A Classification of Symbolic Transition Systems*

Thomas A. Henzinger[1] Rupak Majumdar[1] Jean-François Raskin[2]

[1]Department of Electrical Engineering and Computer Sciences
University of California at Berkeley, CA 94720-1770, USA
[2]Département d'Informatique, Faculté des Sciences
Université Libre de Bruxelles, Belgium
{tah,rupak,jfr}@eecs.berkeley.edu

Abstract. We define five increasingly comprehensive classes of infinite-state systems, called STS1–5, 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 (1) iterating the predecessor and boolean operations starting from a finite set of observable state sets, and (2) terminating when no new state sets are 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 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 enables model checking of reachability properties.

0 Introduction

To explore the state space of an infinite-state transition system, it is often convenient to compute on a data type called "region," whose members represent (possibly infinite) sets of states. Regions might be implemented, for example, as constraints on the integers or reals. 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. For model checking, we are particularly interested in boolean operations on regions as well as the predecessor operation, which, given a target region, computes the region of all states with successors in the target region. While a region algebra supports individual operations on regions, the iteration of these operations may generate an infinite number of distinct regions. In this paper, we study restricted classes of symbolic transition systems for which certain forms of iteration, if terminated after a finite number of operations, still yield sufficient information for checking interesting, unbounded temporal properties of the system.

* This research was supported in part by the DARPA (NASA) grant NAG2-1214, the DARPA (Wright-Patterson AFB) grant F33615-C-98-3614, the MARCO grant 98-DT-660, the ARO MURI grant DAAH-04-96-1-0341, the NSF CAREER award CCR-9501708, and the Belgian National Fund for Scientific Research (FNRS).

0.1 Symbolic Transition Systems

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 $\delta : Q \to \mathbf{2}^Q$ 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 $And\colon R \times R \to R$ is computable.
4. For each pair $\sigma, \tau \in R$ of regions, there is a region $Diff(\sigma,\tau) \in R$ such that $\ulcorner Diff(\sigma,\tau)\urcorner = \ulcorner\sigma\urcorner\backslash\ulcorner\tau\urcorner$; 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 $Empty :$ $R \to \mathbb{B}$ 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 $Member:$ $Q \times R \to \mathbb{B}$ 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: 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 $\ulcorner\overline{Diff}(\sigma,\tau)\urcorner = Q\backslash\ulcorner Diff(\tau,\sigma)\urcorner$, and (5) a computable function $\overline{Empty}$ for deciding all universality questions about regions (that is, $\overline{Empty}(\sigma)$ iff $\ulcorner\sigma\urcorner = Q$). All results of this paper have an alternative, dual formulation. □

Remark: Abstract Interpreation The region algebra of a symbolic transition system may be viewed as the collecting semantics (in the sense of abstract interpretation [CC77]) of the concrete semantics of the transition system. In fact, in a symbolic transition system, the semantics is lifted from individual states to sets of states. We refer the interested reader to [CC77] for more details about collecting semantics and abstract interpretation. □

0.2 Example: Polyhedral Hybrid Automata

A *polyhedral hybrid automaton H of dimension m*, for a positive integer m, consists of the following components [AHH96]:

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 $k_0 \sim k_1x_1 + \cdots + k_mx_m$, 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 L^m be the set of linear predicates over 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* [HKPV98] 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* [AD94] 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 the closed predicate $update(e)[X, X' := \mathbf{x}, \mathbf{x}']$ 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 $\varepsilon \in (0, \Delta)$, the closed predicates $inv(v)[X := f(\varepsilon)]$ and $flow(v)[\dot{X} := \dot{f}(\varepsilon)]$ are true. In case (2), the function f is called a *flow function*.
- $R_H = V \times L^m$; 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 values 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 [AHH96].

0.3 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 $u \cong_a^{\mathcal{S}} v$ implies $u \cong_b^{\mathcal{S}} v$ 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 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 $v \in [\![\varphi]\!]_{\mathcal{S}}$. 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 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.

0.4 Preview

In sections 1–5 of this paper, we shall define five increasingly comprehensive classes of symbolic transition systems. In each case $i \in \{1, \ldots, 5\}$, 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 $i \in \{2, \ldots, 5\}$.

2 Algorithmics: Symbolic state-space exploration We give a symbolic semi-algorithm 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 $\mathsf{STS}(i)$. Termination of the semi-algorithm 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 the semi-algorithm 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 semi-algorithm terminates upon having constructed a representation of the quotient system $\mathcal{S}/_{\cong_i}$. The semi-algorithm can therefore be used to decide all $\cong_i$ equivalence questions for the class $\mathsf{STS}(i)$.

3 Verification: Decidable properties We give a state logic L_i which admits abstraction and induces the state equivalence $\cong_i$. Since $\cong_i$ quotients can be constructed effectively, it follows that the L_i model-checking problem for the class $\mathsf{STS}(i)$ is decidable. However, model-checking algorithms which rely on the explicit construction of quotient systems are usually impractical. Hence, we also give a symbolic semi-algorithm that terminates on the symbolic transition systems in the class $\mathsf{STS}(i)$ and directly decides all L_i model-checking questions for this class.

4 Example: Hybrid systems The interesting members of the class $\mathsf{STS}(i)$ are those with infinitely many states. In four out of the five cases, following [Hen96], we provide certain kinds of polyhedral hybrid automata as examples.

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

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

The termination test $\ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner$, 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

1 Class-1 Symbolic Transition Systems

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.

