_1 s_j) =_{\sigma_{n-1}} \varphi_{n-1}/s_j$. If $\varphi_{n-1}/s_j$ results in quotienting of $X_i$ ($i\in\{n,n-1\}$) against $s_l$, then we generate a formula equation of the form $(X_i/_{k_1} s_l) =_{\sigma_i} \varphi_i/s_l$ where the mapping of $(X_i/_{k_1} s_l)$ is $G^k_{X_i,e',k_1}(.)((X_i/_{k_1} s_l))$ in the computation of $G^N_{X_{n-1},e',1}(.)(X_{n-1}/_1 s_j)$. The quotienting of $\varphi_i$ against $s_l$ and the subsequent generation of formula equations (avoiding repetitions) is handled recursively in similar fashion starting from the function $G^k_{X_i,e',k_1}(.)((X_i/_{k_1} s_l))$.

The formulas are translated to their corresponding non-equational form using (5).

Finally, all variables are renamed as follows: starting from the outermost fixed point formula (in this case starting from $\sigma_{n-1}(X_{n-1}/_1 s_j).\psi$ where $\psi=\texttt{RFTr}(\varphi/s_j)$): $l$-th occurrence of quotienting $X_i$ ($i\in\{n,n-1\}$) against $s_l$ resulting in a formula $\sigma_i(X_l/_k s_l).\psi'$ is renamed to $\sigma_i X_{i,(s_j,l)}.\psi'[(X_i/_k s_j)/X_{i,(s_j,l)}]$ where $\psi'[(X_i/_k s_j)/X_{i,(s_j,l)}]$ denotes renaming of every occurrence of $(X_i/_k s_j)$ in $\psi'$ with $X_{i,(s_j,l)}$.

Note that, we are performing the variable renaming to show that the resultant formula is syntactically identical to the result obtained by applying our quotienting rules; the semantics after renaming remains unaltered. The renaming is explained as follows. The first time any formula $\sigma_i X_i.\varphi_i$ is quotiented against $s_l$, the result is $\sigma_i X_{i,(s_j,1)}.(\varphi_i/_{T\cup\{X_{i,(s_j,1)}\}} s_j)$ (Rule 8 case 2). According to our formulation in equational $\mu$-calculus, this will imply the "last" time the mapping of $(X_i/_k s_j)$ is used to compute the fixed point of a formula with higher nesting depth (if any). Proceeding further, every occurrence $(> 1)$ of formula formed by quotienting $\sigma_i X_i.\psi_i$ against $s_j$ (i.e., $\sigma_i(X_i/_k s_j).\psi'_i$ in the re-translated formula) implies that Rule 8 case 1 in our quotienting rule is applied. The outer fixed point formula expression will be quotiented against each state at most once (see above for the computation of function $G$). The renamed formula is identical to the result obtained via our quotienting of $\sigma_{n-1}X_{n-1}.\mathtt{RFTr}(\varphi_{n-1})$.

The above can be extended to formulas with any nesting depth. As such, $(s_j,q)\in F^N_{X_1,e}(S\,Q^0_1)\Leftrightarrow q\in G^N_{X_1,e',1}(Q^0_{1\ 1},Q^0_{1\ 2},\dots,Q^0_{1\ m})(X_1/_1 s_j)$. Therefore, $(s,q)\in[\![\sigma X.\varphi]\!]_e\Leftrightarrow q\in[\![((\sigma X.\varphi)/_\emptyset s)]\!]_{e'}$. This concludes the proof of Theorem 2. ■

## APPENDIX B
## XSB Encoding of Cat-Mouse Example

```
%% ctrans represent transition relations for
%% each component in the plant model

% cat transitions
ctrans(cat(1), c2, cat(2)).
ctrans(cat(1), c7, cat(3)).
ctrans(cat(2), c3, cat(0)).
ctrans(cat(0), c1, cat(1)).
ctrans(cat(0), c1, cat(3)).
ctrans(cat(3), c5, cat(4)).
ctrans(cat(3), c7, cat(1)).
ctrans(cat(4), c6, cat(0)).

% mouse transitions
ctrans(mouse(2), m2, mouse(1)).
ctrans(mouse(1), m3, mouse(0)).
ctrans(mouse(0), m1, mouse(2)).
ctrans(mouse(0), m4, mouse(4)).
ctrans(mouse(4), m5, mouse(3)).
ctrans(mouse(3), m6, mouse(0)).

uncontrollable(cat(1), [c7]).
uncontrollable(cat(3), [c7]).

%% transition rule for plant model
trans(S, A, T) :- pickone(S, S1),
                  ctrans(S1, A, T1),
                  putback(S, S1, T1, T).

%% start state
startstate([cat(2), mouse(4)]).

%% label
label([cat(I), mouse(J)], p) :- I\=J
label([cat(2), mouse(4)], q).
```

## References

[1] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.* , vol. 25, no. 1, pp. 206–230, Jan. 1987.

