A Classification of Symbolic Transition Systems

THOMAS A. HENZINGER
University of California, Berkeley
RUPAK MAJUMDAR
University of California, Berkeley
and
JEAN-FRANÇOIS RASKIN
Université Libre de Bruxelles, Belgium

Authors' addresses: Thomas A. Henzinger, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, CA 94720-1770, USA. Email: tah@eecs.berkeley.edu. URL: http://www.eecs.berkeley.edu/~tah

Rupak Majumdar, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, CA 94720-1770, USA. Email: rupak@eecs.berkeley.edu. URL: http://www.eecs.berkeley.edu/~rupak

Jean-François Raskin, Département d'Informatique, Faculté des Sciences, Université Libre de Bruxelles, Belgium. Email: jfr@ulb.ac.be. URL: http://www.ulb.ac.be/di/ssd/jfr

A preliminary version of this paper appeared in the *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science* (STACS), Lecture Notes in Computer Science 1770, Springer-Verlag, 2000, pages 13–34.

This research was supported in part by the DARPA grant F33615-C-98-3614, the MARCO grant 98-DT-660, the NSF grants CCR-9988172, CCR-0085949, and CCR-0225610, and the Belgian National Fund for Scientific Research (FNRS).

We define five increasingly comprehensive classes of infinite-state systems, called STS1–STS5, whose state spaces have finitary structure. For four of these classes, we provide examples from hybrid systems.

STS1 These are the systems with finite *bisimilarity* quotients. They can be analyzed symbolically by iteratively applying predecessor and boolean operations on state sets, starting from a finite number of observable state sets. Any such iteration is guaranteed to terminate in that only a finite number of state sets can be generated. This enables model checking of the μ-calculus.

STS2 These are the systems with finite *similarity* quotients. They can be analyzed symbolically by iterating the predecessor and positive boolean operations. This enables model checking of the existential and universal fragments of the μ-calculus.

STS3 These are the systems with finite *trace-equivalence* quotients. They can be analyzed symbolically by iterating the predecessor operation and a restricted form of positive boolean operations (intersection is restricted to intersection with observables). This enables model checking of all ω-regular properties, including linear temporal logic.

STS4 These are the systems with finite *distance-equivalence* quotients (two states are equivalent if for every distance d, the same observables can be reached in d transitions). The systems in this class can be analyzed symbolically by iterating the predecessor operation and terminating when no new state sets are generated. This enables model checking of the existential conjunction-free and universal disjunction-free fragments of the μ-calculus.

STS5 These are the systems with finite *bounded-reachability* quotients (two states are equivalent if for every distance d, the same observables can be reached in d or fewer transitions). The systems in this class can be analyzed symbolically by iterating the predecessor operation and terminating when no new states are encountered (this is a weaker termination condition than above). This enables model checking of reachability properties.

Categories and Subject Descriptors: D.2.4 [**Software Engineering**]: Software/Program Verification; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs

General Terms: Formal Verification, Temporal Logics

Additional Key Words and Phrases: Model Checking, Symbolic Algorithms, State Equivalences, Hybrid Automata

1. INTRODUCTION

Algorithmic verification methods ("model checking") were originally invented for the analysis of finite-state systems, whose state spaces can be explored exhaustively. Much recent interest has concerned the application of such methods to *infinite-state* systems. There are two important approaches. Methods of the first kind reduce an infinite-state system to an "equivalent" finite-state system —often a quotient system— and then explore the resulting finite quotient space. We call these methods *reductionist*. A well-known representative of the reductionist approach is the region-graph method for timed automata [Alur and Dill 1994]: every timed automaton is bisimilar to a finite graph of clock regions, each representing an equivalence class of infinitely many clock states; model checking a timed automaton can therefore be reduced to model checking its region graph. Methods of the second kind explore the infinite state space directly, by manipulating a data type called *region*, whose members represent possibly infinite sets of states. We call these methods *symbolic*.[1] A typical representative of the symbolic approach is the clock-zone method for timed automata [Henzinger et al. 1994]: the symbolic model checking of timed automata manipulates clock zones, each representing a boolean combination of clock constraints, where a clock constraint bounds a clock value or the difference between two clock values (a single clock zone, therefore, may correspond to a union of many clock regions). While sometimes optimal in theoretical complexity, reductionist methods usually experience state explosion and are often outperformed in practice by symbolic methods. It is easy to see why: a symbolic method never distinguishes states that are considered equivalent by the reductionist approach, but when answering a specific verification question, a symbolic method performs an analysis of the given infinite state space that may be considerably less detailed than would be required for constructing the entire finite quotient system. In a sense, the symbolic method performs no more work than necessary for answering the specific question, whereas the reductionist method provides a worst-case upper bound on the amount of work to be performed.

Regions (state sets) in symbolic model checking might be implemented as predicates in some constraint language, for example, as constraints on the booleans, integers, or reals. A *region algebra* supports individual operations on regions. For model checking, we are particularly interested in boolean operations on regions (such as set intersection and set difference) as well as the predecessor operation *Pre*, which, given a target region, computes the region of all states with successors in the target region. The iteration of these operations may generate an infinite number of distinct regions. The main concern with symbolic methods, therefore, is *termination*. We refer to procedures that may or may not terminate as "semi-algorithms." In this paper, we study restricted classes of infinite-state transition systems for which certain forms of iteration terminate after a finite number of region operations, and yet yield sufficient information for checking unbounded temporal properties of the system. In particular, we argue that the semi-algorithms for model checking various temporal logics can be seen as instances of generic *closure semi-algorithms*, which refine a partition of the state space by applying the *Pre* operator together with a possibly limited selection of boolean region operators. Hence, if a closure semi-algorithm terminates, so do the corresponding semi-algorithms for model checking. Second, we show that the closure semi-algorithms terminate exactly when certain equivalence relations on the infinite state space have finite index. Thus, to obtain symbolic model-checking procedures for infinite-state systems, it suffices that the corresponding equivalence relations have finite index. In other words, while the algorithms themselves are completely symbolic, their guarantees of termination are given by a reductionist argument (existence of an "equivalent" finite quotient). For example, while in theory there are infinitely many clock constraints for timed automata, the clock-zone method can be shown to compute only

[1]Our usage of the term "symbolic" is considerably broader than the narrow usage commmon in some model-checking communities, where "symbolic" often means "based on binary-decision diagrams" (BDDs) [Burch et al. 1992]. Note that BDDs are but one data structure for representing state sets (in this case, finite boolean state sets). Similarly, our usage of the term "region" is more general than the clock regions of timed automata: while the latter is a set of clock states of a specific shape, we denote by "region" any representable set of states.

clock constraints that define blocks (unions) of the finitely many clock regions which represent bisimilarity classes of states. Reduction is not part of the algorithm, but part of its correctness proof.

We propose a classification of infinite-state systems into five increasingly general classes, depending on which equivalence relations have finite index. The first class of infinite-state systems are those with finite *bisimilarity* quotients. Timed automata fall into this class. On systems with finite bisimilarity quotients, we can symbolically model check all μ-calculus formulas, including the formulas of the branching temporal logics CTL and CTL* [Emerson 1990]. The second class of infinite-state systems are those with finite *similarity* quotients. On these, we can model check all existential and universal formulas of the μ-calculus (and of branching temporal logics). A formula is existential (resp. universal) if it is in positive normal form (i.e., negations occur only in front of atomic propositions) and contains only existential (resp. universal) path quantifiers. Infinite-state systems of the third class have finite *trace-equivalence* quotients. On these, all ω-regular properties, expressed by Büchi automata or linear temporal-logic formulas, can be model checked. The fourth class constitute systems with finite *distance-equivalence* quotients. Distance equivalence distinguishes two states u and v if there is an observable p and a number n, such that from u a state satisfying p can be reached in n transitions, but from v no p-state can be reached in exactly n transitions. On systems with finite distance-equivalence quotients, we can model check the existential conjunction-free and universal disjunction-free formulas of the μ-calculus. Finally, the fifth class contains systems with finite *bounded-reachability* quotients. Bounded-reach equivalence distinguishes two states u and v if there is an observable p and a number n, such that from u a state satisfying p can be reached in n transitions, but from v no p-state can be reached in n or fewer transitions. These are the well-structured transition systems of [Finkel and Schnoebelen 1998], and they permit the model checking of reachability properties.

This classification provides a syntax-independent way for proving the termination of symbolic model-checking algorithms for infinite-state systems. Examples can be found from subclasses of hybrid automata: while timed automata fall into the first class (finite bisimilarity [Alur and Dill 1994]), two-dimensional rectangular automata fall into the second class (finite similarity [Henzinger et al. 1995]), higher-dimensional rectangular automata fall into the third class (finite trace equivalence [Henzinger et al. 1998; Henzinger and Kopke 1996]), and networks of timed and rectangular automata fall into the fifth class (finite bounded reachability [Abdulla and Jonsson 1998]). While in each of these four specific cases, termination of the symbolic approach has been proved individually [Henzinger et al. 1994; Henzinger et al. 1995; Henzinger and Majumdar 2000; Abdulla and Jonsson 1998], in this paper we provide termination proofs for symbolic model-checking algorithms for the entire classes 1–5. It should be mentioned that we know of no natural representatives for the fourth class of infinite-state systems. However, the closure properties of classes 1–3 immediately give rise to a more general class, which, however, does not encompass all well-structured transition systems (class 5): bisimilarity classes are closed under *Pre*, intersection, and difference; similarity classes, under *Pre* and intersection; trace-equivalence classes, under *Pre* and intersection with observables; distance-equivalence classes, under *Pre* only. This motivates the definition of class 4.

2. SYMBOLIC TRANSITION SYSTEMS

We say that a transition system is "symbolic" if it comes equipped with an algebra of regions which permits the effective computation of certain operations on regions.

Definition: Symbolic transition system. A *symbolic transition system* $\mathcal{S} = (Q, \delta, R, \ulcorner\cdot\urcorner, P)$ consists of a (possibly infinite) set Q of *states*, a (possibly nondeterministic) *transition* function $\delta\colon Q \to \mathbf{2}^Q$ which maps each state to a set of successor states, a (possibly infinite) set R of *regions*, an *extension* function $\ulcorner\cdot\urcorner\colon R \to \mathbf{2}^Q$ which maps each region to a set of contained states, and a finite set $P \subseteq R$ of *observables*, such that the following six conditions are satisfied:

(1) The set P of observables covers the state space Q; that is, $\bigcup\{\ulcorner p\urcorner \mid p \in P\} = Q$. Moreover, for each observable $p \in P$, there is a complementary observable $\overline{p} \in P$ such that $\ulcorner\overline{p}\urcorner = Q \setminus \ulcorner p\urcorner$.

(2) For each region $\sigma \in R$, there is a region $Pre(\sigma) \in R$ such that

$$\ulcorner Pre(\sigma)\urcorner \;=\; \{u \in Q \mid (\exists v \in \delta(u)\colon v \in \sigma)\};$$

furthermore, the function $Pre\colon R \to R$ is computable.

(3) For each pair $\sigma, \tau \in R$ of regions, there is a region $And(\sigma,\tau) \in R$ such that $\ulcorner And(\sigma,\tau)\urcorner = \ulcorner\sigma\urcorner \cap \ulcorner\tau\urcorner$; furthermore, the function $And : R \times R \to R$ is computable.

(4) For each pair $\sigma, \tau \in R$ of regions, there is a region $Diff(\sigma,\tau) \in R$ such that $\ulcorner Diff(\sigma,\tau)\urcorner = \ulcorner\sigma\urcorner\setminus\ulcorner\tau\urcorner$; furthermore, the function $Diff : R \times R \to R$ is computable.

(5) All emptiness questions about regions can be decided; that is, there is a computable function $Empty\colon R \to \mathbb{B}$ such that $Empty(\sigma)$ iff $\ulcorner\sigma\urcorner = \emptyset$.

(6) All membership questions about regions can be decided; that is, there is a computable function $Member\colon Q \times R \to \mathbb{B}$ such that $Member(u,\sigma)$ iff $u \in \ulcorner\sigma\urcorner$.

The tuple $\mathcal{R}_{\mathcal{S}} = (P, Pre, And, Diff, Empty)$ is called the *region algebra* of $\mathcal{S}$. □

Remark: Abstract interpretation. In a symbolic transition system, the semantics is lifted from individual states to sets of states. Hence the region algebra of a symbolic transition system can be viewed as the *collecting semantics* (in the sense of abstract interpretation [Cousot and Cousot 1977]) of the concrete semantics of the transition system. □

Remark: Duality. We take an existential view of symbolic transition systems. The dual, universal view requires (1) $\bigcap\{\ulcorner p\urcorner \mid p \in P\} = \emptyset$, (2–4) closure of R under computable functions $\overline{Pre}$, $\overline{And}$, and $\overline{Diff}$ such that

$$\ulcorner\overline{Pre}(\sigma)\urcorner \;=\; \{u \in Q \mid (\forall v \in \delta(u)\colon v \in \sigma)\},$$

$\ulcorner\overline{And}(\sigma,\tau)\urcorner = \ulcorner\sigma\urcorner \cup \ulcorner\tau\urcorner$, and $\ulcorner\overline{Diff}(\sigma,\tau)\urcorner = Q\setminus\ulcorner Diff(\tau,\sigma)\urcorner$, and (5) a computable function $\overline{Empty}$ for deciding all universality questions about regions (that is, $\overline{Empty}(\sigma)$ iff $\ulcorner\sigma\urcorner = Q$). The tuple $\overline{\mathcal{R}}_{\mathcal{S}} = (P, \overline{Pre}, \overline{And}, \overline{Diff}, \overline{Empty})$ is the *dual* region algebra of $\mathcal{S}$. All results of this paper have an alternative, dual formulation. $\square$

2.1 Example: Polyhedral Hybrid Automata

A *polyhedral hybrid automaton H of dimension m*, for a positive integer m, consists of the following components [Alur et al. 1996]:

Continuous variables. A set $X = \{x_1, \ldots, x_m\}$ of real-valued variables. We write $\dot{X}$ for the set $\{\dot{x}_1, \ldots, \dot{x}_m\}$ of dotted variables (which represent first derivatives during continuous change), and we write X' for the set $\{x'_1, \ldots, x'_m\}$ of primed variables (which represent values at the conclusion of discrete change). A *linear constraint* over X is an expression of the form $k_0 \sim k_1 x_1 + \cdots + k_m x_m$, where $\sim \in \{<, \leq, =, \geq, >\}$ and $k_0, \ldots, k_m$ are integer constants. A *linear predicate* over X is a boolean combination of linear constraints over X. Let $\mathcal{L}^m(X)$ be the set of linear predicates over X. Given a predicate ϕ over X, and a valuation $\mathbf{x} \in \mathbb{R}^m$ for the variables in X, we write $\phi[X := \mathbf{x}]$ for the truth value that results from evaluating ϕ at $\mathbf{x}$.