1.1 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 $\mathcal{S}$ 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.* □

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. □

1.2 Symbolic State-space Exploration: Partition Refinement

The bisimilarity relation of a finite-state system can be computed by partition refinement [KS90]. The symbolic semi-algorithm Closure1 of Figure 1 applies this method to infinite-state systems [BFH90,Hen95]. 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 $s \in \ulcorner \sigma \urcorner$ iff $t \in \ulcorner \sigma \urcorner$, 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.*

1.3 Decidable Properties: Branching Time

Definition: μ-calculus The formulas of the *μ-calculus* are generated by the grammar

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

for constants p from some set Π, and variables x from some set X. 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 $\mathcal{E}$ 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:

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

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 $\mathcal{E}$. □

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 μ-calculus are relevant in our context [AH98]. First, L_1^μ admits abstraction, and the state equivalence induced by L_1^μ is $\cong_1$ (bisimilarity). Second, L_1^μ is very expressive; in particular, L_1^μ is more expressive than the temporal logics CTL* and CTL, which also induce bisimilarity. Third, the definition of L_1^μ naturally suggests a model-checking method for finite-state systems, where each fixpoint can be computed by successive approximation. The symbolic semi-algorithm ModelCheck of Figure 2 applies this method 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, 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 $[\![\varphi]\!]_{\mathcal{S}} = \bigcap\{\ulcorner\sigma\urcorner \mid \sigma \in T\}$. □

Counterexample The converse of Theorem 1B does not hold: there exist symbolic transition systems $\mathcal{S}$ such that for every L_1^μ-formula φ, the symbolic semi-algorithm Model Check terminates on the region algebra $\mathcal{R}_{\mathcal{S}}$ and φ, and yet $\mathcal{S}$ is not in STS1. Indeed, the example of Figure 3 shows a symbolic transition system for which ModelCheck terminates for every formula φ of L_1^μ, but iteration of *Pre* does not terminate. In fact, this is true for every transition system whose transition relation is transitive. □

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 $\varphi = (\varphi_1 \vee \varphi_2)$ **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

1.4 Example: Singular Hybrid Automata

The fundamental theorem of timed automata [AD94] shows that for every timed automaton, the (time-abstract) bisimilarity relation has finite index. The proof can be extended to the singular automata [ACH+95]. It follows that the symbolic semi-algorithm ModelCheck, which has been implemented for polyhedral hybrid automata in the tool HyTech [HHWT95], 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 [Hen95].

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

1.5 Example: The 2-Process Bakery Protocol

Consider the 2-process bakery protocol [Lam74] for mutual exclusion presented as a finite collection of guarded commands in Figure 4. As presented, the protocol uses two variables (the "tokens") that range over the natural numbers. The state of the protocol is given by a 4-tuple (pc_1, pc_2, y_1, y_2) denoting the values of the program counters in the two processes, and the values of the tokens y_1 and y_2. The observables are boolean formulae over the values of the program counter. However, we can show that the bisimilarity relation of this transition system has finite index. Indeed, define the relation $\cong$ between states of the protocol as $u \cong v$ iff (1) $u(pc_i) = v(pc_i)$ for $i = 1,2$ (where $u(x)$ denotes the valuation to variable x in state u); (2) $u(y_i) = 0$ iff

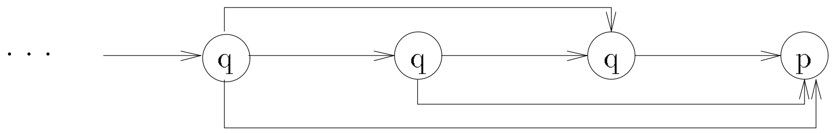

Fig. 3. A symbolic transition system for which ModelCheck terminates for every φ, while closure under *Pre* (and hence Closure1) does not.

$$
\begin{array}{l}
\textbf{var } pc_1, pc_2 \ : \ \{N, W, C\} \\
\textbf{var } y_1, y_2 \ : \ \mathbb{N}
\end{array}
$$

$$
\begin{array}{ll}
[\!]\ pc_1 = N & \to pc_1, y_1 := W, y_2 + 1 \\
[\!]\ pc_1 = W \wedge (y_2 = 0 \vee y_1 \leq y_2) & \to pc_1 := C \\
[\!]\ pc_1 = C & \to pc_1, y_1 := N, 0 \\
 & \\
[\!]\ pc_2 = N & \to pc_2, y_2 := W, y_1 + 1 \\
[\!]\ pc_2 = W \wedge (y_1 = 0 \vee y_2 < y_1) & \to pc_2 := C \\
[\!]\ pc_2 = C & \to pc_2, y_2 := N, 0
\end{array}
$$

Fig. 4. The 2-process bakery mutual exclusion algorithm

$v(y_i) = 0$ for $i = 1, 2$; and (3) $u(y_1) \leq u(y_2)$ iff $v(y_1) \leq v(y_2)$. By a simple case enumeration, it can be seen that $\cong$ is a bisimulation relation on the state space. Moreover, the relation has a finite index (the number of equivalence classes is 72). Thus, the 2-process bakery protocol is in STS1. By Theorem 1A, the closure algorithm Closure1 will terminate on the region algebra of the 2-process bakery mutual exclusion protocol.

2 Class-2 Symbolic Transition Systems

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.

2.1 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 simulations $\preceq_1$, $\preceq_2$ on $\mathcal{S}$ such that $u \preceq_1 v$ and $v \preceq_2 u$. The state equivalence $\cong_2$ is called *similarity*. $\square$

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. $\square$

Since similarity is coarser than bisimilarity [vG90], the class STS2 of symbolic transition systems is a proper extension of STS1.

2.2 Symbolic State-space Exploration: Intersection Refinement

The symbolic semi-algorithm Closure2 of Figure 5 is an abstract version of the method presented in [HHK95] 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

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

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. 5. Intersection refinement

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.*

2.3 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

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

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 negation-free μ-calculus and its dual are relevant in our context [AH98]. First, both L_2^μ and $\overline{L_2^\mu}$ admit abstraction, and the state equivalence induced by both L_2^μ and $\overline{L_2^\mu}$ is $\cong_2$ (similarity). It follows that the logic L_1^μ with negation is more expressive than either L_2^μ or $\overline{L_2^\mu}$. Second, the negation-free logic L_2^μ is more expressive than the existential fragments of CTL* and CTL, which also induce similarity, and the dual logic $\overline{L_2^\mu}$ is more expressive than the universal fragments of CTL* and CTL, which again induce similarity.

If we apply the symbolic semi-algorithm ModelCheck of Figure 2 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.*

2.4 Example: 2D Rectangular Hybrid Automata

For every 2D rectangular automaton, the (time-abstract) similarity relation has finite index [HHK95]. 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 [HK96].