[2] R. Kumar and V. K. Garg, *Modeling and Control of Logical Discrete Event Systems*. Boston, MA: Kluwer Academic Publishers, 1995.

[3] A. Arnold, A. Vincent, and I. Walukiewicz, "Games for synthesis of controllers with partial observation," *Theor. Comput. Sci.* , vol. 303, no. 1, pp. 7–34, Jun. 2003.

[4] A. Arnold, X. Briand, G. Point, and A. Vincent, "A generic approach to the control of discrete event systems," in *Proc. 44th IEEE Conf. Decision and Control*, Seville, Spain, 2005, pp. 1–5.

[5] X. Briand, "Dynamic control with indistinguishable events," *Discrete Event Dyn. Syst.*, vol. 16, no. 3, pp. 353–384, Sept. 2006.

[6] S. Basu and R. Kumar, " Quotient-based control synthesis for non-deterministic plants with $\mu$-calculus specifications," in *Proc. 45th IEEE Conf. Decision and Control*, San Diego, CA, USA, 2006, pp. 6041–6046.

[7] A. Arnold and I. Walukiewicz, " Nondeterministic controllers of nondeterministic processes," *Logic and Automata*, vol. 2, pp. 29–52, 2008.

[8] S. Basu and R. Kumar, "Quotient-based control synthesis for partially observed non-deterministic plants with $\mu$ -calculus specifications," in *Proc. 46th IEEE Conf. Decision and Control*, New Orleans, LA, USA, 2007, 5294–5299.

[9] S. Riedweg and S. Pinchinat, "Quantified $\mu$ -calculus for control synthesis," in *Mathematical Foundations of Computer Science*. Berlin, Heidelberg: Springer, 2003, pp. 642–651.

[10] S. Pinchinat and S. Riedweg, "A decidable class of problems for control under partial observation," *Inform. Process. Lett.* , vol. 95, no. 4, pp. 454–460, Aug. 2005.

[11] H. R. Andersen, " Partial model checking," in *Logic in Computer Science*, 1995.

[12] S. Basu and C. R. Ramakrishnan, " Compositional analysis for verification of parameterized systems," *Theor. Comput. Sci.*, vol. 354, no. 2, pp. 211–229, Mar. 2006.

[13] E. A. Emerson and C. S. Jutla, "The complexity of tree automata and logics of programs," *SIAM J. Comput.*, vol. 29, no. 1, pp. 132–158, Sept. 1999.

[14] C. Stirling, "Games and modal mu-calculus," in *Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Heidelberg: Springer, 1996, pp. 298–312.

[15] J. Zappe, "Modal $\mu$-calculus and alternating tree automata," in *Automata Logics, and Infinite Games*. Berlin, Heidelberg: Springer, 2002, pp. 171–184.

[16] M. A. Shayman and R. Kumar, " Supervisory control of nondeterministic systems with driven events via prioritized synchronization and trajectory models," *SIAM J. Control Optim.*, vol. 33, no. 2, pp. 469–497, Mar. 1995.

[17] R. Kumar and M. A. Shayman, "Nonblocking supervisory control of nondeterministic systems via prioritized synchronization," *IEEE Trans. Automat. Control*, vol. 41, no. 8, pp. 1160–1175, Aug. 1996.

[18] R. Kumar and M. A. Shayman, " Centralized and decentralized supervisory control of nondeterministic systems under partial observation," *SIAM J. Control Optim.*, vol. 35, no. 2, pp. 363–383, Mar. 1997.

[19] A. Overkamp, "Supervisory control for nondeterministic systems," in *11th Int. Conf. Analysis and Optimization of Systems Discrete Event Systems. Lecture Notes in Control and Information Sciences*. Berlin, Heidelberg: Springer-Verlag, 1994, pp. 59–65.

[20] M. Heymann and F. Lin, " Discrete-event control of nondeterministic systems," *IEEE Trans. Automat. Control* , vol. 43, no. 1, pp. 3–17, Jan. 1998.

[21] R. Kumar and M. Heymann, "Masked prioritized synchronization for interaction and control of discrete event systems," *IEEE Trans. Automat. Control*, vol. 45, no. 11, pp. 1970–1982, Nov. 2000.