Discrete locations. A finite, directed multigraph (V, E). The vertices in V are called *locations*; the edges in E are called *jumps*.

Invariant and flow conditions. Two vertex-labeling functions *inv* and *flow*. For each location $v \in V$, the invariant condition $inv(v)$ is a conjunction of linear constraints over X, and the flow condition $flow(v)$ is a conjunction of linear constraints over $\dot{X}$. While the automaton control resides in location v, the variables may evolve according to $flow(v)$ as long as $inv(v)$ remains true.

Update conditions. An edge-labeling function *update*. For each jump $e \in E$, the update condition $update(e)$ is a conjunction of linear constraints over $X \cup X'$. The predicate $update(e)$ relates the possible values of the variables at the beginning of the jump (represented by X) and at the conclusion of the jump (represented by X').

The polyhedral hybrid automaton H is a *rectangular automaton* [Henzinger et al. 1998] if

—all linear constraints that occur in invariant conditions of H have the form $x \sim k$, for $x \in X$ and $k \in \mathbb{Z}$;
—all linear constraints that occur in flow conditions of H have the form $\dot{x} \sim k$, for $x \in X$ and $k \in \mathbb{Z}$;
—all linear constraints that occur in jump conditions of H have the form $x \sim k$ or $x' = x$ or $x' \sim k$, for $x \in X$ and $k \in \mathbb{Z}$;
—if e is a jump from location v to location v', and $update(e)$ contains the conjunct $x' = x$, then both $flow(v)$ and $flow(v')$ contain the same constraints on $\dot{x}$.

The rectangular automaton H is a *singular automaton* if each flow condition of H has the form $\dot{x}_1 = k_1 \wedge \ldots \wedge \dot{x}_m = k_m$. The singular automaton H is a *timed automaton* [Alur and Dill 1994] if each flow condition of H has the form $\dot{x}_1 = 1 \wedge \ldots \wedge \dot{x}_m = 1$.

The polyhedral hybrid automaton H defines the symbolic transition system $\mathcal{S}_H = (Q_H, \delta_H, R_H, \ulcorner\cdot\urcorner_H, P_H)$ with the following components:

—$Q_H = V \times \mathbb{R}^m$; that is, every state $(v, \mathbf{x})$ consists of a location v (the discrete component of the state) and values $\mathbf{x}$ for the variables in X (the continuous component).

—$(v', \mathbf{x}') \in \delta_H(v, \mathbf{x})$ if either (1) there is a jump $e \in E$ from v to v' such that $update(e)[X, X' := \mathbf{x}, \mathbf{x}']$ is true, or (2) $v' = v$ and there is a real $\Delta \geq 0$ and a differentiable function $f : [0, \Delta] \to \mathbb{R}^m$ with first derivative $\dot{f}$ such that $f(0) = \mathbf{x}$ and $f(\Delta) = \mathbf{x}'$, and for all reals $\varepsilon \in (0, \Delta)$, both $inv(v)[X := f(\varepsilon)]$ and $flow(v)[\dot{X} := \dot{f}(\varepsilon)]$ are true. In case (2), the function f is called a *flow function*.

—$R_H = V \times \mathcal{L}^m(X)$; that is, every region (v, ϕ) consists of a location v (the discrete component of the region) and a linear predicate ϕ over X (the continuous component).

—$\ulcorner(v, \phi)\urcorner_H = \{(v, \mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^m \text{ and } \phi[X := \mathbf{x}] \text{ is true}\}$; that is, the extension function maps the continuous component ϕ of a region to the valuations for the variables in X which satisfy the predicate ϕ. Consequently, the extension of every region consists of a location and a polyhedral subset of $\mathbb{R}^m$.

—$P_H = V \times \{true\}$; that is, only the discrete component of a state is observable.

It requires some work to see that $\mathcal{S}_H$ is indeed a symbolic transition system. First, notice that the linear predicates over X are closed under all boolean operations, and that satisfiability is decidable for the linear predicates. Second, the *Pre* operator is computable on R_H, because all flow functions can be replaced by straight lines [Alur et al. 1996].

2.2 Background Definitions

The symbolic transition systems are a special case of transition systems. A *transition system* $\mathcal{S} = (Q, \delta, \cdot, \ulcorner\cdot\urcorner, P)$ has the same components as a symbolic transition system, except that no regions are specified and the extension function is defined only for the observables (that is, $\ulcorner\cdot\urcorner : P \to \mathbf{2}^Q$).

State equivalences. A *state equivalence* $\cong$ is a family of relations which contains for each transition system $\mathcal{S}$ an equivalence relation $\cong^{\mathcal{S}}$ on the states of $\mathcal{S}$. The $\cong$ *equivalence problem* for a class C of transition systems asks, given two states u and v of a transition system $\mathcal{S}$ from the class C, whether $u \cong^{\mathcal{S}} v$. The state equivalence $\cong_a$ is *as coarse as* the state equivalence $\cong_b$ if $u \cong_b^{\mathcal{S}} v$ implies $u \cong_a^{\mathcal{S}} v$ for all transition systems $\mathcal{S}$. The equivalence $\cong_a$ is *coarser than* $\cong_b$ if $\cong_a$ is as coarse as $\cong_b$, but $\cong_b$ is not as coarse as $\cong_a$. Given a transition system $\mathcal{S} = (Q, \delta, \cdot, \ulcorner\cdot\urcorner, P)$ and a state equivalence $\cong$, the *quotient system* is the transition system $\mathcal{S}/_{\cong} = (Q/_{\cong}, \delta/_{\cong}, \cdot, \ulcorner\cdot\urcorner/_{\cong}, P)$ with the following components:

—the states in $\mathcal{S}/_{\cong}$ are the equivalence classes of $\cong^{\mathcal{S}}$;

—$\tau \in \delta/_{\cong}(\sigma)$ if there is a state $u \in \sigma$ and a state $v \in \tau$ such that $v \in \delta(u)$;

—$\sigma \in \ulcorner p \urcorner/_{\cong}$ if there is a state $u \in \sigma$ such that $u \in \ulcorner p \urcorner$.

The quotient construction is of particular interest to us when it transforms an infinite-state system $\mathcal{S}$ into a finite-state system $\mathcal{S}/_{\cong}$.

State logics. A *state logic* L is a logic whose formulas are interpreted over the states of transition systems; that is, for every L-formula φ and every transition system $\mathcal{S}$, there is a set $[\![\varphi]\!]_\mathcal{S}$ of states of $\mathcal{S}$ which satisfy φ. The L *model-checking problem* for a class C of transition systems asks, given an L-formula φ and a state u of a transition system $\mathcal{S}$ from the class C, whether $u \in [\![\varphi]\!]_\mathcal{S}$. Two formulas φ and ψ of state logics are *equivalent* if $[\![\varphi]\!]_\mathcal{S} = [\![\psi]\!]_\mathcal{S}$ for all transition systems $\mathcal{S}$. The state logic L_a is *as expressive as* the state logic L_b if for every L_b-formula φ, there is an L_a-formula ψ which is equivalent to φ. The logics L_a and L_b are *equally expressive* if L_a is as expressive as L_b, and L_b is as expressive as L_a. The logic L_a is *more expressive than* L_b if L_a is as expressive as L_b, but L_b is not as expressive as L_a.

Every state logic L *induces* a state equivalence, denoted $\cong_L$: for all states u and v of a transition system $\mathcal{S}$, define $u \cong_L^\mathcal{S} v$ if for all L-formulas φ, we have $u \in [\![\varphi]\!]_\mathcal{S}$ iff $v \in [\![\varphi]\!]_\mathcal{S}$. We say that the state logic L *preserves* the state equivalence $\cong$ if $\cong_L$ is as coarse as $\cong$, and L *defines* $\cong$ over a class C of transition systems if $\cong_L^\mathcal{S}$ coincides with $\cong^\mathcal{S}$ for all transition systems $\mathcal{S}$ in C. The state logic L *admits abstraction* if for every L-formula φ and every transition system $\mathcal{S}$, we have $[\![\varphi]\!]_\mathcal{S} = \bigcup\{\sigma \mid \sigma \in [\![\varphi]\!]_{\mathcal{S}/\cong_L}\}$; that is, a state u of $\mathcal{S}$ satisfies an L-formula φ iff the $\cong_L$ equivalence class of u satisfies φ in the quotient system. Consequently, if L admits abstraction, then every L model-checking question on a transition system $\mathcal{S}$ can be reduced to an L model-checking question on the induced quotient system $\mathcal{S}/_{\cong_L}$. Below, we shall repeatedly prove the L model-checking problem for a class C of transition systems to be decidable by observing that for every transition system $\mathcal{S}$ from C, the quotient system $\mathcal{S}/_{\cong_L}$ has finitely many states and can be constructed effectively.

Symbolic semi-algorithms. A *symbolic semi-algorithm* takes as input the region algebra $\mathcal{R}_\mathcal{S} = (P, \mathit{Pre}, \mathit{And}, \mathit{Diff}, \mathit{Empty})$ of a symbolic transition system $\mathcal{S} = (Q, \delta, R, \ulcorner\cdot\urcorner, P)$, and generates regions in R using the operations P, Pre, And, Diff, and Empty. Depending on the input $\mathcal{S}$, a symbolic semi-algorithm on $\mathcal{S}$ may or may not terminate. The *dual* of a symbolic semi-algorithm is obtained by replacing the operators Pre, And, Diff, and Empty with their duals $\overline{\mathit{Pre}}$, $\overline{\mathit{And}}$, $\overline{\mathit{Diff}}$, and $\overline{\mathit{Empty}}$, respectively.

2.3 Preview

In sections 3–7 of this paper, we shall define five increasingly comprehensive classes $\mathsf{STS1}$–$\mathsf{STS5}$ of symbolic transition systems. In each case $i \in \{1, \dots, 5\}$, we will proceed in four steps:

1 Definition: Finite characterization. We give a state equivalence $\cong_i$ and define the class $\mathsf{STS}(i)$ to contain precisely the symbolic transition systems $\mathcal{S}$ for which the equivalence relation $\cong_i^\mathcal{S}$ has finite index (i.e., there are finitely many $\cong_i^\mathcal{S}$ equivalence classes). Each state equivalence $\cong_i$ is coarser than its predecessor $\cong_{i-1}$, which implies that $\mathsf{STS}(i-1) \subsetneq \mathsf{STS}(i)$ for $i \in \{2, \dots, 5\}$.

2 Algorithmics: Symbolic state-space exploration. We give a symbolic semi-algorithm A_i that terminates precisely on the symbolic transition systems in the class $\mathsf{STS}(i)$. This provides an operational characterization of the class $\mathsf{STS}(i)$ which is equivalent to the denotational definition of $\mathsf{STS}(i)$. The symbolic semi-algorithm A_i is called a *closure* semi-algorithm, because it closes the finite set P of observables under certain operations. The termination of the closure semi-algorithm A_i is proved by observing that, if given the region algebra of a symbolic transition system $\mathcal{S}$ as input, then the extensions of all regions generated by A_i are $\cong_i^{\mathcal{S}}$ blocks (i.e., unions of $\cong_i^{\mathcal{S}}$ equivalence classes). If $\mathcal{S}$ is in the class $\mathsf{STS}(i)$, then there are only finitely many $\cong_i^{\mathcal{S}}$ blocks, and the closure semi-algorithm A_i terminates upon having constructed a representation of the quotient system $\mathcal{S}/_{\cong_i}$. The closure semi-algorithm A_i can therefore be used to decide the $\cong_i$ equivalence problem for the class $\mathsf{STS}(i)$ of symbolic transition systems.

3 Verification: Decidable properties. We give a state logic L_i which admits abstraction and defines the state equivalence $\cong_i$ over the class $\mathsf{STS}(i)$ of symbolic transition systems. Since the finite $\cong_i$ quotient can be constructed effectively using the closure semi-algorithm A_i, it follows that the L_i model-checking problem for the class $\mathsf{STS}(i)$ is decidable. However, model-checking algorithms that rely on the explicit construction of quotient systems are usually impractical. Hence, we also present a symbolic semi-algorithm B_i that, given the region algebra of a symbolic transition system $\mathcal{S}$ from $\mathsf{STS}(i)$, and an L_i-formula φ, computes the region $[\![\varphi]\!]_{\mathcal{S}}$. The symbolic semi-algorithm B_i is called a *model-checking* semi-algorithm, because it directly solves all L_i model-checking questions for the class $\mathsf{STS}(i)$. The termination of the model-checking semi-algorithm B_i is proved by observing that B_i never generates any regions that are not generated by the closure semi-algorithm A_i. However, for a given system $\mathcal{S}$ and formula φ, the model-checking semi-algorithm B_i often generates fewer regions than the closure semi-algorithm A_i and is thus preferable in practice.

4 Example: Hybrid systems. The interesting members of the class $\mathsf{STS}(i)$ are those symbolic transition systems which have infinitely many states. In four out of the five cases STS1–STS5, following [Henzinger 1996], we provide certain kinds of polyhedral hybrid automata as examples. In this way, we obtain uniform decidability proofs and uniform symbolic model-checking algorithms for several classes of hybrid automata.

3. CLASS-1 SYMBOLIC TRANSITION SYSTEMS

The class-1 systems are characterized by finite bisimilarity quotients. The region algebra of a class-1 system has a finite subalgebra that contains the observables and is closed under *Pre*, *And*, and *Diff* operations. This enables the model checking of all μ-calculus properties. Infinite-state examples of class-1 systems are provided by the singular hybrid automata, and by the bakery protocol for mutual exclusion.