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

3 Class-3 Symbolic Transition Systems

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.

3.1 Finite Characterization: Traces

Definition: Trace equivalence Let $\mathcal{S} = (Q, \delta, \cdot, \ulcorner\cdot\urcorner, P)$ be a transition system. Given a state $u_0 \in Q$, a *source-*u_0 *trace* π of $\mathcal{S}$ is a finite or infinite sequence $p_0p_1\ldots$ of observables $p_i \in P$ such that

1. $u_0 \in \ulcorner p_0 \urcorner$;
2. for all $0 \leq i$, there is a state $u_{i+1} \in (\delta(u_i) \cap \ulcorner p_{i+1} \urcorner)$.

If the trace is a finite sequence $p_0p_1\ldots p_n$, the number n of observables (minus 1) is called the *length* of the trace π, the final state u_n is the *sink* of π, and the final observable p_n is the *target* of π. The length of an infinite trace is infinity. Two states $u, v \in Q$ are *trace equivalent*, denoted $u \cong_3^{\mathcal{S}} v$, if every source-$u$ trace of $\mathcal{S}$ is a source-v trace of $\mathcal{S}$, and vice versa. The state equivalence $\cong_3$ is called *trace equivalence.* Two states $u, v \in Q$ are *finite trace equivalent*, denoted $u \cong_{3f}^{\mathcal{S}} v$, if every finite source-u trace of $\mathcal{S}$ is a source-v trace of $\mathcal{S}$, and vice versa. The state equivalence $\cong_{3f}$ is called *finite 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 [vG90], the class STS3 of symbolic transition systems is a proper extension of STS2.

3.2 Symbolic State-space Exploration: Observation Refinement

Trace equivalence can be characterized operationally by the symbolic semi-algorithm Closure3 of Figure 6. We shall show that, when the input is the region algebra of a symbolic transition system $\mathcal{S} = (Q, \delta, R, \ulcorner\cdot\urcorner, P)$, then Closure3 terminates iff the trace-equivalence relation $\cong_3^{\mathcal{S}}$ has finite index. Furthermore, upon termination, $u \cong_3^{\mathcal{S}} v$ iff for each region $\sigma \in T_i$, we have $u \in \ulcorner\sigma\urcorner$ iff $v \in \ulcorner\sigma\urcorner$.

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.

Proof We proceed in two steps. First, we show that Closure3 terminates on the region algebra $\mathcal{R}_{\mathcal{S}}$ iff the equivalence relation $\cong_{L_3^\mu}^{\mathcal{S}}$ induced by the deterministic μ-calculus (defined below) has finite index. Second, we show that $\cong_{L_3^\mu}$ coincides with trace equivalence. The proof of the first part proceeds as usual. It can be seen by induction that for all $i \geq 0$, the extension of every region in T_i, as computed by Closure3, is a $\cong_{L_3^\mu}^{\mathcal{S}}$ block. Thus, if $\cong_{L_3^\mu}^{\mathcal{S}}$ has finite index, then Closure3 terminates. Conversely, suppose that Closure3 terminates with $\ulcorner T_{i+1} \urcorner \subseteq \ulcorner T_i \urcorner$. It can be shown that if two states are not $\cong_{L_3^\mu}^{\mathcal{S}}$-equivalent, then there is a region in T_i

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. 6. Observation refinement

which contains one state but not the other. 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 $u \cong^{\mathcal{S}}_{L_3^\mu} v$. This implies that $\cong^{\mathcal{S}}_{L_3^\mu}$ has finite index.

For the second part, we show that L_3^μ is as expressive as the logic $\exists$BÜCHI, whose formulas are the existentially interpreted Büchi automata, and $\exists$BÜCHI is as expressive as L_3^μ. This result is implicit in a proof by [EJS93]. We recall a few definitions. A *Büchi automaton* BÜCHI is a tuple $(S, \Phi, \rightarrow, s_0, F)$, where S is a finite set of states, Φ is a finite input alphabet, $\rightarrow \subseteq S \times \Phi \times S$ is the transition relation, $s_0 \in S$ is the start state, and $F \subseteq S$ is the set of Büchi accepting states. An *execution* of BÜCHI on an ω-word $w = w_0 w_1 \ldots \in \Phi^\omega$ is an infinite sequence $r = s_0 s_1 \ldots$ of states in S, starting from the initial state s_0, such that $s_i \xrightarrow{w_i} s_{i+1}$ for all $i \geq 0$. The execution r is *accepting* if some state in F occurs infinitely often in r. The automaton BÜCHI *accepts* the word w if it has an accepting execution on w. The *language* $L(\text{BÜCHI}) \subseteq \Phi^\omega$ is the set of ω-words accepted by BÜCHI.

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 B_φ such that for all transition systems $\mathcal{S}$, a state u of $\mathcal{S}$ satisfies φ iff for some infinite source-u trace of $\mathcal{S}$ is accepted by B_φ. Conversely, given a Büchi automaton B, we construct an L_3^μ-formula which is equivalent to $\exists B$. Let BÜCHI be a Büchi automaton. For notational convenience, we present the formula in equational form [CKS93]; it can be easily converted to the standard representation by unrolling the equations, and binding variables with μ or ν-fixpoints. For each set $R \in 2^P$, let ψ_R abbreviate the formula $\bigwedge R \wedge \bigwedge\{\overline{p} \mid p \in P \backslash R\}$. For each state s of BÜCHI, we introduce a propositional variable X_s. The equation for X_s is

$$X_s \;=_\lambda\; \bigvee\{\psi_R \wedge \exists\bigcirc X_{s'} \mid s \xrightarrow{R} s'\},$$

where $\lambda = \nu$ if $s \in F$ is an accepting state, and $\lambda = \mu$ otherwise. The top-level variable is X_{s_0}, where s_0 is the initial state. The correctness of the procedure follows from [BC96]. An equivalent construction is given in [Dam94].