[22] S. B. Jiang and R. Kumar, " Supervisory control of nondeterministic

discrete-event systems with driven events via masked prioritized synchronization," *IEEE Trans. Automat. Control*, vol. 47, no. 9, pp. 1438–1449, Sept. 2002.

[23] K. Inan, "Nondeterministic supervision under partial observations," in *11th Int. Conf. Analysis and Optimization of Systems Discrete Event Systems. Lecture Notes in Control and Information Sciences*. Berlin, Heidelberg: Springer-Verlag, 1994, pp. 39–48.

[24] M. A. Shayman and R. Kumar, "Process objects/masked composition: An object-oriented approach for modeling and control of discrete-event systems," *IEEE Trans. Automat. Control*, vol. 44, no. 10, pp. 1864–1869, Oct. 1999.

[25] R. Kumar, S. B. Jiang, C. Y. Zhou, and W. B. Qiu, "Polynomial synthesis of supervisor for partially observed discrete-event systems by allowing nondeterminism in control," *IEEE Trans. Automat. Control*, vol. 50, no. 4, pp. 463–475, Apr. 2005.

[26] S. B. Jiang and R. Kumar, " Supervisory control of discrete event systems with CTL* temporal logic specifications," *SIAM J. Control Optim.*, vol. 44, no. 6, pp. 2079–2103, Jan. 2006.

[27] M. Antoniotti, " Synthesis and verification of discrete controllers for robotics and manufacturing devices with temporal logic and the control-D system," Ph.D. dissertation, Department of Computer Science, New York University, New York, NY, 1995.

[28] O. Kupferman, M. Y. Vardi, and P. Wolper, " Module checking," *Inform. Comput.*, vol. 164, no. 2, pp. 322–344, Jan. 2001.

[29] O. Kupferman, P. Madhusudan, P. S. Thiagarajan, and M. Y. Vardi, "Open systems and reactive environments: Control and synthesis," in *CONCUR 2000 — Concurrency Theory. Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer-Verlag, 2000, pp. 92–107.

[30] O. Kupferman and M. Y. Vardi, "Robust satisfaction," in *CONCUR'99 Concurrency Theory. Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer-Verlag, 1999, pp. 382–398.

[31] S. Riedweg and S. Pinchinat, " Quantified mu-calculus for control synthesis," in *Mathematical Foundations of Computer Science*. Berlin, Heidelberg: Springer, 2003, pp. 642–651.

[32] J. J. M. M. Rutten, "Coalgebra, concurrency, and control," in *Discrete Event Systems*. Boston, MA: Springer, 2000, pp. 31–38.

[33] J. Komenda, " Computation of supremal sublanguages of supervisory control using coalgebra," in *Proc. 6th Int. Workshop on Discrete Event Systems*, Zaragoza, Spain, 2002, pp. 26–33.

[34] G. Barrett and S. Lafortune, "Using bisimulation to solve discrete event control problems," in *Proc. American Control Conf.*, Albuquerque, New Mexico, USA, 1997, pp. 2337–2341.

[35] H. Marchand and S. Pinchinat, " Supervisory control problem using symbolic bisimulation techniques," in *Proc. American Control Conf.*, Chicago, Illinois, USA, 2000, pp. 4067–4071.

[36] J. Komenda, " Coalgebra and supervisory control of discrete-event systems with partial observations," in *Int. Symposium*, Notre Dame, 2002, pp. 12–16.

[37] H. J. Qin and P. Lewis, " Factorization of finite state machines under observational equivalence," in *Proc. Int. Conf. Concurrency Theory. Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer-Verlag, 1990, pp. 427–441.

[38] P. Madhusudan and P. S. Thiagarajan, "Branching time controllers for discrete event systems," *Theor. Comput. Sci.*, vol. 274, no. 1–2, pp. 117–149, Mar. 2002.

[39] P. Tabuada, "Open maps, alternating simulations and control synthesis," in *Proc. Int. Conf. Concurrency Theory*. Berlin, Heidelberg: Springer, 2004, pp. 466–480.

[40] C. Y. Zhou, R. Kumar, and S. B. Jiang, " Control of nondeterministic discrete-event systems for bisimulation equivalence," *IEEE Trans. Automat. Control*, vol. 51, no. 5, pp. 754–765, May 2006.

[41] C. Y. Zhou and R. Kumar, "Bisimilarity enforcement for discrete event systems using deterministic control," *IEEE Trans. Automat. Control*, vol. 56, no. 12, pp. 2986–2991, Dec. 2011.