3.1 Finite Characterization: Bisimilarity

Definition: Bisimilarity. Let $\mathcal{S} = (Q, \delta, \cdot, \ulcorner\cdot\urcorner, P)$ be a transition system. A binary relation $\preceq$ on the state space Q is a *simulation* on $\mathcal{S}$ [Milner 1971] if $u \preceq v$ implies the following two conditions:

1. For each observable $p \in P$, we have $u \in \ulcorner p\urcorner$ iff $v \in \ulcorner p\urcorner$.
2. For each state $u' \in \delta(u)$, there is a state $v' \in \delta(v)$ such that $u' \preceq v'$.

Two states $u, v \in Q$ are *bisimilar*, denoted $u \cong_1^{\mathcal{S}} v$, if there is a symmetric simulation $\preceq$ on $\mathcal{S}$ such that $u \preceq v$. The state equivalence $\cong_1$ is called *bisimilarity*. $\square$

Definition: Class STS1. A symbolic transition system $\mathcal{S}$ belongs to the class STS1 if the bisimilarity relation $\cong_1^{\mathcal{S}}$ has finite index. $\square$

Symbolic semi-algorithm Closure1
Input: a region algebra $\mathcal{R} = (P, Pre, And, Diff, Empty)$.

$$\begin{array}{l} T_0 := P; \\ \textbf{for } i = 0, 1, 2, \ldots \textbf{ do} \\ \quad T_{i+1} := T_i \\ \qquad\qquad \cup \{Pre(\sigma) \mid \sigma \in T_i\} \\ \qquad\qquad \cup \{And(\sigma, \tau) \mid \sigma, \tau \in T_i\} \\ \qquad\qquad \cup \{Diff(\sigma, \tau) \mid \sigma, \tau \in T_i\} \\ \quad \textbf{until } \ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner. \end{array}$$

The termination test $\ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner$, which is shorthand for $\{\ulcorner \sigma \urcorner \mid \sigma \in T_{i+1}\} \subseteq \{\ulcorner \sigma \urcorner \mid \sigma \in T_i\}$, is decided as follows: for each region $\sigma \in T_{i+1}$ check that there is a region $\tau \in T_i$ such that both $Empty(Diff(\sigma, \tau))$ and $Empty(Diff(\tau, \sigma))$.

Fig. 1. Partition refinement.

3.2 Symbolic State-Space Exploration: Partition Refinement

The bisimilarity relation of a finite-state system can be computed by partition refinement [Kanellakis and Smolka 1990]. The symbolic semi-algorithm Closure1 of Figure 1 applies this method to infinite-state systems [Bouajjani et al. 1990; Henzinger 1995]. The semi-algorithm Closure1 inductively constructs regions, starting from the observables, by applying the three operations *Pre*, *And*, and *Diff*. It terminates when no new regions are generated. Suppose that the input given to Closure1 is the region algebra of a symbolic transition system $\mathcal{S} = (Q, \delta, R, \ulcorner\cdot\urcorner, P)$. Then each T_i, for $i \geq 0$, is a finite set of regions; that is, $T_i \subseteq R$. By induction it is easy to check that for all $i \geq 0$, the extension of every region in T_i is a $\cong_1^{\mathcal{S}}$ block. Thus, if $\cong_1^{\mathcal{S}}$ has finite index, then Closure1 terminates. Conversely, suppose that Closure1 terminates with $\ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner$. From the definition of bisimilarity it follows that if for each region $\sigma \in T_i$, we have $u \in \ulcorner \sigma \urcorner$ iff $v \in \ulcorner \sigma \urcorner$, then $u \cong_1^{\mathcal{S}} v$. This implies that $\cong_1^{\mathcal{S}}$ has finite index.

Theorem 1A *For all symbolic transition systems $\mathcal{S}$, the symbolic semi-algorithm* Closure1 *terminates on the region algebra $\mathcal{R}_{\mathcal{S}}$ iff $\mathcal{S}$ belongs to the class* STS1.

Corollary 1A *The $\cong_1$ (bisimilarity) equivalence problem is decidable for the class* STS1 *of symbolic transition systems.*

3.3 Decidable Properties: Branching Time

Definition: μ-calculus. The formulas of the *μ-calculus* [Kozen 1983] are generated by the grammar[2]

$$\varphi \ ::= \ p \mid \overline{p} \mid x \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists\bigcirc \varphi \mid \forall\bigcirc \varphi \mid (\mu x \colon \varphi) \mid (\nu x \colon \varphi),$$

for constants p from some set Π, and variables x from some set X. The $\bigcirc$ constructor is a "next-state" modality; the μ and ν constructors are least and greatest fixpoint quantifiers. Let $\mathcal{S} = (Q, \delta, \cdot, \ulcorner\cdot\urcorner, P)$ be a transition system whose observables include all constants; that is, $\Pi \subseteq P$. Let $\mathcal{E} : X \to \mathbf{2}^Q$ be a mapping from the variables to sets of states. We write $\mathcal{E}[x \mapsto \rho]$ for the mapping that agrees with

[2] We define all formulas to be in positive normal form, where only constants are negated.

$\mathcal{E}$ on all variables, except that $x \in X$ is mapped to $\rho \subseteq Q$. Given $\mathcal{S}$ and $\mathcal{E}$, every formula φ of the μ-calculus defines a set $[\![\varphi]\!]_{\mathcal{S},\mathcal{E}} \subseteq Q$ of states:

$$\begin{array}{rcl}
[\![p]\!]_{\mathcal{S},\mathcal{E}} &=& \ulcorner p \urcorner; \\
[\![\overline{p}]\!]_{\mathcal{S},\mathcal{E}} &=& Q \backslash \ulcorner p \urcorner; \\
[\![x]\!]_{\mathcal{S},\mathcal{E}} &=& \mathcal{E}(x); \\
[\![\varphi_1 \{{}^{\vee}_{\wedge}\} \varphi_2]\!]_{\mathcal{S},\mathcal{E}} &=& [\![\varphi_1]\!]_{\mathcal{S},\mathcal{E}} \ \{{}^{\cup}_{\cap}\} \ [\![\varphi_2]\!]_{\mathcal{S},\mathcal{E}}; \\
[\![\{{}^{\exists}_{\forall}\} \bigcirc \varphi]\!]_{\mathcal{S},\mathcal{E}} &=& \{u \in Q \mid (\{{}^{\exists}_{\forall}\} v \in \delta(u) \colon v \in [\![\varphi]\!]_{\mathcal{S},\mathcal{E}})\}; \\
[\![\{{}^{\mu}_{\nu}\} x \colon \varphi]\!]_{\mathcal{S},\mathcal{E}} &=& \{{}^{\cap}_{\cup}\} \{\rho \subseteq Q \mid \rho = [\![\varphi]\!]_{\mathcal{S},\mathcal{E}[x \mapsto \rho]}\}.
\end{array}$$

A variable x is called *free* in a formula φ if there is an occurrence of x in φ that is not in the scope of some μx or νx. A μ-calculus formula is *closed* if it has no free variables. If we restrict ourselves to the closed formulas of the μ-calculus, then we obtain a state logic, denoted L_1^μ: the state $u \in Q$ *satisfies* the L_1^μ-formula φ if $u \in [\![\varphi]\!]_{\mathcal{S},\mathcal{E}}$ for any variable mapping $\mathcal{E}$; that is, $[\![\varphi]\!]_{\mathcal{S}} = [\![\varphi]\!]_{\mathcal{S},\mathcal{E}}$ for any and all $\mathcal{E}$. □

Remark: Duality. For every L_1^μ-formula φ, the *dual* L_1^μ-formula $\overline{\varphi}$ is obtained by replacing the constructors p, $\overline{p}$, $\vee$, $\wedge$, $\exists\bigcirc$, $\forall\bigcirc$, μ, and ν by $\overline{p}$, p, $\wedge$, $\vee$, $\forall\bigcirc$, $\exists\bigcirc$, ν, and μ, respectively. Then, $[\![\overline{\varphi}]\!]_{\mathcal{S}} = Q \backslash [\![\varphi]\!]_{\mathcal{S}}$. It follows that the answer of the model-checking question for a state $u \in Q$ and an L_1^μ-formula φ is complementary to the answer of the model-checking question for u and the dual formula $\overline{\varphi}$. □

The following facts about the distinguishing and expressive powers of the μ-calculus are well-known. First, L_1^μ preserves $\cong_1$ (bisimilarity); indeed, the state equivalence defined by L_1^μ over the class STS1 of symbolic transition systems with finite bisimilarity quotients is exactly $\cong_1$ [Browne et al. 1988]. Second, L_1^μ is very expressive (see [Janin and Walukiewicz 1996] for a characterization); in particular, L_1^μ is more expressive than the temporal logics CTL and CTL* [Emerson 1990], which like L_1^μ define bisimilarity over the STS1 systems. Third, L_1^μ admits abstraction [Alur and Henzinger 1998] (the proof is by induction on L_1^μ-formulas). It follows that that the L_1^μ model-checking problem is decidable for the STS1 systems. However, we now present a model-checking algorithm for L_1^μ over STS1 systems which avoids the construction of bisimilarity quotients.

The definition of L_1^μ naturally suggests a model-checking algorithm for finite-state systems, where each fixpoint can be computed by successive approximation [Emerson and Lei 1986; Burch et al. 1992]. The symbolic semi-algorithm ModelCheck of Figure 2 applies this procedure to infinite-state systems. Suppose that the input given to ModelCheck is the region algebra of a symbolic transition system $\mathcal{S} = (Q, \delta, R, \ulcorner \cdot \urcorner, P)$, a μ-calculus formula φ, and any mapping $E \colon X \to \mathbf{2}^R$ from the variables to sets of regions. Then for each recursive call of ModelCheck, each T_i, for $i \geq 0$, is a finite set of regions from R, and each recursive call returns a finite set of regions from R. It is easy to check that all of these regions are also generated by the semi-algorithm Closure1 on input $\mathcal{R}_{\mathcal{S}}$. Thus, if Closure1 terminates, then so does ModelCheck. Furthermore, if it terminates, then ModelCheck returns a set $[\varphi]_E \subseteq R$ of regions such that $\bigcup\{\ulcorner \sigma \urcorner \mid \sigma \in [\varphi]_E\} = [\![\varphi]\!]_{\mathcal{S},\mathcal{E}}$, where $\mathcal{E}(x) = \bigcup\{\ulcorner \sigma \urcorner \mid \sigma \in E(x)\}$ for all $x \in X$. In particular, if φ is closed, then a state $u \in Q$ satisfies φ iff $Member(u, \sigma)$ for some region $\sigma \in [\varphi]_E$.

Theorem 1B *For all symbolic transition systems $\mathcal{S}$ in* STS1 *and every L_1^μ-formula φ,*

Symbolic semi-algorithm ModelCheck
Input: a region algebra $\mathcal{R} = (P, Pre, And, Diff, Empty)$, a formula $\varphi \in L_1^\mu$, and a mapping E with domain X.

Output: $[\varphi]_E :=$
if $\varphi = p$ **then return** $\{p\}$;
if $\varphi = \overline{p}$ **then return** $\{Diff(q, p) \mid q \in P\}$;
if $\varphi = x$ **then return** $E(x)$;
if $\varphi = (\varphi_1 \vee \varphi_2)$ **then return** $[\varphi_1]_E \cup [\varphi_2]_E$;
if $\varphi = (\varphi_1 \wedge \varphi_2)$ **then**
  return $\{And(\sigma, \tau) \mid \sigma \in [\varphi_1]_E$ and $\tau \in [\varphi_2]_E\}$;
if $\varphi = \exists\bigcirc\,\varphi'$ **then return** $\{Pre(\sigma) \mid \sigma \in [\varphi']_E\}$;
if $\varphi = \forall\bigcirc\,\varphi'$ **then return** $P \backslash\backslash \{Pre(\sigma) \mid \sigma \in (P \backslash\backslash [\varphi']_E)\}$;
if $\varphi = (\mu x\colon \varphi')$ **then**
  $T_0 := \emptyset$;
  for $i = 0, 1, 2, \ldots$ **do**
   $T_{i+1} := [\varphi']_{E[x \mapsto T_i]}$
   until $\bigcup\{\ulcorner\sigma\urcorner \mid \sigma \in T_{i+1}\} \subseteq \bigcup\{\ulcorner\sigma\urcorner \mid \sigma \in T_i\}$;
  return T_i;
if $\varphi = (\nu x\colon \varphi')$ **then**
  $T_0 := P$;
  for $i = 0, 1, 2, \ldots$ **do**
   $T_{i+1} := [\varphi']_{E[x \mapsto T_i]}$
   until $\bigcup\{\ulcorner\sigma\urcorner \mid \sigma \in T_{i+1}\} \supseteq \bigcup\{\ulcorner\sigma\urcorner \mid \sigma \in T_i\}$;
  return T_i.

The *pairwise-difference* operation $T \backslash\backslash T'$ between two finite sets T and T' of regions is computed inductively as follows:

$$T \backslash\backslash \emptyset \;=\; T;$$
$$T \backslash\backslash (\{\tau\} \cup T') \;=\; \{Diff(\sigma, \tau) \mid \sigma \in T\} \backslash\backslash T'.$$

The termination test $\bigcup\{\ulcorner\sigma\urcorner \mid \sigma \in T\} \subseteq \bigcup\{\ulcorner\sigma\urcorner \mid \sigma \in T'\}$ is decided by checking that $Empty(\sigma)$ for each region $\sigma \in (T \backslash\backslash T')$.

Fig. 2. Model checking.

the symbolic semi-algorithm ModelCheck *terminates on the region algebra* $\mathcal{R}_\mathcal{S}$ *and the input formula* φ.