Since the state equivalence induced by $\exists$BÜCHI is trace equivalence, it follows that $\cong_{L_3^\mu}$ is also trace equivalence. □

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

3.3 Decidable Properties: Linear Time

Definition: Deterministic μ-calculus The *deterministic μ-calculus* (also called "L_1" in [EJS93]) consists of the μ-calculus formulas that are generated by the grammar

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

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. □

The following facts about the deterministic μ-calculus and its dual are relevant in our context (cf. the second part of the proof of Theorem 3A). First, both L_3^μ and $\overline{L_3^\mu}$ admit abstraction, and the state equivalence induced by both L_3^μ and $\overline{L_3^\mu}$ is $\cong_3$ (trace equivalence). It follows that the logic L_2^μ with unrestricted conjunction is more expressive than L_3^μ, and $\overline{L_2^\mu}$ is more expressive than $\overline{L_3^\mu}$. Second, the logic L_3^μ with restricted conjunction is more expressive than the existential interpretation of the linear temporal logic LTL, which also induces trace equivalence. 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 $(\mu x : q \vee (p \wedge \exists\bigcirc x))$ (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 induces trace equivalence. For example, the (universal) LTL formula $p\mathcal{W}q$ ("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).

If we apply the symbolic semi-algorithm ModelCheck of Figure 2 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 These results suggest, in particular, a symbolic procedure for model checking LTL properties over STS3 systems [HM00]. Suppose that $\mathcal{S}$ is a symbolic transition system in the class STS3, and φ is an LTL formula. First, convert $\neg\varphi$ to a Büchi automaton $\text{BÜCHI}_{\neg\varphi}$ using a tableau construction, and then to an equivalent L_3^μ-formula ψ (introduce one variable per state of $\text{BÜCHI}_{\neg\varphi}$). Second, run the symbolic semi-algorithm ModelCheck on inputs $\mathcal{R}_\mathcal{S}$ and ψ. It will terminate with a representation of the complement of the set of states that satisfy φ in $\mathcal{S}$.

While ModelCheck provides a symbolic semi-algorithm for LTL, traditionally, a different method is used for symbolic model checking of LTL formulas [CGL94]. Given a state u of a finite-state transition structure $\mathcal{S}$, and an LTL formula φ, the model-checking question for LTL can be solved by constructing the product of $\mathcal{S}$ with the tableau automaton BÜCHI_φ, and then checking the nonemptiness of a Büchi condition on the product structure. A Büchi condition is an LTL formula of the form $\Box\Diamond\psi$, where ψ is a disjunction of observables; therefore nonemptiness can be checked symbolically by evaluating the equivalent formula

$$\chi \;=\; \nu X_1.\, \mu X_2.\, (\exists\bigcirc X_2 \vee (\psi \wedge \exists\bigcirc X_1))$$

of L_3^μ.

To extend this method to infinite-state structures, we need to be more formal. Let $\mathcal{S} = (Q, \delta, R, \ulcorner\cdot\urcorner, P)$ be a symbolic transition system and let $\text{BÜCHI}_\varphi = (S, 2^\Pi, \rightarrow, s_0, F)$ be a tableau automaton. The *product structure* $\mathcal{S}_\varphi = (S \times Q, \delta_\varphi, S \times R, \ulcorner\cdot\urcorner_\varphi, P_\varphi)$ is defined as follows. The set of states of $\mathcal{S}_\varphi$ is the Cartesian product $S \times Q$, and the set of regions of $\mathcal{S}_\varphi$ is the Cartesian product $S \times R$. The extension $\ulcorner(s\sigma)\urcorner_\varphi$ for the region (s, σ) is the set of states $\{s\} \times \ulcorner\sigma\urcorner$. The set of observables P_φ is $S \times P$, for an observable $(s, p) \in P_\varphi$, define $(s', u) \in \ulcorner(s, p)\urcorner_\varphi$ iff $s' = s$ and $u \in \ulcorner p\urcorner$; that is, the state of the tableau automaton is also observable. Define $(s', v) \in \delta_\varphi(s, u)$ iff $s \xrightarrow{p} s'$ and $v \in \delta(u)$ and $u \in \ulcorner p\urcorner$. Then $u \in [\![\varphi]\!]_\mathcal{S}$, for $u \in Q$, iff $(s_0, u) \in [\![\Box\Diamond\psi]\!]_{\mathcal{S}_\varphi}$, where $\psi = \bigvee_{s\in F, p\in P}(s, p)$. Since the tableau automaton BÜCHI_φ is finite, it is easy to check that $\mathcal{R}_\varphi$, with the extension function $\ulcorner\cdot\urcorner_\varphi$, is a region algebra for $\mathcal{S}_\varphi$. Let AutomLTL be the *product-automaton based algorithm for* LTL *model checking* which, given an LTL formula φ and a symbolic transition system $\mathcal{S}$, evaluates the L_3^μ formula χ (representing a Büchi condition) on the product system $\mathcal{S}_{\neg\varphi}$ (using the semi-algorithm ModelCheck). It is not difficult to see that if observation refinement terminates on $\mathcal{S}$ in k steps, then it also terminates on $\mathcal{S}_\varphi$ in k steps (if Closure3 generates m regions on $\mathcal{S}$, then it generates at most $m \cdot |S|$ regions on $\mathcal{S}_\varphi$).

Corollary 3B′ *For all symbolic transition systems $\mathcal{S}$ in* STS3, *and every* LTL *formula φ, the symbolic semi-algorithm* AutomLTL *terminates on the region algebra $\mathcal{R}_\mathcal{S}$ and the input formula φ.*

Indeed, by induction on the construction of regions, one can show that for each region representative (s, σ) computed in the product-automaton based algorithm, the variable X_s in the μ-calculus based algorithm represents the region $\ulcorner\sigma\urcorner$ at some stage of the computation, and conversely, for each valuation R of the variable X_s in the μ-calculus based algorithm, a region representative of $\{s\} \times R$ is computed in the product-automaton based algorithm. Thus, the two methods are equivalent in the regions they generate. $\square$

Remark: Finite Trace Equivalence Let STS3f be the class of symbolic transition systems whose finite trace equivalence relation has finite index.