[42] R. Kumar, S. B. Jiang, and C. Y. Zhou, " Comment on "bisimilarity control of partially observed nondeterministic discrete event systems and a test algorithm" [automatica 47 (2011) 782–788]," *Automatica*, vol. 50, no. 1, pp. 296–297, Jan. 2014.

[43] S. Takai, " Synthesis of bisimilarity enforcing supervisors for nondeterministic discrete event systems," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 1–6, Jan. 2018.

[44] C. Y. Zhou and R. Kumar, " Bisimilarity control of partially observed deterministic systems," *IEEE Trans. Automat. Control*, vol. 52, no. 9, pp. 1642–1653, Sept. 2007.

[45] C. Y. Zhou and R. Kumar, " A small model theorem for bisimilarity control under partial observation," *IEEE Trans. Automat. Sci. Eng.*, vol. 4, no. 1, pp. 93–97, Jan. 2007.

[46] C. Y. Zhou and R. Kumar, "Control of nondeterministic discrete event systems for simulation equivalence," *IEEE Trans. Automat. Sci. Eng.*, vol. 4, no. 3, pp. 340–349, Jul. 2007.

[47] S. Takai, " Synthesis of maximally permissive supervisors for nondeterministic discrete event systems with nondeterministic specifications," *IEEE Trans. Automat. Control*, 2020. DOI: 10.1109/TAC.2020.3015453

[48] T. H. Xu, H. F. Wang, T. M. Yuan, and M. C. Zhou, "BDD-based synthesis of fail-safe supervisory controllers for safety-critical discrete event systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 9, pp. 2385–2394, Sept. 2016.

[49] D. Kozen, " Results on the propositional $\mu$-calculus," *Theor. Comput. Sci.*, vol. 27, no. 3, pp. 333–354, Dec. 1983.

[50] E. A. Emerson, C. S. Jutla, and A. P. Sistla, "On model checking for the $\mu$-calculus and its fragments," *Theor. Comput. Sci.*, vol. 258, no. 1–2, pp. 491–522, May 2001.

[51] G. Bhat and R. Cleaveland, " Efficient model checking via the equational $\mu$-calculus" , in *Proc. 11th Annu. IEEE Symp. Logic in Computer Science*, New Brunswick, NJ, USA, 1996, pp. 304–312.

[52] A. Tarski, " A lattice-theoretical fixpoint theorem and its applications," *Pacific J. Math.*, vol. 5, pp. 285–309, 1955.

[53] F. Cassez and F. Laroussinie, " Model-checking for hybrid systems by quotienting and constraints solving", in *Proc. 12th Int. Conf. Computer Aided Verification*, Berlin, Heidelberg, 2000, pp. 373–388.

[54] The XSB Group. The XSB logic programming system, 2004. [Online]. Available: http://xsb.sourceforge.net

[55] R. Milner, *A Calculus of Communicating Systems*. Berlin Heidelberg: Springer Verlag, 1980.

**Samik Basu** is Professor in the Department of Computer Science at Iowa State University, USA. His research interests include formal methods, software verification, social network analysis, preference reasoning and security. He received the B.E. in computer science and engineering from Jadavpur University, India, in 1998, and the M.S. and Ph.D. degrees in computer science from Stony Brook University in 2001 and 2003, respectively.

**Ratnesh Kumar** (F'07) is a Harpole Professor at the Department of Electrical and Computer Engineering at Iowa State University, where he directs the ESSeNCE (Embedded Software, Sensors, Networks, Cyberphysical, and Energy) Lab. Previously, he held faulty position at the University of Kentucky, and various visiting positions with the University of Maryland (College Park), the Applied Research Laboratory at the Pennsylvania State University (State College), the NASA Ames, the Idaho National Laboratory, the United Technologies Research Center, and the Air Force Research Laboratory. He received the B.Tech. degree in electrical engineering from IIT, India, in 1987, and the M.S. and Ph.D. degrees in electrical and computer engineering from The University of Texas at Austin in 1989 and 1991, respectively. He was a recipient of the Gold Medals for the Best EE Undergrad, the Best EE Project, and the Best All Rounder from IIT Kanpur, the Best Dissertation Award from UT Austin, the Best Paper Award from the IEEE Transactions on Automation Science and Engineering, and Keynote Speaker and paper awards recipient from multiple conferences. He is or has been an Editor of several journals (including of IEEE, SIAM, ACM, Springer, IET, MDPI), was a Distinguished Lecturer of the IEEE Control Systems Society, is a recipient of D. R. Boylan Eminent Faculty Award for Research from Iowa State University, a Fellow of IEEE, and also a Fellow of AAAS.