Corollary 1B *The* L_1^μ *model-checking problem is decidable for the class* STS1 *of symbolic transition systems.*

Remark: Duality. Model checking of L_1^μ-formulas on STS1 systems can also be performed by the dual of the semi-algorithm ModelCheck. Suppose that the input given to the dual semi-algorithm $\overline{\text{ModelCheck}}$ is the dual region algebra of a symbolic transition system $\mathcal{S} = (Q, \delta, R, \ulcorner\cdot\urcorner, P)$, and the L_1^μ-formula φ. If $\mathcal{S}$ belongs to the class STS1, then $\overline{\text{ModelCheck}}$ terminates with the output $T \subseteq R$ such that $[\![\varphi]\!]_\mathcal{S} = \bigcap\{\ulcorner\sigma\urcorner \mid \sigma \in T\}$. $\square$

3.4 Example: The Bakery Protocol

Consider the 2-process bakery protocol [Lamport 1974] for mutual exclusion, which is presented as a collection of guarded commands in Figure 3. The program counters of the two processes, pc_1 and pc_2, each range over three locations: $pc_i = N$ if process i is not in its critical section; $pc_i = W$ if process i is waiting to enter its

$$
\begin{array}{lll}
 & \textbf{var } pc_1, pc_2 : \{N, W, C\} & \\
 & \textbf{var } y_1, y_2 : \mathbb{N} & \\
\| \; pc_1 = N & & \to pc_1, y_1 := W, y_2 + 1 \\
\| \; pc_1 = W \wedge (y_2 = 0 \vee y_1 \leq y_2) & & \to pc_1 := C \\
\| \; pc_1 = C & & \to pc_1, y_1 := N, 0 \\
\| \; pc_2 = N & & \to pc_2, y_2 := W, y_1 + 1 \\
\| \; pc_2 = W \wedge (y_1 = 0 \vee y_2 < y_1) & & \to pc_2 := C \\
\| \; pc_2 = C & & \to pc_2, y_2 := N, 0
\end{array}
$$

Fig. 3. The 2-process bakery protocol.

critical section; and $pc_i = C$ if process i is in its critical section. The protocol uses two variables y_1 and y_2, which range over the natural numbers and represent the "tokens" of the two processes. A state of the protocol, therefore, is a quadruple of values in $\{N, W, C\}^2 \times \mathbb{N}^2$. The observables are the boolean formulas over the program counters, in particular, the observable $p_{init} = (pc_1 = N \wedge pc_2 = N)$ is the initial region, and $\overline{q}_{mutex} = (pc_1 = C \wedge pc_2 = C)$ is the region that violates mutual exclusion. The values of the tokens are not observable.

The bisimilarity relation of this infinite-state system has finite index. To see this, define the binary relation $\approx$ on states as follows: for all states u and v, let $u \approx v$ iff (1) $u(pc_i) = v(pc_i)$ for $i = 1, 2$; (2) $u(y_i) = 0$ iff $v(y_i) = 0$ for $i = 1, 2$; and (3) $u(y_1) \leq u(y_2)$ iff $v(y_1) \leq v(y_2)$, where $u(x)$ denotes the value of variable x in state u. By a simple case enumeration, it can be checked that $\approx$ is a bisimulation (i.e., a symmetric simulation). Moreover, the relation has finite index (the number of equivalence classes is 72). Thus, the 2-process bakery protocol is in STS1. By Theorem 1A, the closure semi-algorithm Closure1 terminates on the region algebra of the protocol. By Theorem 1B, the model-checking semi-algorithm ModelCheck terminates as well for all L_1^μ-formulas. In particular, the region $[\![p_{init} \wedge (\mu x : \overline{q}_{mutex} \vee \exists \bigcirc x)]\!]$ can be computed to be empty in a finite number of steps. This proves that there is no sequence of transitions that leads from an initial state to a state that violates mutual exclusion; thus the protocol guarantees mutual exclusion.

3.5 Example: Singular Hybrid Automata

The fundamental theorem of timed automata [Alur and Dill 1994] shows that for every timed automaton, the (time-abstract) bisimilarity relation has finite index. The proof can be extended to the singular automata [Alur et al. 1995]. It follows that the symbolic semi-algorithm ModelCheck, which has been implemented for polyhedral hybrid automata in the tool HyTech [Henzinger et al. 1995], decides all L_1^μ model-checking questions for singular automata. The singular automata form a maximal class of hybrid automata in STS1. This is because there is a 2D (two-dimensional) rectangular automaton whose bisimilarity relation is state equality [Henzinger 1995].

Theorem 1C *The singular automata belong to the class* STS1. *There is a 2D rectangular automaton that does not belong to* STS1.

Symbolic semi-algorithm Closure2
Input: a region algebra $\mathcal{R} = (P, Pre, And, Diff, Empty)$.

$T_0 := P$;
for $i = 0, 1, 2, \ldots$ **do**
  $T_{i+1} := T_i$
    $\cup \{Pre(\sigma) \mid \sigma \in T_i\}$
    $\cup \{And(\sigma, \tau) \mid \sigma, \tau \in T_i\}$
  until $\ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner$.

The termination test $\ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner$ is decided as in Figure 1.

Fig. 4. Intersection refinement.

4. CLASS-2 SYMBOLIC TRANSITION SYSTEMS

The class-2 systems are characterized by finite similarity quotients. The region algebra of a class-2 system has a finite subalgebra that contains the observables and is closed under *Pre* and *And* operations. This enables the model checking of all existential and universal μ-calculus properties. Infinite-state examples of class-2 systems are provided by the 2D rectangular hybrid automata.

4.1 Finite Characterization: Similarity

Definition: Similarity. Let $\mathcal{S}$ be a transition system. Two states u and v of $\mathcal{S}$ are *similar*, denoted $u \cong_2^{\mathcal{S}} v$, if there are two simulations $\preceq_1$ and $\preceq_2$ on $\mathcal{S}$ such that $u \preceq_1 v$ and $v \preceq_2 u$. The state equivalence $\cong_2$ is called *similarity*. □

Definition: Class STS2**.** A symbolic transition system $\mathcal{S}$ belongs to the class STS2 if the similarity relation $\cong_2^{\mathcal{S}}$ has finite index. □

Since similarity is coarser than bisimilarity [van Glabbeek 1990], the class STS2 of symbolic transition systems is a proper extension of STS1.

4.2 Symbolic State-Space Exploration: Intersection Refinement

The symbolic semi-algorithm Closure2 of Figure 4 is an abstract version of the method presented in [Henzinger et al. 1995] for computing the similarity relation of an infinite-state system. Suppose that the input given to Closure2 is the region algebra of a symbolic transition system $\mathcal{S} = (Q, \delta, R, \ulcorner\cdot\urcorner, P)$. Given two states $u, v \in Q$, we say that v *simulates* u if $u \preceq v$ for some simulation $\preceq$ on $\mathcal{S}$. For $i \geq 0$ and $u \in Q$, define

$$Sim_i(u) \;=\; \bigcap\{\ulcorner\sigma\urcorner \mid \sigma \in T_i \text{ and } u \in \ulcorner\sigma\urcorner\},$$

where the set T_i of regions is computed by Closure2. By induction it is easy to check that for all $i \geq 0$, if v simulates u, then $v \in Sim_i(u)$. Thus, the extension of every region in T_i is a $\cong_2^{\mathcal{S}}$ block, and if $\cong_2^{\mathcal{S}}$ has finite index, then Closure2 terminates. Conversely, suppose that Closure2 terminates with $\ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner$. From the definition of simulations it follows that if $v \in Sim_i(u)$, then v simulates u. This implies that $\cong_2^{\mathcal{S}}$ has finite index.

Theorem 2A *For all symbolic transition systems $\mathcal{S}$, the symbolic semi-algorithm* Closure2 *terminates on the region algebra $\mathcal{R}_{\mathcal{S}}$ iff $\mathcal{S}$ belongs to the class* STS2.

Corollary 2A *The $\cong_2$ (similarity) equivalence problem is decidable for the class* STS2 *of symbolic transition systems.*

4.3 Decidable Properties: Negation-free Branching Time

Definition: Negation-free μ-calculus. The *negation-free μ-calculus* consists of the μ-calculus formulas that are generated by the grammar

$$\varphi \ ::= \ p \mid x \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists\bigcirc\, \varphi \mid (\mu x\colon \varphi) \mid (\nu x\colon \varphi),$$

for constants $p \in \Pi$ and variables $x \in X$. The state logic L_2^μ consists of the closed formulas of the negation-free μ-calculus. The state logic $\overline{L_2^\mu}$ consists of the duals of all L_2^μ-formulas. □

The following facts about the distinguishing and expressive powers of the negation-free μ-calculus and its dual are well-known. First, both L_2^μ and $\overline{L_2^\mu}$ preserve $\cong_2$ (similarity); indeed, the state equivalence defined by L_2^μ (respectively, $\overline{L_2^\mu}$) over the class STS2 of symbolic transition systems with finite similarity quotients is exactly $\cong_2$ [Loiseaux et al. 1995]. Therefore the μ-calculus L_1^μ with negation is more expressive than either L_2^μ or $\overline{L_2^\mu}$. Second, the negation-free μ-calculus L_2^μ is more expressive than the existential fragments of Ctl and Ctl* [Emerson 1990], which also define similarity over the STS2 systems. The dual logic $\overline{L_2^\mu}$ is more expressive than the universal fragments of Ctl and Ctl*, which again define similarity over the STS2 systems. Third, both L_2^μ and $\overline{L_2^\mu}$ admit abstraction [Alur and Henzinger 1998]. It follows that the L_2^μ and $\overline{L_2^\mu}$ model-checking problems are decidable for the STS2 systems. However, we now show that the procedure ModelCheck from Figure 2 decides the model-checking problems for L_2^μ and $\overline{L_2^\mu}$ over STS2 systems without constructing similarity quotients.

If we apply the symbolic semi-algorithm ModelCheck to the region algebra of a symbolic transition system $\mathcal{S}$ and an input formula from L_2^μ, then the cases $\varphi = \overline{p}$ and $\varphi = \forall\bigcirc\, \varphi'$ are never executed. It follows that all regions which are generated by ModelCheck are also generated by the semi-algorithm Closure2 on input $\mathcal{R}_\mathcal{S}$. Thus, if Closure2 terminates, then so does ModelCheck.

Theorem 2B *For all symbolic transition systems $\mathcal{S}$ in* STS2 *and every L_2^μ-formula φ, the symbolic semi-algorithm* ModelCheck *terminates on the region algebra $\mathcal{R}_\mathcal{S}$ and the input formula φ.*

Corollary 2B *The L_2^μ and $\overline{L_2^\mu}$ model-checking problems are decidable for the class* STS2 *of symbolic transition systems.*

4.4 Example: 2D Rectangular Hybrid Automata

For every 2D rectangular automaton, the (time-abstract) similarity relation has finite index [Henzinger et al. 1995]. It follows that the symbolic semi-algorithm ModelCheck, as implemented in HyTech, decides all L_2^μ and $\overline{L_2^\mu}$ model-checking questions for 2D rectangular automata. The 2D rectangular automata form a maximal class of hybrid automata in STS2. This is because there is a 3D rectangular automaton whose similarity relation is state equality [Henzinger and Kopke 1996].

Theorem 2C *The 2D rectangular automata belong to the class* STS2*. There is a 3D rectangular automaton that does not belong to* STS2*.*

5. CLASS-3 SYMBOLIC TRANSITION SYSTEMS

The class-3 systems are characterized by finite trace-equivalence quotients. The region algebra of a class-3 system has a finite subalgebra that contains the observables and is closed under *Pre* operations and those *And* operations for which one of the two arguments is an observable. This enables the model checking of all linear temporal properties. Infinite-state examples of class-3 systems are provided by the rectangular hybrid automata.

5.1 Finite Characterization: Traces

Definition: Trace equivalence. Let $\mathcal{S} = (Q, \delta, \cdot, \ulcorner\cdot\urcorner, P)$ be a transition system. A *trajectory* of $\mathcal{S}$ is a finite sequence $\theta = u_0 u_1 \ldots u_n$ of states $u_i \in Q$ such that $u_{i+1} \in \delta(u_i)$ for all $0 \leq i < n$. The first state u_0 is called the *source* of the trajectory θ, the last state u_n is its *sink*, and the *length* of the trajectory is n. An *observation* is a set of observables. Every state $u \in Q$ induces an observation, denoted by $\langle\langle u \rangle\rangle = \{p \in P \mid u \in \ulcorner p \urcorner\}$. A trace from state u is a finite sequence of observations which is induced by some trajectory with source u; that is, $\vartheta = \pi_0 \pi_1 \ldots \pi_n$ is a *trace from* $u \in Q$ if there exists a trajectory $u_0 u_1 \ldots u_n$ of $\mathcal{S}$ such that (1) $u_0 = u$ and (2) $\langle\langle u_i \rangle\rangle = \pi_i$ for all $0 \leq i \leq n$. The observables in the last observation π_n are called the *targets* of the trace ϑ, and its *length* is n. Two states $u, v \in Q$ are *trace equivalent*, denoted $u \cong_3^{\mathcal{S}} v$, if every trace from u is a trace from v, and vice versa. The state equivalence $\cong_3$ is called *trace equivalence*. □

Definition: Class STS3. A symbolic transition system $\mathcal{S}$ belongs to the class STS3 if the trace-equivalence relation $\cong_3^{\mathcal{S}}$ has finite index. □

Since trace equivalence is coarser than similarity [van Glabbeek 1990], the class STS3 of symbolic transition systems is a proper extension of STS2.

5.2 Symbolic State-Space Exploration: Observation Refinement

Trace equivalence can be characterized operationally by the symbolic semi-algorithm Closure3 of Figure 5. Suppose that the input given to Closure3 is the region algebra of a symbolic transition system $\mathcal{S} = (Q, \delta, R, \ulcorner\cdot\urcorner, P)$. By induction it is easy to check that for all $i \geq 0$, the extension of every region in T_i is a $\cong_3^{\mathcal{S}}$ block. Thus, if $\cong_3^{\mathcal{S}}$ has finite index, then Closure3 terminates. Conversely, suppose that Closure3 terminates with $\ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner$. Suppose further that there are two states $u, v \in Q$ and a trace $\pi_0 \pi_1 \ldots \pi_n$ from u which is not a trace from v. For $1 \leq j \leq n$, let $\hat{\pi}_j$ be the region that results from conjoining by *And* the observables in the finite set π_j (note that for every observable $p \in P$, either $p \in \pi_j$ or $\overline{p} \in \pi_j$). Then T_i contains the region $And(\hat{\pi}_0, Pre(And(\hat{\pi}_1, Pre(\cdots, And(\hat{\pi}_{n-1}, Pre(\hat{\pi}_n))))))$, which contains u but not v. It follows that if for each region $\sigma \in T_i$, we have $u \in \ulcorner \sigma \urcorner$ iff $v \in \ulcorner \sigma \urcorner$, then $u \cong_3^{\mathcal{S}} v$. This implies that $\cong_3^{\mathcal{S}}$ has finite index.