Definition: Finitary Deterministic μ-calculus The *finitary fragment* of the deterministic μ-calculus consists of the formulas of the deterministic μ-calculus without the greatest fixpoint operator. Formally, formulas are generated by the grammar

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

for constants $p \in \Pi$ and variables $x \in X$. The state logic L_{3f}^{μ} consists of the closed formulas of the finitary deterministic μ-calculus. The state logic $\overline{L_{3f}^{\mu}}$ consists of the duals of all L_{3f}^{μ}-formulas. $\square$

From the proof of Theorem 3A, we notice that the finitary deterministic μ-calculus is equally expressive as the logic $\exists$A whose formulas are the existentially interpreted finite automata, in other words, L_{3f}^{μ} expresses exactly the regular sets. Thus the following corollary is immediate.

Corollary 3BFinite *For all symbolic transition systems $\mathcal{S}$ in* STS3f *and every L_{3f}^{μ}-formula φ, the symbolic semi-algorithm* ModelCheck *terminates on the region algebra $\mathcal{R}_{\mathcal{S}}$ and the input formula φ. Hence, the L_{3f}^{μ} and $\overline{L_{3}^{\mu}}$ model-checking problems are decidable for the class* STS3f *of symbolic transition systems.*

3.4 Example: Rectangular Hybrid Automata

For every rectangular automaton, the (time-abstract) trace-equivalence relation has finite index [HKPV98]. 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 [HKPV98].

Theorem 3C *The rectangular automata belong to the class* STS3*.*

4 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. 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.

4.1 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 source-$u$ trace of $\mathcal{S}$ with length n and target p, there is a source-v trace of $\mathcal{S}$ 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 (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.

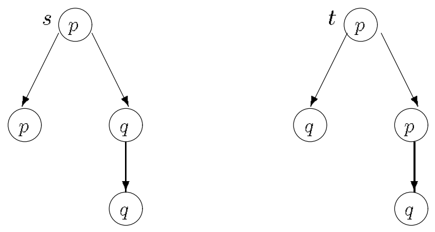

Fig. 7. Distance equivalence is coarser than trace equivalence

> **Symbolic semi-algorithm** Closure4
> Input: a region algebra $\mathcal{R} = (P, Pre, \cdot, Diff, Empty)$.
>
> $T_0 := P$;
> **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

4.2 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 $u, v \in Q$, define $u \sim_i^\mathcal{S} v$ if for every source-u trace of $\mathcal{S}$ with length $n \leq i$ and target p, there is a source-v trace of $\mathcal{S}$ 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 $\cong_2^\mathcal{S}$ is equal to $\bigcap\{\sim_i^\mathcal{S} \mid i \geq 0\}$, if $\cong_2^\mathcal{S}$ has finite index, then $\cong_2^\mathcal{S}$ is equal to $\sim_i^\mathcal{S}$ for some $i \geq 0$. Then, Closure2 will terminate in i 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.*

4.3 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

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

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. □

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. 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:

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

(The constructor $\exists\Diamond_{\le d}$ 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, $\vee$, $\exists\bigcirc$, $\exists\Diamond_{\le d}$, and $\exists\Diamond$ by $\overline{p}$, $\wedge$, $\forall\bigcirc$, $\forall\Box_{\le d}$, 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. □

The following facts about the conjunction-free μ-calculus, conjunction-free temporal logic, and their duals are relevant in our context. First, both L_4^μ and $\overline{L_4^\mu}$ admit abstraction, and the state equivalence induced by both L_4^μ and $\overline{L_4^\mu}$ is $\cong_4$ (distance equivalence). It follows that the logic L_3^μ with restricted conjunction 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 can be reached in an even number of transitions can be expressed in L_4^μ but not in $L_4^\diamond$.

If we apply the symbolic semi-algorithm ModelCheck of Figure 2 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.*

5 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. 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" [AČJT96]. Infinite-state examples of class-5 systems are provided by networks of rectangular hybrid automata.

5.1 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 source-u trace of $\mathcal{S}$ with length n and target p, there is a source-v trace of $\mathcal{S}$ with length at most n and target p, and vice versa. The state equivalence $\cong_5$ is called *bounded-reach equivalence*. □

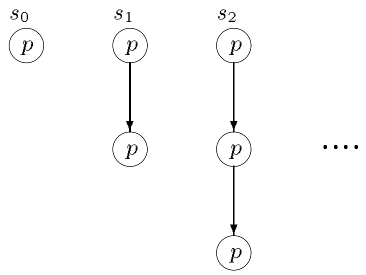

Fig. 9. Bounded-reach equivalence is coarser than distance equivalence

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

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 (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.

5.2 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, when 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 observation $p \in P$ 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 [AČJT96,FS98]. 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 observations p and all nonnegative integers d, the set $[\![\exists\Diamond_{\leq d}\, p]\!]_{\mathcal{S}}$ is upward-closed.

Proof ($2 \Rightarrow 1$) Define $u \sim^{\mathcal{S}}_{\leq n} v$ if for all observations p, for every source-u trace with length n and target p, there is a source-v trace with length at most n and target p, and vice versa. Note that $\sim^{\mathcal{S}}_{\leq n}$ has finite index for all $n \geq 0$. Suppose that the semi-algorithm Reach terminates in at most i iterations for each observation p. 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 and all $n \geq 0$, for every source-u trace with length n and target p, there is a source-v trace with length at most n and target p. Then each set $[\![\exists\Diamond_{\leq d}\, p]\!]_{\mathcal{S}}$, for an observable p and a nonnegative integer d, 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 [AČJT96]. □

5.3 Decidable Properties: Bounded Reachability

Definition: Bounded-reachability logic The *bounded-reachability logic* $L^{\Diamond}_5$ consists of the $L^{\Diamond}_4$-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 $\overline{L^{\Diamond}_5}$ consists of the duals of all $L^{\Diamond}_5$-formulas. □

The following facts about bounded-reachability logic and its dual are relevant in our context. Both $L^{\Diamond}_5$ and $\overline{L^{\Diamond}_5}$ admit abstraction, and the state equivalence induced by both $L^{\Diamond}_5$ and $\overline{L^{\Diamond}_5}$ is $\cong_5$ (bounded-reach equivalence). It follows that the conjunction-free temporal logic $L^{\Diamond}_4$ is more expressive than $L^{\Diamond}_5$, and $\overline{L^{\Diamond}_4}$ is more expressive than $\overline{L^{\Diamond}_5}$. For example, the property that an observable can be reached in exactly d transitions can be expressed in $L^{\Diamond}_4$ but not in $L^{\Diamond}_5$. Since $L^{\Diamond}_5$ admits abstraction, and for STS5 systems the induced quotient can be constructed using the symbolic semi-algorithm Reach, we have the following theorem.

Theorem 5B *The $L^{\Diamond}_5$ and $\overline{L^{\Diamond}_5}$ model-checking problems are decidable for the class* STS5 *of symbolic transition systems.*

A direct symbolic model-checking semi-algorithm for $L^{\Diamond}_5$ and, indeed, $L^{\Diamond}_4$ is easily derived from the semi-algorithm Reach. Then, if Reach terminates, so does model checking for all $L^{\Diamond}_4$-formulas, including unbounded $\exists\Diamond$ properties. The extension to $L^{\Diamond}_4$ is possible, because $\exists\bigcirc$ properties pose no threat to termination. However this is not true for L^{μ}_4: Figure 5.3 shows a symbolic transition system in the class STS5 for which the naive evaluation of the formula $(\mu x : p \vee \exists\bigcirc\,\exists\bigcirc\, x)$ does not terminate. We now show that this is not suprising as L^{μ}_4 is undecidable on STS5 systems. To establish this result, we proceed as follows: given a two-counter machine $M = \langle\{b_1, \cdots, b_m\}, C, D\rangle$, we define a symbolic transition system $\mathcal{S}_M$ that belongs to the class STS5 and that encodes the computations of M using Pre^2. On such a structure we prove that the formula $(\mu x : \mathsf{Final} \vee \exists\bigcirc\,\exists\bigcirc\, x)$ characterizes exactly the set of configurations of the two-counter machine that can reach a final location. This will establish the undecidability of L^{μ}_4 on STS5 systems.

Without lost of generality, we make the following hypothesis on the two-counter machine M: there is only one initial location and only one final location in M, we denote them b_0 and b_m respectively. Furthermore, the initial location of M is never reached after the first instruction. A *configuration* of M is a triple $\gamma = \langle i, c, d\rangle$, where i is the program counter indicating the current instruction, and c and d are the values of the counters C and D. A *computation* of M is a finite or infinite sequence $\sigma = \gamma_0\gamma_1\ldots$ of configurations such that for every γ_{i+1} is a M-successor of γ_i. In the sequel, we write $(\gamma_i, \gamma_{i+1}) \in R_M$ to denote that γ_{i+1} is a M-successor of γ_i. We say that a computation σ is *initial* if $\gamma_0 = \langle 0, 0, 0\rangle$, that is the first instruction is the initial instruction and

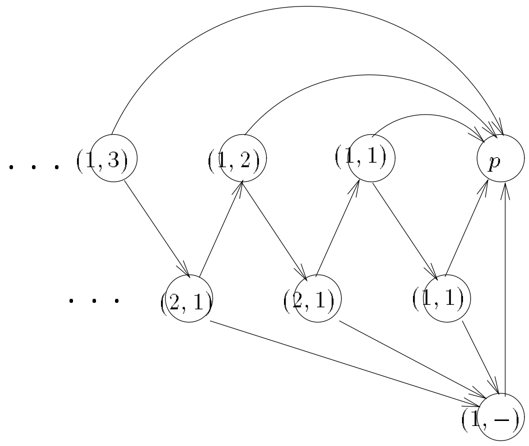

Fig. 11. An STS5 system on which L_4^μ does not terminate

the two counters have the value 0. We say that a computation σ is *final* if σ is finite and its last configuration contains the stop instruction. The *halting problem* for a two-counter machine M is to decide whether or not the execution of M has at least one initial computation that ends in a stop instruction. The problem of deciding if a two-counter machine has a halting computation is undecidable [HU79].

We define the transition system $\mathcal{S}_M$ that encodes the computations of M using Pre^2 as follows.

- The states of the transition system are pairs (γ, i) where γ is a configuration of M and $i \in \{1, 2\}$. We call $(\gamma, 1)$ the copy-1 of configuration γ, and $(\gamma, 2)$ the copy-2 of configuration γ. Formally the set of states Q is the union $I \cup B \cup F$, where : (i) $I = \{(\langle 0, 0, 0\rangle, 1)\}$, that is, the singleton containing the copy-1 of the initial configuration of M; (ii) $B = \{(\langle 0, 0, 0\rangle, 2)\} \cup \{(\langle b, c, d\rangle, i) \mid b \neq 0 \wedge b \neq m \wedge c, d \geq 0\}$, that is, the set containing the copy-2 of the initial configuration of M and two copies of each configuration of M which is not initial and not final; (iii) $F = \{(\langle b, c, d\rangle, 1) \mid b = m \wedge c, d \geq 0\}$, that is the copy-1 of each final configurations of M.
- The transition relation δ is defined as follows: for every $(\gamma_1, i_1), (\gamma_2, i_2) \in Q$, we have that $(\gamma_2, i_2) \in \delta(\gamma_1, i_1)$ if and only if one of the following conditions is satisfied : (i) $(\gamma_1, i_1) \in I \cup B \wedge i_1 = 1 \wedge (\gamma_2, i_2) \in F$, that is every copy-1 of a configuration which is not final is linked to every final configuration; (ii) $\gamma_1 = \gamma_2 \wedge i_1 = 1 \wedge i_2 = 2 \wedge (\gamma_1, i_1) \in B \cup I$, that is every copy-1 of a configuration γ is linked to the copy-2 of γ; (iii) $(\gamma_1, i_1) \in B \wedge i_1 = 2 \wedge (\gamma_2, i_2) \in B \cup F \wedge i_2 = 1 \wedge (\gamma_1, \gamma_2) \in R_M$, that is the copy-2 of a configuration γ_1 is linked to the copy-1 of a configuration γ_2 if γ_2 is a M-successor of γ_1.
- The set of regions R is the set of sets of states definable by Presburger formulas.
- The set of propositions P is $\{\mathsf{Init}, \mathsf{Between}, \mathsf{Final}\}$, with the following extension function: (i) $\ulcorner \mathsf{Init} \urcorner = I$, (ii) $\ulcorner \mathsf{Between} \urcorner = B$, and (iii) $\ulcorner \mathsf{Final} \urcorner = F$.