Theorem 3A *For all symbolic transition systems $\mathcal{S}$, the symbolic semi-algorithm* Closure3 *terminates on the region algebra $\mathcal{R}_{\mathcal{S}}$ iff $\mathcal{S}$ belongs to the class* STS3.

Corollary 3A *The $\cong_3$ (trace) equivalence problem is decidable for the class* STS3 *of symbolic transition systems.*

Remark: Infinite traces. We have defined trace equivalence based on finite traces only. An *ω-trajectory* of $\mathcal{S}$ is an infinite sequence $u_0 u_1 u_2 \ldots$ of states such

Symbolic semi-algorithm Closure3
Input: a region algebra $\mathcal{R} = (P, Pre, And, Diff, Empty)$.

$T_0 := P$;
for $i = 0, 1, 2, \ldots$ **do**
$T_{i+1} := T_i$
$\cup \{Pre(\sigma) \mid \sigma \in T_i\}$
$\cup \{And(\sigma, p) \mid \sigma \in T_i \text{ and } p \in P\}$
until $\ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner$.

The termination test $\ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner$ is decided as in Figure 1.

Fig. 5. Observation refinement.

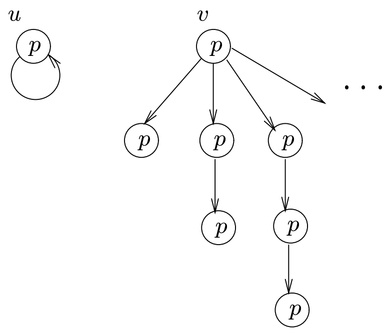

Fig. 6. Trace equivalence does not imply ω-trace equivalence.

that all finite prefixes of θ are trajectories of $\mathcal{S}$. An *ω-trace from* state $u \in Q$ is an infinite sequence of observations which is induced by some ω-trajectory of $\mathcal{S}$ with source u. Two states $u, v \in Q$ are *ω-trace equivalent* if every trace and ω-trace from u is also a trace or ω-trace from v, and vice versa. As transition systems may not be finitely branching, trace equivalence does not imply ω-trace equivalence. To see this, consider the states u and v of Figure 6, which are trace equivalent but not ω-trace equivalent. This is because u and v agree on all finite traces of the form $\{p\}^n$, for $n \geq 1$, but only u has the ω-trace $\{p\}^\omega$. Note that Algorithm Closure3 does not terminate on the symbolic transition system from Figure 6. The following lemma shows that if Algorithm Closure3 terminates, then trace equivalence and ω-trace equivalence coincide.

Lemma 3a *For all states u and v of a symbolic transition system $\mathcal{S}$ in the class* STS3*, if $u \cong_3^{\mathcal{S}} v$, then u and v are ω-trace equivalent.*

Proof. Suppose that Algorithm Closure3 terminates, and consider two states u and v that are trace equivalent. We show that every ω-trace from u is also an ω-trace from v. Consider an ω-trace $\vartheta = \pi_0\pi_1\pi_2\ldots$ from u. By way of contradiction, suppose that there is no ω-trajectory with source v which induces the ω-trace ϑ. For each $n \geq 1$, denote the prefix $\pi_0\pi_1\ldots\pi_n$ of ϑ by ϑ_n. We build the following subtree $\mathcal{T}$ of the unfolding of the transition function from state v: the root of $\mathcal{T}$ is $\langle v \rangle$; the nodes of $\mathcal{T}$ have the form $\langle u_0u_1\ldots u_n \rangle$, where $u_0u_1\ldots u_n$ is a trajectory with source v which induces the trace ϑ_n; there is an edge in $\mathcal{T}$ from $\langle u_0 u_1 \ldots u_n \rangle$ to $\langle u_0 u_1 \ldots u_{n+1} \rangle$ if $u_{n+1} \in \delta(u_n)$. We say that a node z of $\mathcal{T}$ is *infinitely reaching* if z has a descendant at distance n for all $n \geq 1$; and z is *infinitely branching* if z has infinitely many immediate descendants (at distance 1). Since u and v are trace equivalent, for each $n \geq 1$, there is a trajectory with source v which induces ϑ_n. It follows that the root of $\mathcal{T}$ is infinitely reaching. However, since there is no ω-trajectory with source v which induces ϑ, the tree $\mathcal{T}$ has no infinite path. By König's lemma, $\mathcal{T}$ must contain a node z which is both infinitely reaching and infinitely branching. For each $n \geq 1$, there must exist $f(n) \geq n$ such that z has an immediate successor $\langle \ldots u_{f(n)} \rangle$ which is the root of a subtree of height $f(n)$ but not $f(n)+1$. Then Algorithm Closure3 should have separated all states $u_{f(n)}$, for $n \geq 1$, into different equivalence classes, and hence could not have terminated. $\square$

5.3 Decidable Properties: Linear Time

Definition: Deterministic μ-calculus. The *deterministic μ-calculus* (also called "L_1" in [Emerson et al. 1993]) consists of the μ-calculus formulas that are generated by the grammar

$$\varphi \ ::= \ p \mid x \mid \varphi \vee \varphi \mid p \wedge \varphi \mid \exists \bigcirc \varphi \mid (\mu x \colon \varphi) \mid (\nu x \colon \varphi),$$

for constants $p \in \Pi$ and variables $x \in X$. The state logic L_3^μ consists of the closed formulas of the deterministic μ-calculus. The state logic $\overline{L_3^\mu}$ consists of the duals of all L_3^μ-formulas. $\square$

Following [Emerson et al. 2001], we show that L_3^μ has the same expressive power as existentially interpreted Büchi automata. We recall a few definitions. A *Büchi automaton* [Thomas 1990] is a tuple $\mathcal{B} = (S, \Phi, \rightarrow, s_0, F, B)$, where S is a finite set of states, Φ is a finite input alphabet, $\rightarrow \subseteq S \times \Phi \times S$ is a labeled transition relation, $s_0 \in S$ is the start state, $F \subseteq S$ is a set of finite accepting states, and $B \subseteq S$ is a set of Büchi accepting states. The automaton $\mathcal{B}$ *accepts a finite word* $w_0 w_1 \ldots w_n \in \Phi^*$ if there is a finite sequence $s_0 s_1 \ldots s_{n+1}$ of states $s_i \in S$, beginning from the start state s_0, such that (1) $s_i \xrightarrow{w_i} s_{i+1}$ for all $0 \leq i \leq n$, and (2) $s_{n+1} \in F$. The automaton $\mathcal{B}$ *accepts an infinite word* $w_0 w_1 w_2 \ldots \in \Phi^\omega$ if there is an infinite sequence $s_0 s_1 s_2 \ldots$ of states, beginning with s_0, such that (1) $s_i \xrightarrow{w_i} s_{i+1}$ for all $i \geq 0$, and (2) $s_i \in B$ for infinitely many $i \geq 0$. The state logic $\exists$Büchi has a formula of the form $\exists \mathcal{B}$ for every Büchi automaton $\mathcal{B}$ with $\Phi = \mathbf{2}^\Pi$; that is, each input letter is a set of constants. Then, given a transition system $\mathcal{S} = (Q, \delta, \cdot, \ulcorner \cdot \urcorner, P)$ with $\Pi \subseteq P$, and a state $u \in Q$, let $u \in [\![\exists \mathcal{B}]\!]_\mathcal{S}$ if there exist a trace or ω-trace $\vartheta = \pi_0 \pi_1 \pi_2 \ldots$ from u, and a finite or infinite word $\varpi = w_0 w_1 w_2 \ldots$ with the same length as ϑ, such that (1) $w_i \subseteq \pi_i$ for all $i \geq 0$, and (2) the word ϖ is accepted by $\mathcal{B}$. If (1) and (2), then we say that the trace or ω-trace ϑ is accepted by the Büchi automaton $\mathcal{B}$.

Lemma 3b *The state logics L_3^μ and $\exists$Büchi are equally expressive.*

Proof. The proof is based on the following constructions. By induction on the structure of an L_3^μ-formula φ, we can construct a Büchi automaton $\mathcal{B}_\varphi$ such that $\exists \mathcal{B}_\varphi$ is equivalent to φ; see [Emerson et al. 2001] for details. Conversely, given a Büchi automaton $\mathcal{B}$, we construct an L_3^μ-formula $\varphi_\mathcal{B}$ such that for all transition systems $\mathcal{S}$, a state u of $\mathcal{S}$ satisfies φ iff some trace or ω-trace from u is accepted by $\mathcal{B}_\varphi$. Let

$\mathcal{B} = (S, \mathbf{2}^P, \rightarrow, s_0, F, B)$ be a Büchi automaton over the alphabet $\mathbf{2}^P$. For notational convenience, we present the formula $\varphi_{\mathcal{B}}$ in equational form [Cleaveland et al. 1992]; it can be easily converted into the standard fixpoint representation by unrolling the equations and binding variables with least and greatest fixpoint quantifiers. For a set $\pi \subseteq P$ of observables, let $\tilde{\pi}$ be the conjunction $\bigwedge \pi$ of the observables in π. For each automaton state $s \in S$, we introduce a variable $x_s \in X$. If $s \in F$ is a finite accepting state, then $x_s = \tilde{\pi}$. If $s \notin F$, then the equation for x_s is

$$x_s \;=_\lambda\; \bigvee\{\tilde{\pi} \wedge \exists\bigcirc\, x_{s'} \mid s \xrightarrow{\pi} s'\}$$

where $\lambda = \nu$ if $s \in B$ is a Büchi accepting state, and $\lambda = \mu$ otherwise. The top-level variable is x_{s_0}, where s_0 is the start state of the automaton. The correctness of the procedure follows from [Bhat and Cleaveland 1996] (an equivalent construction is given in [Dam 1994]). $\square$

The state logic $\exists$Büchi, by its very definition, preserves ω-trace equivalence. Moreover, as every finite trace can be specified by a Büchi automaton, over the class STS3 of symbolic transition systems with finite trace-equivalence quotients, by Lemma 3a, the state equivalence defined by $\exists$Büchi is exactly $\cong_3$ (trace equivalence). By Lemma 3b, it follows that both the deterministic μ-calculus L_3^μ and its dual $\overline{L_3^\mu}$ also preserve ω-trace equivalence, and the state equivalence defined by L_3^μ (respectively, $\overline{L_3^\mu}$) over the STS3 systems is $\cong_3$. Therefore the negation-free μ-calculus L_2^μ with unrestricted conjunction is more expressive than L_3^μ, and $\overline{L_2^\mu}$ is more expressive than $\overline{L_3^\mu}$. Since Büchi automata are more expressive than the linear temporal logic Ltl [Wolper 1983], by Lemma 3b, the deterministic μ-calculus L_3^μ is more expressive than the existential interpretation of Ltl, which also defines trace equivalence over the STS3 systems. For example, the existential Ltl formula $\exists(p\mathcal{U}q)$ ("on some trace, p until q") is equivalent to the L_3^μ-formula $(\mu x\colon q \vee (p \wedge \exists\bigcirc\, x))$ (notice that one argument of the conjunction is a constant). The dual logic $\overline{L_3^\mu}$ is more expressive than the usual, universal interpretation of Ltl, which again defines trace equivalence over the STS3 systems. For example, the universal Ltl formula $\forall(p\mathcal{W}q)$ ("on all traces, either p forever, or p until q") is equivalent to the $\overline{L_3^\mu}$-formula $(\nu x : p \wedge \forall\bigcirc\, (q \vee x))$ (notice that one argument of the disjunction is a constant). Finally, both L_3^μ and $\overline{L_3^\mu}$ admit abstraction [Alur and Henzinger 1998]. It follows that the L_3^μ and $\overline{L_3^\mu}$ model-checking problems are decidable for the STS3 systems. However, we now show that the procedure ModelCheck from Figure 2 decides the model-checking problems for L_3^μ and $\overline{L_3^\mu}$ over STS2 systems without constructing trace-equivalence quotients.

If we apply the symbolic semi-algorithm ModelCheck to the region algebra of a symbolic transition system $\mathcal{S}$ and an input formula from L_3^μ, then all regions which are generated by ModelCheck are also generated by the semi-algorithm Closure3 on input $\mathcal{R}_{\mathcal{S}}$. Thus, if Closure3 terminates, then so does ModelCheck.

Theorem 3B *For all symbolic transition systems $\mathcal{S}$ in* STS3 *and every L_3^μ-formula φ, the symbolic semi-algorithm* ModelCheck *terminates on the region algebra $\mathcal{R}_{\mathcal{S}}$ and the input formula φ.*

Corollary 3B *The L_3^μ and $\overline{L_3^\mu}$ model-checking problems are decidable for the class* STS3 *of symbolic transition systems.*

Remark: LTL model checking. The LTL model-checking problem for a class C of transition systems asks, given an existential (respectively, universal) LTL formula $\exists\psi$ (respectively, $\forall\psi$) and a state u of a transition system from the class C, whether some (respectively, all) ω-traces from u satisfy ψ [Emerson 1990]. Our results suggest a symbolic procedure for LTL model checking over STS3 systems [Henzinger and Majumdar 2000]. Suppose that $\mathcal{S}$ is a symbolic transition system in the class STS3, and $\exists\psi$ is an existential LTL formula (for a universal LTL formula, take the negation). First, convert ψ into a Büchi automaton $\mathcal{B}_\psi$ using a tableau construction [Wolper 1983], and then into an equivalent L_3^μ-formula φ_ψ in equational form (introduce one variable x_s per state s of $\mathcal{B}_\psi$, as in the proof of Lemma 3B). Second, run the symbolic semi-algorithm ModelCheck on the inputs $\mathcal{R}_\mathcal{S}$ and φ_ψ.[3] It will terminate with a representation of the set of states that satisfy $\exists\psi$ in $\mathcal{S}$.

While ModelCheck provides, in this way, a symbolic semi-algorithm for model checking LTL, traditionally a different method has been used for the symbolic model checking of LTL formulas [Clarke et al. 1994]. Given a state u of a finite-state transition system $\mathcal{S}$, and an existential LTL formula $\exists\psi$, the model-checking question can be answered by constructing the product of $\mathcal{S}$ with the tableau automaton $\mathcal{B}_\psi$, and then checking the nonemptiness of a Büchi condition on the product structure. A Büchi condition is an LTL formula of the form $\Box\Diamond\phi$, where ϕ is a boolean combination of observables; therefore Büchi nonemptiness can be checked symbolically by evaluating the equivalent L_3^μ-formula

$$\chi \;=\; (\nu x_1 \colon (\mu x_2 \colon (\exists\bigcirc x_2) \vee (\phi \wedge \exists\bigcirc x_1))).$$

To extend this method to infinite-state systems, we need to be more formal. Let $\mathcal{S} = (Q, \delta, R, \ulcorner\cdot\urcorner, P)$ be a symbolic transition system, and let $\mathcal{B}_\psi = (S, \mathbf{2}^P, \to, s_0, \emptyset, B)$ be a Büchi automaton. Notice that since we interpret LTL only over ω-traces, the set F of finite accepting states of the Büchi automaton is empty. The *product structure* $\mathcal{S}_\psi = (S \times Q, \delta_\psi, S \times R, \ulcorner\cdot\urcorner_\psi, S \times P)$ is the following symbolic transition system with state space $S \times Q$, region set $S \times R$, and observable set $S \times P$: the extension $\ulcorner(s,\sigma)\urcorner_\psi$ is the set $\{s\} \times \ulcorner\sigma\urcorner$ of states, and $(s', u') \in \delta_\psi(s, u)$ iff (1) $s \xrightarrow{\pi} s'$ and $u' \in \delta(u)$, and (2) $u \in \ulcorner p \urcorner$ for all observables $p \in \pi$. Since the state space of the Büchi automaton $\mathcal{B}_\psi$ is finite, it is easy to check that $\mathcal{S}_\psi$ is again a symbolic transition system. Furthermore, for all states $u \in Q$, we have $u \in [\![\exists\mathcal{B}_\psi]\!]_\mathcal{S}$ iff $(s_0, u) \in [\![\Box\Diamond\phi]\!]_{\mathcal{S}_\psi}$, where $\phi = \bigvee_{s \in B, p \in P}(s, p)$.

Let DirectCheck be the instance of the symbolic semi-algorithm ModelCheck which evaluates the L_3^μ-formula φ_ψ on the region algebra of the symbolic transition system $\mathcal{S}$, and let ProductCheck be the instance of ModelCheck which evaluates the L_3^μ-formula χ on the region algebra of the product structure $\mathcal{S}_\psi$. We show that there is a direct correspondence between the regions computed by ProductCheck and the regions computed by DirectCheck; in fact, the algorithm DirectCheck mimics the evaluation of χ with respect to $\mathcal{S}_\psi$. To see this, we first write the formula χ in equational form as $x_1 =_\nu x_2$ and $x_2 =_\mu ((\exists\bigcirc x_2) \vee (\phi \wedge \exists\bigcirc x_1))$. Now we

[3] In practice, we run a variant of ModelCheck directly on the equational representation of the L_3^μ-formula φ_ψ [Cleaveland et al. 1992]. This is because in equational form, the size of φ_ψ is linear in the size of the Büchi automaton $\mathcal{B}_\psi$ (and exponential in the size of the LTL formula ψ). Conversion of φ_ψ into fixpoint representation would involve another exponential.

introduce two sets of variables, $\{x_1^s \mid s \in S\}$ and $\{x_2^s \mid s \in S\}$. For $j = 1, 2$, we use the variable x_j^s to track the value of x_j on the product structure at the automaton state s, that is, at each step, $x_1 = \bigvee\{x_1^s \mid s \in S\}$ and $x_2 = \bigvee\{x_2^s \mid s \in S\}$. With these new variables, the equations for χ become $x_1^s =_\nu x_2^s$ for all $s \in S$, and $x_2^s =_\mu \bigvee\{\tilde{\pi} \wedge \exists\bigcirc x_1^{s'} \mid s \xrightarrow{\pi} s'\}$ for all $s \in B$, and $x_2^s =_\mu \bigvee\{\tilde{\pi} \wedge \exists\bigcirc x_2^{s'} \mid s \xrightarrow{\pi} s'\}$ for all $s \in S \backslash B$. The top-level variable is $x_1^{s_0}$. From this, it is clear that these two methods are equivalent in the regions they compute. It follows that ProductCheck terminates iff DirectCheck does. $\square$

Corollary 3B* *For all symbolic transition systems $\mathcal{S}$ in* STS3*, and every existential* LTL *formula $\exists\psi$, the symbolic semi-algorithm* ProductCheck *terminates for $\mathcal{S}$ and ψ.*

5.4 Example: Rectangular Hybrid Automata

For every rectangular automaton, the (time-abstract) trace-equivalence relation has finite index [Henzinger et al. 1998]. It follows that the symbolic semi-algorithm ModelCheck, as implemented in HyTech, decides all L_3^μ and $\overline{L_3^\mu}$ model-checking questions for rectangular automata. The rectangular automata form a maximal class of hybrid automata in STS3. This is because for simple generalizations of rectangular automata, the reachability problem is undecidable [Henzinger et al. 1998].

Theorem 3C *The rectangular automata belong to the class* STS3*. There is a polyhedral hybrid automaton that does not belong to* STS3*.*

6. CLASS-4 SYMBOLIC TRANSITION SYSTEMS

We define two states of a transition system to be "distance equivalent" if for every distance d, the same observables can be reached in d transitions. The class-4 systems are characterized by finite distance-equivalence quotients. The region algebra of a class-4 system has a finite subalgebra that contains the observables and is closed under *Pre* operations. This enables the model checking of all existential conjunction-free and universal disjunction-free μ-calculus properties, such as the property that an observable can be reached in an even number of transitions.

6.1 Finite Characterization: Equi-distant Targets

Definition: Distance equivalence. Let $\mathcal{S}$ be a transition system. Two states u and v of $\mathcal{S}$ are *distance equivalent*, denoted $u \cong_4^{\mathcal{S}} v$, if for every trace from u with length n and target p, there is a trace from v with length n and target p, and vice versa. The state equivalence $\cong_4$ is called *distance equivalence.* $\square$

Definition: Class STS4. A symbolic transition system $\mathcal{S}$ belongs to the class STS4 if the distance-equivalence relation $\cong_4^{\mathcal{S}}$ has finite index. $\square$

Figure 7 shows that distance equivalence is coarser than trace equivalence: the states u and v are distance equivalent but not trace equivalent. It follows that the class STS4 of symbolic transition systems is a proper extension of STS3.

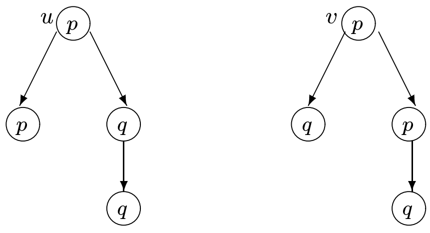

Fig. 7. Distance equivalence is coarser than trace equivalence.

Symbolic semi-algorithm Closure4
Input: a region algebra $\mathcal{R} = (P, Pre, And, Diff, Empty)$.

$T_0 := P$;
for $i = 0, 1, 2, \ldots$ **do**
$T_{i+1} := T_i$
$\cup \{Pre(\sigma) \mid \sigma \in T_i\}$
until $\ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner$.

The termination test $\ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner$ is decided as in Figure 1.

Fig. 8. Predecessor iteration.

6.2 Symbolic State-Space Exploration: Predecessor Iteration

The symbolic semi-algorithm Closure4 of Figure 8 computes the subalgebra of a region algebra $\mathcal{R}_{\mathcal{S}}$ that contains the observables and is closed under the *Pre* operation. Suppose that the input given to Closure4 is the region algebra of a symbolic transition system $\mathcal{S} = (Q, \delta, R, \ulcorner \cdot \urcorner, P)$. For $i \geq 0$ and two states $u, v \in Q$, define $u \sim_i^{\mathcal{S}} v$ if for every trace from u with length $n \leq i$ and target p, there is a trace from v with length n and target p, and vice versa. By induction it is easy to check that for all $i \geq 0$, the extension of every region in T_i, as computed by Closure4, is a $\sim_i^{\mathcal{S}}$ block. Since $\sim_i^{\mathcal{S}}$ is as coarse as $\sim_{i+1}^{\mathcal{S}}$ for all $i \geq 0$, and $\cong_4^{\mathcal{S}}$ is equal to $\bigcap\{\sim_i^{\mathcal{S}} \mid i \geq 0\}$, if $\cong_4^{\mathcal{S}}$ has finite index, then $\cong_4^{\mathcal{S}}$ is equal to $\sim_j^{\mathcal{S}}$ for some $j \geq 0$. Then, Closure4 will terminate in j iterations. Conversely, suppose that Closure4 terminates with $\ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner$. In this case, if for all regions $\sigma \in T_i$, we have $u \in \ulcorner \sigma \urcorner$ iff $v \in \ulcorner \sigma \urcorner$, then $u \cong_4^{\mathcal{S}} v$. This is because if u can reach an observable p in n transitions, but v cannot, then there is a region in T_i, namely, $Pre^n(p)$, such that $u \in \ulcorner Pre^n(p) \urcorner$ and $v \notin \ulcorner Pre^n(p) \urcorner$. It follows that $\cong_4^{\mathcal{S}}$ has finite index.

Theorem 4A *For all symbolic transition systems $\mathcal{S}$, the symbolic semi-algorithm* Closure4 *terminates on the region algebra $\mathcal{R}_{\mathcal{S}}$ iff $\mathcal{S}$ belongs to the class* STS4*.*

Corollary 4A *The $\cong_4$ (distance) equivalence problem is decidable for the class* STS4 *of symbolic transition systems.*

6.3 Decidable Properties: Conjunction-free Linear Time

Definition: Conjunction-free μ-calculus. The *conjunction-free μ-calculus* consists of the μ-calculus formulas that are generated by the grammar

$$\varphi \ ::= \ p \mid x \mid \varphi \vee \varphi \mid \exists\bigcirc\,\varphi \mid (\mu x\colon \varphi)$$

for constants $p \in \Pi$ and variables $x \in X$. The state logic L_4^μ consists of the closed formulas of the conjunction-free μ-calculus. The state logic $\overline{L_4^\mu}$ consists of the duals of all L_4^μ-formulas. $\Box$

Definition: Conjunction-free temporal logic. The formulas of the *conjunction-free temporal logic* $L_4^\Diamond$ are generated by the grammar

$$\varphi \ ::= \ p \mid \varphi \vee \varphi \mid \exists\bigcirc\,\varphi \mid \exists\Diamond_{\leq d}\,\varphi \mid \exists\Diamond\varphi,$$

for constants $p \in \Pi$ and nonnegative integers $d \geq 0$. Let $\mathcal{S} = (Q, \delta, \cdot, \ulcorner\cdot\urcorner, P)$ be a transition system whose observables include all constants; that is, $\Pi \subseteq P$. The $L_4^\Diamond$-formula φ defines the set $[\![\varphi]\!]_\mathcal{S} \subseteq Q$ of satisfying states:

$$\begin{array}{rcl}
[\![p]\!]_\mathcal{S} &=& \ulcorner p \urcorner; \\
[\![\varphi_1 \vee \varphi_2]\!]_\mathcal{S} &=& [\![\varphi_1]\!]_\mathcal{S} \cup [\![\varphi_2]\!]_\mathcal{S}; \\
[\![\exists\bigcirc\,\varphi]\!]_\mathcal{S} &=& \{u \in Q \mid (\exists v \in \delta(u)\colon v \in [\![\varphi]\!]_\mathcal{S})\}; \\
[\![\exists\Diamond_{\leq d}\,\varphi]\!]_\mathcal{S} &=& \{u \in Q \mid \text{there is a trajectory of } \mathcal{S} \text{ with source } u, \\
&& \qquad \text{length at most } d, \text{ and sink in } [\![\varphi]\!]_\mathcal{S}\}; \\
[\![\exists\Diamond\varphi]\!]_\mathcal{S} &=& \{u \in Q \mid \text{there is a trajectory of } \mathcal{S} \text{ with source } u \text{ and} \\
&& \qquad \text{sink in } [\![\varphi]\!]_\mathcal{S}\}. \quad \Box
\end{array}$$

Note that the $L_4^\Diamond$-formula $\exists\Diamond\varphi$ is equivalent to the L_4^μ-formula $(\mu x : \varphi \vee \exists\bigcirc\, x)$. Moreover, the constructor $\exists\Diamond_{\leq d}$ is definable from $\exists\bigcirc$ and $\vee$; however, it will be essential in the $\exists\bigcirc$-free fragment of $L_4^\Diamond$ we will consider below.

Remark: Duality. For every $L_4^\Diamond$-formula φ, the *dual* formula $\overline{\varphi}$ is obtained by replacing the constructors p, $\vee$, $\exists\bigcirc$, $\exists\Diamond_{\leq d}$, and $\exists\Diamond$ by $\overline{p}$, $\wedge$, $\forall\bigcirc$, $\forall\Box_{\leq d}$, and $\forall\Box$, respectively. The semantics of the dual constructors is defined as usual, such that $[\![\overline{\varphi}]\!]_\mathcal{S} = Q \backslash [\![\varphi]\!]_\mathcal{S}$. The state logic $\overline{L_4^\Diamond}$ consists of the duals of all $L_4^\Diamond$-formulas. It follows that the answer of the model-checking question for a state $u \in Q$ and an $\overline{L_4^\Diamond}$-formula $\overline{\varphi}$ is complementary to the answer of the model-checking question for u and the $L_4^\Diamond$-formula φ. $\Box$

The following easy facts about the conjunction-free μ-calculus, conjunction-free temporal logic, and their duals are relevant in our context. First, the state equivalence induced by both L_4^μ and $\overline{L_4^\mu}$ is $\cong_4$ (distance equivalence). Therefore the deterministic μ-calculus L_3^μ is more expressive than L_4^μ, and $\overline{L_3^\mu}$ is more expressive than $\overline{L_4^\mu}$. Second, the conjunction-free μ-calculus L_4^μ is more expressive than the conjunction-free temporal logic $L_4^\Diamond$, and $\overline{L_4^\mu}$ is more expressive than $\overline{L_4^\Diamond}$, both of which also induce distance equivalence. For example, the property that an observable p can be reached in an even number of transitions can be expressed in L_4^μ by the formula $(\mu x : p \vee \exists\bigcirc\, \exists\bigcirc\, x)$, for which there is no equivalent $L_4^\Diamond$-formula [Wolper 1983]. Third, both L_4^μ and $\overline{L_4^\mu}$ admit abstraction. It follows that the L_4^μ and $\overline{L_4^\mu}$ model-checking problems are decidable for the STS4 systems. As usual, we now show that the procedure ModelCheck from Figure 2 decides the model-checking problems for L_4^μ and $\overline{L_4^\mu}$ over STS4 systems without constructing distance-equivalence quotients.