We now establish three properties of the symbolic transition system $\mathcal{S}_M$.

Lemma 5A *Presburger formulas form a region algebra for the transition system $\mathcal{S}_M$.*

Proof. This algebra is trivially closed under all boolean operations, furthermore the problems of satisfiability and of membership for Presburger formulas are decidable. So, it remains us to show that the set of states satisfying the propositions are expressible as Presburger formula and for all regions R, $Pre(R)$ is expressible by a Presburger formula. Let us consider the proposition $\mathsf{Between}$, the set of states of $\mathcal{S}_M$ that satisfy $\mathsf{Between}$ is expressed by the following Presburger formula: $0 < i < m \wedge ((c \geq 0 \wedge d \geq 0 \wedge (copy = 1 \vee copy = 2)) \vee (i = 0 \wedge copy = 2 \wedge c = 0 \wedge d = 0))$. The other propositions are left to the reader. Let us now show that the region algebra is closed under Pre. We show how to construct the formula Φ that represent $Pre(R_\Psi)$, where R_Ψ is the set of states defined by the Presburger formula Ψ with free variable $i', c', d', copy'$. By definition of δ, we have to consider three cases. We treat the third one, the two first are trivial and left to the reader. The final formula is obtained by taking the disjunction of the three formulas. To construct the formula for the third case, we proceed as follows. For each instruction j of the two-counter machine, we construct a Presburger formula. We treat the case where the instruction j is of the form $i_1 : c := c+1 \to i_2$. The corresponding Presburger formula is: $\exists i', c', d', copy' \cdot \Psi(i', c', d', copy') \wedge i = i_1 \wedge i = i_2 \wedge c = c' - 1 \wedge d = d' \wedge copy = 2 \wedge copy' = 1$. □

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

Proof. We show that for every proposition $p \in P$, the iteration of Pre terminates:

- $p \equiv$ Init. Trivially, $Pre^*(\ulcorner \mathsf{Init} \urcorner) = I$ as $\ulcorner \mathsf{Init} \urcorner = I = \{(\langle 0,0,0\rangle, 1)\}$ and $(\langle 0,0,0\rangle, 1)$ has no predecessors by definition of δ;
- $p \equiv$ Between. We have $Pre^*(\ulcorner \mathsf{Between} \urcorner) = I \cup B = Pre^{\leq 1}(B)$, in fact the copy-1 of the initial configuration of M is reached after one iteration, no other states can be added (the states of F has no outgoing edges).
- $p \equiv$ Final. We have $Pre^*(\ulcorner \mathsf{Final} \urcorner) = Pre^{\leq 2}(\ulcorner \mathsf{Final} \urcorner)$, in fact $\ulcorner \mathsf{Final} \urcorner = F$, $Pre(F) = I \cup \{(\gamma_1, i_1) \in B \mid i_1 = 1 \vee (\exists (\gamma_2, 1) \in F \wedge (\gamma_1, \gamma_2) \in R_M)\}$, and $Pre(Pre(F)) \supseteq \{(\gamma_1, 2) \mid \exists (\gamma_2, 1) \in B \wedge (\gamma_2, 1) \in \delta((\gamma_1, 2))\}$, thus $Pre^{\leq 2}(\ulcorner \mathsf{Final} \urcorner)$ contains every states of Q that is either final or has at least one outgoing edge, and thus no other state can be added.

□

Lemma 5C *For every $(\gamma_a, 1) \in Q$, $(\gamma_a, 1) \in (Pre^2)^n(F)$ if and only if there exists a computation $\sigma = \gamma_0\gamma_1 \ldots \gamma_n$ of M such that $\gamma_a = \gamma_0$ and γ_n is a final configuration.*

Proof. Let us first establish the left to right direction. We reason by induction on i. Base case: $i = 0$. As $(Pre^2)^0(F) = F$, this is trivial. Induction case: $i = k > 0$. Let us consider $(\gamma_a, 1) \in (Pre^2)((Pre^2)^{k-1}(F))$. By construction of $\mathcal{S}_M$, we know that $B \cap \delta(\gamma_a, 1) = \{(\gamma_a, 2)\}$ and $\delta(\gamma_a, 2) = \{(\gamma_c, 1) \mid (\gamma_a, \gamma_c) \in R_M\}$. By hypothesis, $\delta(\gamma_a, 2) \cap (Pre^2)^{k-1}(F)$ is non-empty. Consider $(\gamma_c, 1) \in \delta(\gamma_a, 2) \cap (Pre^2)^{k-1}(F)$, by induction hypothesis, there exists a final M-computation $\sigma' = \gamma_0\gamma_1 \ldots \gamma_{k-1}$. We construct $\sigma = \gamma_a \cdot \sigma'$ which is a final M-computation that goes from γ_a to a final configuration of M.

Let us now establish the right to left implication. We show that if $\sigma = \gamma_0\gamma_1 \ldots \gamma_n$ is a final M-computation then $(\gamma_1, 1) \in (Pre^2)^n(F)$. We reason by induction on the value of n. Base case : $n = 0$, that is $\sigma = \gamma_0$. In this case, γ_0 is a final configuration and $(\gamma_0, 1) \in F$ and trivially, $(\gamma_0, 1) \in (Pre^2)^0(F)$. Induction case: $n = k > 0$. Let us consider the final M-computation $\sigma = \gamma_0\gamma_1 \ldots \gamma_k$. By definition $\gamma_1 \ldots \gamma_k$ is a final M-computation and by induction hypothesis $(\gamma_1, 1) \in (Pre^2)^{k-1}(F)$. Let us show that $(\gamma_0, 1) \in Pre^2(\{(\gamma_1, 1)\})$ holds. We know that $(\gamma_0, \gamma_1) \in R_M$ as σ is a M-computation, and by definition of δ, we have $(\gamma_1, 1) \in \delta(\gamma_0, 2)$, and as $(\gamma_0, 2) \in \delta(\gamma_0, 1)$, we have $(\gamma_0, 1) \in Pre^2(\{(\gamma_1, 1)\})$. It follows that $(\gamma_0, 1) \in (Pre^2)^k(F)$. □

From the above lemmas, it follows that the L_4^μ formula $(\mu x : \mathsf{Final} \vee \exists\bigcirc \exists\bigcirc x)$ expresses on $\mathcal{S}_M$ exactly the set of configurations of M that can reach a final location of M. The undecidability of model-checking L_4^μ on the class STS5 follows as a consequence.