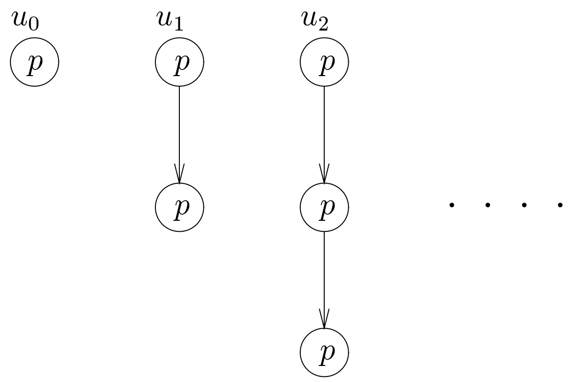

Fig. 9. Bounded-reach equivalence is coarser than distance equivalence.

If we apply the symbolic semi-algorithm ModelCheck of to the region algebra of a symbolic transition system $\mathcal{S}$ and an input formula from L_4^μ, then all regions which are generated by ModelCheck are also generated by the semi-algorithm Closure4 on input $\mathcal{R}_\mathcal{S}$. Thus, if Closure4 terminates, then so does ModelCheck.

Theorem 4B *For all symbolic transition systems $\mathcal{S}$ in* STS4 *and every L_4^μ-formula φ, the symbolic semi-algorithm* ModelCheck *terminates on the region algebra $\mathcal{R}_\mathcal{S}$ and the input formula φ.*

Corollary 4B *The L_4^μ and $\overline{L_4^\mu}$ model-checking problems are decidable for the class* STS4 *of symbolic transition systems.*

7. CLASS-5 SYMBOLIC TRANSITION SYSTEMS

We define two states of a transition system to be "bounded-reach equivalent" if for every distance d, the same observables can be reached in d or fewer transitions. The class-5 systems are characterized by finite bounded-reach-equivalence quotients. Equivalently, for every observable p there is a finite bound n_p such that all states that can reach p can do so in at most n_p transitions. This enables the model checking of all reachability and (by duality) invariance properties. The transition systems in class 5 have also been called "well-structured" [Abdulla et al. 1996]. Infinite-state examples of class-5 systems are provided by networks of timed automata.

7.1 Finite Characterization: Bounded-distance Targets

Definition: Bounded-reach equivalence. Let $\mathcal{S}$ be a transition system. Two states u and v of $\mathcal{S}$ are *bounded-reach equivalent*, denoted $u \cong_5^\mathcal{S} v$, if for every trace from u with length n and target p, there is a trace from v with length at most n and target p, and vice versa. The state equivalence $\cong_5$ is called *bounded-reach equivalence*. □

Definition: Class STS5. A symbolic transition system $\mathcal{S}$ belongs to the class STS5 if the bounded-reach-equivalence relation $\cong_\mathcal{S}^5$ has finite index. □

Figure 9 shows that bounded-reach equivalence is coarser than distance equivalence:

Symbolic semi-algorithm Reach
Input: a region algebra $\mathcal{R} = (P, Pre, And, Diff, Empty)$.

for each $p \in P$ **do**
$T_0 := \{p\}$;
for $i = 0, 1, 2, \ldots$ **do**
$T_{i+1} := T_i \cup \{Pre(\sigma) \mid \sigma \in T_i\}$
until $\bigcup\{\ulcorner\sigma\urcorner \mid \sigma \in T_{i+1}\} \subseteq \bigcup\{\ulcorner\sigma\urcorner \mid \sigma \in T_i\}$
end.

The termination test $\bigcup\{\ulcorner\sigma\urcorner \mid \sigma \in T_{i+1}\} \subseteq \bigcup\{\ulcorner\sigma\urcorner \mid \sigma \in T_i\}$ is decided as in Figure 2.

Fig. 10. Predecessor aggregation.

all states u_i, for $i \geq 0$, are bounded-reach equivalent, but no two of them are distance equivalent. It follows that the class STS5 of symbolic transition systems is a proper extension of STS4.

7.2 Symbolic State-Space Exploration: Predecessor Aggregation

The symbolic semi-algorithm Reach of Figure 10 starts from the observables and repeatedly applies the *Pre* operation, but its termination criterion is more easily met than the termination criterion of the semi-algorithm Closure4; that is, Reach may terminate on more inputs than Closure4. Indeed, we shall show that, if the input is the region algebra of a symbolic transition system $\mathcal{S} = (Q, \delta, R, \ulcorner\cdot\urcorner, P)$, then Reach terminates iff $\mathcal{S}$ belongs to the class STS5. Furthermore, upon termination, $u \cong_5^{\mathcal{S}} v$ iff for each observable $p \in P$ and each region $\sigma \in T_i^p$, we have $u \in \ulcorner\sigma\urcorner$ iff $v \in \ulcorner\sigma\urcorner$.

An alternative characterization of the class STS5 can be given using well-quasi-orders on states [Abdulla et al. 1996; Finkel and Schnoebelen 1998]. A *quasi-order* on a set A is a reflexive and transitive binary relation on A. A *well-quasi-order* on A is a quasi-order $\preceq$ on A such that for every infinite sequence $a_0, a_1, a_2, \ldots$ of elements $a_i \in A$ there exist indices i and j with $i < j$ and $a_i \preceq a_j$. A set $B \subseteq A$ is *upward-closed* if for all $b \in B$ and $a \in A$, if $b \preceq a$, then $a \in B$. It can be shown that if $\preceq$ is a well-quasi-order on A, then every infinite increasing sequence $B_0 \subseteq B_1 \subseteq B_2 \subseteq \cdots$ of upward-closed sets $B_i \subseteq A$ eventually stabilizes; that is, there exists an index $i \geq 0$ such that $B_j = B_i$ for all $j \geq i$.

Theorem 5A *For all symbolic transition systems $\mathcal{S}$, the following three conditions are equivalent:*

1. *$\mathcal{S}$ belongs to the class* STS5.
2. *The symbolic semi-algorithm* Reach *terminates on the region algebra $\mathcal{R}_{\mathcal{S}}$.*
3. *There is a well-quasi-order $\preceq$ on the states of $\mathcal{S}$ such that for all observables p and all nonnegative integers d, the set $[\![\exists\Diamond_{\leq d}\, p]\!]_{\mathcal{S}}$ is upward-closed.*

Proof. $(2 \Rightarrow 1)$ Define $u \sim_{\leq n}^{\mathcal{S}} v$ if for all observables $p \in P$, for every trace from u with length n and target p, there is a trace from v with length at most n and target p, and vice versa. Note that $\sim_{\leq n}^{\mathcal{S}}$ has finite index for all $n \geq 0$. Suppose

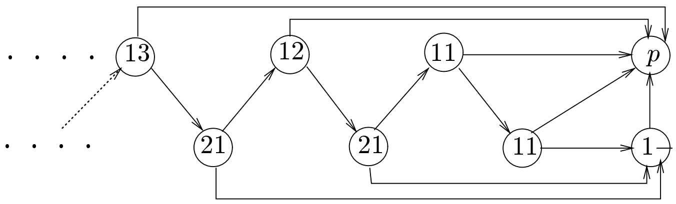

Fig. 11. An STS5 system on which L_4^μ model checking does not terminate.

that the semi-algorithm Reach terminates, for each observable $p \in P$, in at most i iterations. Then for all $n \geq i$, the equivalence relation $\sim^{\mathcal{S}}_{\leq n}$ is equal to $\sim^{\mathcal{S}}_{\leq i}$. Since $\cong^{\mathcal{S}}_5$ is equal to $\bigcap\{\sim^{\mathcal{S}}_{\leq n} \mid n \geq 0\}$, it has finite index.

($1 \Rightarrow 3$) Define the quasi-order $u \preceq^{\mathcal{S}}_5 v$ if for all observables $p \in P$ and all $n \geq 0$, for every trace from u with length n and target p, there is a trace from v with length at most n and target p. Then each set $[\![\exists\Diamond_{\leq d}\, p]\!]_{\mathcal{S}}$, for an observable $p \in P$ and a nonnegative integer $d \geq 0$, is upward-closed with respect to $\preceq^{\mathcal{S}}_5$. Furthermore, if $\cong^{\mathcal{S}}_5$ has finite index, then $\preceq^{\mathcal{S}}_5$ is a well-quasi-order. This is because $u \cong^{\mathcal{S}}_5 v$ implies $u \preceq^{\mathcal{S}}_5 v$: if there were an infinite sequence $u_0, u_1, u_2, \ldots$ of states such that for all $i \geq 0$ and $j < i$, we have $u_j \not\preceq^{\mathcal{S}}_5 u_i$, then no two of these states would be $\cong^{\mathcal{S}}_5$ equivalent.

($3 \Rightarrow 2$) This part of the proof follows immediately from the stabilization property of well-quasi-orders [Abdulla et al. 1996]. □

7.3 Decidable Properties: Bounded Reachability

Definition: Bounded-reachability logic. The *bounded-reachability logic* $L_5^\Diamond$ consists of the $L_4^\Diamond$-formulas that are generated by the grammar

$$\varphi \;::=\; p \mid \varphi \vee \varphi \mid \exists\Diamond_{\leq d}\, \varphi \mid \exists\Diamond\varphi,$$

for constants $p \in \Pi$ and nonnegative integers d. The state logic $\overline{L_5^\Diamond}$ consists of the duals of all $L_5^\Diamond$-formulas. □

The following easy facts about bounded-reachability logic and its dual are relevant in our context. First, the state equivalence induced by both $L_5^\Diamond$ and $\overline{L_5^\Diamond}$ is $\cong_5$ (bounded-reach equivalence). Therefore the conjunction-free temporal logic $L_4^\Diamond$ is more expressive than $L_5^\Diamond$, and $\overline{L_4^\Diamond}$ is more expressive than $\overline{L_5^\Diamond}$. For example, the property that an observable p can be reached in exactly d transitions can be expressed in $L_4^\Diamond$ by the formula $\exists\bigcirc \ldots \exists\bigcirc\, p$ with d next operators, for which there is no equivalent $L_5^\Diamond$-formula. Second, both $L_5^\Diamond$ and $\overline{L_5^\Diamond}$ admit abstraction. Since for STS5 systems the bounded-reach-equivalence quotient can be constructed using the symbolic semi-algorithm Reach, we have the following theorem.

Theorem 5B *The $L_5^\Diamond$ and $\overline{L_5^\Diamond}$ model-checking problems are decidable for the class* STS5 *of symbolic transition systems.*

A direct symbolic model-checking semi-algorithm for $L_5^\Diamond$ and, indeed, $L_4^\Diamond$ is easily derived from the semi-algorithm Reach. Then, if Reach terminates, so does model checking for all $L_4^\Diamond$-formulas, including unbounded $\exists\Diamond$ properties. The extension to $L_4^{\diamond}$ is possible, because $\exists\bigcirc$ properties pose no threat to termination. However the same is not true for L_4^{μ}. Figure 7.3 shows a symbolic transition system from the class STS5 for which the evaluation of the L_4^{μ}-formula $(\mu x\colon p \vee \exists\bigcirc\exists\bigcirc x)$ using the symbolic semi-algorithm ModelCheck does not terminate (note that in this example, the symbolic semi-algorithm Reach with *Pre* replaced by Pre^2, which uses a more relaxed termination criterion, does not terminate either). We show that this is not suprising as L_4^{μ} is undecidable on STS5 systems. To establish this result, we proceed as follows. Given a two-counter machine M, we define a symbolic transition system $\mathcal{S}_M$ which belongs to the class STS5, and which encodes the computations of M such that the L_4^{μ}-formula $(\mu x : p_{\mathit{fin}} \vee \exists\bigcirc\exists\bigcirc x)$ characterizes exactly the set of configurations of M from which there is a halting computation.

Let $M = \langle K, C, D, I\rangle$ be a *two-counter machine*, where $K = \{\ell_0, \ldots, \ell_m\}$ is a finite set of control locations with initial location ℓ_0 and final location ℓ_m, the nonnegative integer variables C and D are the two counters, and I is a function that labels each nonfinal location with an instruction (increment a counter, decrement a counter, or test a counter for zero) and a successor location (in the case of zero-test there are two successor locations). A *configuration* of M is a triple $\gamma = \langle \ell, c, d\rangle$, where $\ell \in K$ is the value of the program counter, which indicates the current control location, and $c, d \in \mathbb{N}$ are the values of the two counters C and D. The configuration γ is *final* if $\ell = \ell_m$. We write Γ for the set of configurations of M, and $\Gamma_{\mathit{fin}} \subseteq \Gamma$ for the set of final configurations. If $\gamma \notin \Gamma_{\mathit{fin}}$, then $M(\gamma)$ denotes the *successor configuration* of γ, which results from γ by executing the instruction $I(\ell)$. A *computation of M from configuration* γ is a finite sequence $\gamma_0, \gamma_1, \ldots, \gamma_k$ of configurations such that (1) $\gamma_0 = \gamma$ and (2) $\gamma_{i+1} = M(\gamma_i)$ for all $0 \le i < k$. The computation is *halting* if $\gamma_k \in \Gamma_{\mathit{fin}}$. The problem of deciding if a two-counter machine has a halting computation from the initial configuration $\langle \ell_0, 0, 0\rangle$ is undecidable [Hopcroft and Ullman 1979].

We define the symbolic transition system $\mathcal{S}_M = (Q, \delta, R, \ulcorner\cdot\urcorner, P)$ which encodes the computations of M as follows.

—The state space $Q = \Gamma \times \{1, 2\}$ contains two copies for each configuration of M.

—The transition function δ is defined as follows: $(\gamma', j') \in \delta(\gamma, j)$ if either (1) $\gamma \notin \Gamma_{\mathit{fin}}$ and $j = 1$ and $\gamma' \in \Gamma_{\mathit{fin}}$ and $j' = 1$ (that is, every copy 1 of a nonfinal configuration has a transition to every copy 1 of a final configuration); or (2) $\gamma \notin \Gamma_{\mathit{fin}}$ and $j = 1$ and $\gamma' = \gamma$ and $j' = 2$ (that is, every copy 1 of a nonfinal configuration has a transition to the copy 2 of the same configuration); or (3) $\gamma \notin \Gamma_{\mathit{fin}}$ and $j = 2$ and $\gamma' = M(\gamma)$ and $j' = 1$ (that is, every copy 2 of a nonfinal configuration has a transition to the copy 1 of the successor configuration).

—The regions in R are the Presburger formulas. More precisely, a region is a first-order formula over the free variables pc, C, D, J, where pc is interpreted over the set K of control locations, C and D are interpreted as nonnegative integers, and J is interpreted over the set $\{1, 2\}$. The program counter pc occurs only in atomic subformulas of the form $pc = \ell$, for constants $\ell \in K$; the two counters C and D occurs only in atomic subformulas from $(\mathbb{N}, =, +, 0, 1)$; and the copy designator J occurs only in atomic subformulas of the form $J = 1$ or $J = 2$. The extension of a region σ is the set of states that satisfy the formula σ.

—The only observables in P are the formula $pc = \ell_m$, which we abbreviate to p_{fin}, and its negation. Note that $\ulcorner p_{\mathit{fin}} \urcorner = \Gamma_{\mathit{fin}} \times \{1,2\}$.

We now establish three properties of $\mathcal{S}_M$.

Lemma 5a *$\mathcal{S}_M$ is a symbolic transition system.*

Proof. The Presburger formulas are trivially closed under all boolean operations, and the emptiness (satisfiability) and membership (satisfaction) problems are decidable for the Presburger formulas [Cooper 1972; Berman 1980]. So, it remains to be shown that for every Presburger formula σ, we can construct a Presburger formula $\mathit{Pre}(\sigma)$ such that a state (γ, j) satisfies $\mathit{Pre}(\sigma)$ iff there is a state $(\gamma', j') \in \delta(\gamma, j)$ which satisfies σ. Following the definition of δ, we construct $\mathit{Pre}(\sigma)$ as a disjunction of three parts. The first two parts are simple and left to the reader. The third part is a disjunction over all instructions of M. Suppose that in location ℓ_i, the instruction $C := C + 1$ is executed and leads to location ℓ_{i+1}. Then the corresponding disjunct of $\mathit{Pre}(\sigma)$ is

$$\begin{aligned}(\exists pc', C', D', J'\colon pc = \ell_i \wedge J = 2 \wedge pc' = \ell_{i+1} \wedge C' = C + 1 \wedge D' = D \wedge J' = 1 \\ \wedge\, \sigma[pc, C, D, J := pc', C', D', J']),\end{aligned}$$

where the last conjunct results from σ by replacing the four free variables with their primed versions. □

Lemma 5b *The symbolic transition system $\mathcal{S}_M$ is in the class* STS5.

Proof. We show that the symbolic semi-algorithm Reach terminates for both observations in P. Every state that can reach a state in $\ulcorner p_{\mathit{fin}} \urcorner$ can do so in at most two transitions, because the first copy of every nonfinal configuration has a transition to the first copy of every final configuration. Every state that can reach a state in $\ulcorner \overline{p}_{\mathit{fin}} \urcorner$ can do so in zero transitions, that is, $\ulcorner \mathit{Pre}(\overline{p}_{\mathit{fin}}) \urcorner \subseteq \ulcorner \overline{p}_{\mathit{fin}} \urcorner$, because there are no outgoing transitions from final configurations. □

Lemma 5c *For all configurations γ of M, there exists a halting computation of M from γ iff the symbolic transition system $\mathcal{S}_M$ has a trace from $(\gamma, 1)$ with even length and target p_{fin}.*

Proof. The left-to-right direction follows directly from the construction of $\mathcal{S}_M$. For the right-to-left direction, we reason by induction on the even length k of the trace. Base case ($k = 0$): the configuration γ itself is final, and therefore there is a halting compution (of length 0) from γ. Inductive case ($k > 0$): by the construction of $\mathcal{S}_M$, the configuration γ cannot be final, and a trace from $(\gamma, 1)$ with length k and target p_{fin} must begin with the three states $(\gamma, 1), (\gamma, 2), (M(\gamma), 1)$ followed by an even number of transitions. By the induction hypothesis, there is a halting computation of M from $M(\gamma)$, and therefore also from γ. □

This reduces the halting problem for a two-counter machine M to the problem of checking if the state $(\langle \ell_0, 0, 0 \rangle, 1)$ of the symbolic transition system $\mathcal{S}_M$ satisfies the L_4^μ-formula $(\mu x\colon p_{\mathit{fin}} \vee \exists\bigcirc\, \exists\bigcirc\, x)$.

Theorem 5B* *The L_4^μ and $\overline{L_4^\mu}$ model-checking problems are undecidable for the class* STS5 *of symbolic transition systems.*

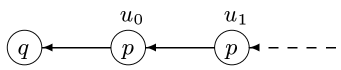

Fig. 12. Reach equivalence is coarser than bounded-reach equivalence.

7.4 Example: Networks of Timed Automata

A *network of timed automata* [Abdulla and Jonsson 1998] consists of a finite-state controller and an arbitrarily large set of identical 1D timed automata. The continuous evolution of the system increases the values of all variables. The discrete transitions of the system are specified by a set of synchronization rules. Formally, a *network of timed automata* is a triple (C, H, Λ), where C is a finite set of controller locations, H is a 1D timed automaton, and Λ is a finite set of rules of the form $r = (\langle c, c' \rangle, e_1, \ldots, e_n)$, where $c, c' \in C$ and $e_1, \ldots, e_n$ are jumps of H. The rule r is enabled if the controller location is c and there are n timed automata $H_1, \ldots, H_n$ whose states are such that the jumps $e_1, \ldots, e_n$, respectively, can be performed. The rule r is executed by simultaneously changing the controller location to c' and the state of each H_i, for $1 \leq i \leq n$, according to the jump e_i. The following result is proved in [Abdulla and Jonsson 1998].

Theorem 5C *The networks of timed automata belong to the class* STS5. *There is a network of timed automata that does not belong to* STS4.

8. GENERAL SYMBOLIC TRANSITION SYSTEMS

For studying reachability questions on symbolic transition systems, it is natural to consider the following fragment of bounded-reachability logic.

Definition: Reachability logic. The *reachability logic* $L_6^\diamond$ consists of the $L_5^\diamond$-formulas that are generated by the grammar

$$\varphi \ ::= \ p \mid \varphi \vee \varphi \mid \exists\Diamond\varphi,$$

for constants $p \in \Pi$. □

The reachability logic $L_6^\diamond$ induces the state equivalence $\cong_6$, which can be defined as follows.

Definition: Reach equivalence. Let $\mathcal{S}$ be a transition system. Two states u and v of $\mathcal{S}$ are *reach equivalent*, denoted $u \cong_6^{\mathcal{S}} v$, if for every trace from u with target p, there is a trace from v with target p, and vice versa. The state equivalence $\cong_6$ is called *reach equivalence*. □

Reach equivalence is coarser than bounded-reach equivalence: in Figure 12, all states u_i, for $i \geq 0$, are reach equivalent, but no two of them are bounded-reach-equivalent. Therefore the reachability logic $L_6^\diamond$ is less expressive than the bounded-reachability logic $L_5^\diamond$. For every symbolic transition system $\mathcal{R}$ with k observables, the reach-equivalence relation $\cong_6^{\mathcal{R}}$ has at most 2^k equivalence classes and, therefore, finite index. Since the reachability problem is undecidable for many kinds of symbolic transition systems (including Turing machines and polyhedral hybrid automata [Alur et al. 1995]), it follows that there cannot be a general algorithm for computing the reach-equivalence quotient of symbolic transition systems.

REFERENCES

Abdulla, P. and Jonsson, B. 1998. Verifying networks of timed automata. In *TACAS 98: Tools and Algorithms for the Construction and Analysis of Systems*. Lecture Notes in Computer Science 1384. Springer-Verlag, 298–312.

Abdulla, P., Čerāns, K., Jonsson, B., and Tsay, Y.-K. 1996. General decidability theorems for infinite-state systems. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, 313–321.

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. 1995. The algorithmic analysis of hybrid systems. *Theoretical Computer Science 138*, 3–34.

Alur, R. and Dill, D. 1994. A theory of timed automata. *Theoretical Computer Science 126*, 183–235.

Alur, R. and Henzinger, T. 1998. *Computer-Aided Verification: An Introduction to Model Building and Model Checking for Concurrent Systems*. Draft.

Alur, R., Henzinger, T., and Ho, P.-H. 1996. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering 22*, 181–201.

Berman, L. 1980. The complexity of logical theories. *Theoretical Computer Science 11*, 71–77.

Bhat, G. and Cleaveland, R. 1996. Efficient model checking via the equational μ-calculus. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, 304–312.

Bouajjani, A., Fernandez, J.-C., and Halbwachs, N. 1990. Minimal model generation. In *CAV 90: Computer-Aided Verification*. Lecture Notes in Computer Science 531. Springer-Verlag, 197–203.

Browne, M., Clarke, E.M., and Grumberg, O. 1988. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science 59*, 115–131.

Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., and Hwang, L.J. 1992. Symbolic model checking: 10^{20} states and beyond. *Information and Computation 98*, 142–170.

Clarke, E.M., Grumberg, O., and Long, D. 1994. Verification tools for finite-state concurrent systems. In *A Decade of Concurrency: Reflections and Perspectives*. Lecture Notes in Computer Science 803. Springer-Verlag, 124–175.

Cleaveland, R., Klein, M., and Steffen, B. 1992. Faster model checking for the modal μ-calculus. In *CAV 92: Computer-Aided Verification*. Lecture Notes in Computer Science 663. Springer-Verlag, 410–422.

Cooper, D.C. 1972. Theorem proving in arithmetic without multiplication. *Machine Intelligence 7*, 91–100.

Cousot, P. and Cousot, R. 1977. Abstract interpretation: A unified lattice model for the static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the Fourth Annual Symposium on Principles of Programming Languages*. ACM Press, 238–252.

Dam, M. 1994. Ctl* and Ectl* as fragments of the modal μ-calculus. *Theoretical Computer Science 126*, 77–96.

Emerson, E. 1990. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. Vol. B. Elsevier Science Publishers, 995–1072.

Emerson, E., Jutla, C., and Sistla, A. 1993. On model checking for fragments of μ-calculus. In *CAV 93: Computer-Aided Verification*. Lecture Notes in Computer Science 697. Springer-Verlag, 385–396.

Emerson, E., Jutla, C., and Sistla, A. 2001. On model checking for the μ-calculus and its fragments. In *Theoretical Computer Science 258*, 491–522.

Emerson, E. and Lei, C. 1986. Efficient model checking in fragments of the propositional μ-calculus. In *Proceedings of the First Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, 267–278.

Finkel, A. and Schnoebelen, P. 1998. *Well-structured Transition Systems Everywhere*. Tech. Rep. LSV-98-4, Laboratoire Spécification et Vérification, ENS de Cachan, France.

HENZINGER, M., HENZINGER, T., AND KOPKE, P. 1995. Computing simulations on finite and infinite graphs. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 453–462.

HENZINGER, T. 1995. Hybrid automata with finite bisimulations. In *ICALP 95: Automata, Languages, and Programming*. Lecture Notes in Computer Science 944. Springer-Verlag, 324–335.

HENZINGER, T. 1996. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, 278–292.

HENZINGER, T., HO, P.-H., AND WONG-TOI, H. 1995. HYTECH: The next generation. In *Proceedings of the 16th Annual Real-time Systems Symposium*. IEEE Computer Society Press, 56–65.

HENZINGER, T. AND KOPKE, P. 1996. State equivalences for rectangular hybrid automata. In *CONCUR 96: Concurrency Theory*. Lecture Notes in Computer Science 1119. Springer-Verlag, 530–545.

HENZINGER, T., KOPKE, P., PURI, A., AND VARAIYA, P. 1998. What's decidable about hybrid automata? *Journal of Computer and System Sciences 57*, 94–124.

HENZINGER, T. AND MAJUMDAR, R. 2000. Symbolic model checking for rectangular hybrid systems. In *TACAS 00: Tools and Algorithms for the Construction and Analysis of Systems*. Lecture Notes in Computer Science 1785. Springer-Verlag, 142–156.

HENZINGER, T., NICOLLIN, X., SIFAKIS, J., AND YOVINE, S. 1994. Symbolic model checking for real-time systems. *Information and Computation 111*, 193–244.

HOPCROFT, J. AND ULLMAN, J. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company.

JANIN, D. AND WALUKIEWICZ, I. 1996. On the expressive completeness of the propositional μ-calculus with respect to the monadic second order logic. In *CONCUR 96: Concurrency Theory*. Lecture Notes in Computer Science 1119. Springer-Verlag, 263–277.

KANELLAKIS, P. AND SMOLKA, S. 1990. CCS expressions, finite-state processes, and three problems of equivalence. *Information and Computation 86*, 43–68.

KOZEN, D. 1983. Results on the propositional μ-calculus. *Theoretical Computer Science 27*, 333–354.

LAMPORT, L. 1974. A new solution of Dijkstra's concurrent programming problem. *Communications of the ACM 17*, 453–455.

LOISEAUX, C., GRAF, S., SIFAKIS, J., BOUAJJANI, A., AND BENSALEM, S. 1995. Property-preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design 6*, 11–44.

MILNER, R. 1971. An algebraic definition of simulation between programs. In *Second International Joint Conference on Artificial Intelligence*. The British Computer Society, 481–489.

THOMAS, W. 1990. Automata on infinite objects. In *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. Vol. B. Elsevier Science Publishers, 133–191.

VAN GLABBEEK, R. 1990. *Comparative Concurrency Semantics and Refinement of Actions*. Ph.D. Thesis, Vrije Universiteit te Amsterdam, The Netherlands.

WOLPER, P. 1983. Temporal logic can be more expressive. *Information and Control 56*, 72–99.

Received January 2001; revised July 2002; accepted June 2003