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

5.4 Example: Networks of Rectangular Hybrid Automata

A *network of timed automata* [AJ98] 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. We generalize the definition to rectangular automata. Formally, a *network of rectangular automata* is a triple (C, H, R), where C is a finite set of controller locations, H is a 1D rectangular automaton, and R 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 state is c and there are n rectangular 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 state 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 [AJ98] for networks of timed automata. The proof can be extended to rectangular automata using the observation that every rectangular automaton is simulated by an appropriate timed automaton [HKPV98].

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

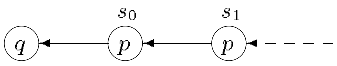

Fig. 12. Reach equivalence is coarser than bounded-reach equivalence

6 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 $p \in \Pi$. $\square$

The reachability logic $L_6^\diamond$ is less expressive than the bounded-reachability logic $L_5^\diamond$, because it induces the following state equivalence, $\cong_6$, which is coarser than bounded-reach equivalence (see Figure 12: all states u_i, for $i \geq 0$, are reach equivalent, but no two of them are bounded-reach-equivalent).

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 source-$u$ trace of $\mathcal{S}$ with target p, there is a source-v trace of $\mathcal{S}$ with target p, and vice versa. The state equivalence $\cong_6$ is called *reach equivalence*. $\square$

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 [ACH+95]), it follows that there cannot be a general algorithm for computing the reach-equivalence quotient of symbolic transition systems.

References

ACH+95. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

AČJT96. P. A. Abdulla, K. Čerāns, B. Jonsson, and Yih-Kuan Tsay. General decidability theorems for infinite-state systems. In *Proceedings of the Eleventh Annual Symposium on Logic in Computer Science*, pages 313–321. IEEE Computer Society Press, 1996.

AD94. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

AH98. R. Alur and T.A. Henzinger. *Computer-Aided Verification. An Introduction to Model Building and Model Checking for Concurrent Systems.* Draft, 1998.

AHH96. R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

AJ98. P. Abdulla and B. Jonsson. Verifying networks of timed automata. In *Proceedings of the International Conference on Tools and Algorithms for Construction and Analysis of Systems*, Lecture Notes in Computer Science 1384, pages 298–312. Springer-Verlag, 1998.

BC96. G. Bhat and R. Cleaveland. Efficient model checking via the equational μ-calculus. In *Proceedings of the Eleventh Annual Symposium on Logic in Computer Science*, pages 304–312. IEEE Computer Society Press, 1996.

BFH90. A. Bouajjani, J.-C. Fernandez, and N. Halbwachs. Minimal model generation. In *CAV 90: Computer-aided Verification*, Lecture Notes in Computer Science 531, pages 197–203. Springer-Verlag, 1990.

CC77. P. Cousot and R. Cousot. 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, 1977.

CGL94. E.M. Clarke, O. Grumberg, and D.E. Long. Verification tools for finite-state concurrent systems. In *A Decade of Concurrency: Reflections and Perspectives*, Lecture Notes in Computer Science 803. Springer-Verlag, 1994.

CKS93. R. Cleaveland, M. Klein, and B. Steffen. Faster model checking for the modal μ-calculus. In G.v. Bochmann and D. Probst, editors, *CAV 92: Computer-aided Verification*, Lecture Notes in Computer Science 663. Springer-Verlag, 1993.

Dam94. M. Dam. CTL* and ECTL* as fragments of the modal μ-calculus. *Theoretical Computer Science*, 126:77–96, 1994.

EJS93. E.A. Emerson, C.S. Jutla, and A.P. Sistla. On model checking for fragments of μ-calculus. In *CAV 93: Computer-aided Verification*, Lecture Notes in Computer Science 697, pages 385–396. Springer-Verlag, 1993.

FS98. A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere. Technical Report LSV-98-4, Laboratoire Spécification et Vérification, ENS de Cachan, Cedex, 1998.

Hen95. T.A. Henzinger. Hybrid automata with finite bisimulations. In *ICALP 95: Automata, Languages, and Programming*, Lecture Notes in Computer Science 944, pages 324–335. Springer-Verlag, 1995.

Hen96. T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.

HHK95. M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *Proceedings of the 36rd Annual Symposium on Foundations of Computer Science*, pages 453–462. IEEE Computer Society Press, 1995.

HHWT95. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: the next generation. In *Proceedings of the 16th Annual Real-time Systems Symposium*, pages 56–65. IEEE Computer Society Press, 1995.

HK96. T.A. Henzinger and P.W. Kopke. State equivalences for rectangular hybrid automata. In *CONCUR 96: Concurrency Theory*, Lecture Notes in Computer Science 1119, pages 530–545. Springer-Verlag, 1996.

HKPV98. T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.

HM00. T.A. Henzinger and R. Majumdar. Symbolic model checking for rectangular hybrid systems. In S. Graf and M. Schwarzbach, editors, *TACAS '00: Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science 1785, pages 142–156. Springer-Verlag, 2000.

HU79. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publishing Company, 1979.

KS90. P.C. Kanellakis and S.A. Smolka. CCS expressions, finite-state processes, and three problems of equivalence. *Information and Computation*, 86:43–68, 1990.

Lam74. L. Lamport. A new solution of dijkstra's concurrent programming problem. *Communications of the ACM*, 17:453–455, 1974.

vG90. R.J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions.* PhD thesis, Vrije Universiteit te Amsterdam, The Netherlands, 1990.