Algorithmic analysis for linear hybrid systems

Prof. Dr. Erika Ábrahám

Informatik 2 - Theory of Hybrid Systems
RWTH Aachen

SS09

Literature

Alur et al.: The algorithmic analysis of hybrid systems

Theoretical Computer Science, 138(1):3–34, 1995

Linear hybrid automata

- A *linear term* over the set Var of variables is a linear combination of variables in Var with integer (rational) coefficients.
- A *linear formula* over Var is a Boolean combination of (in)equalities between linear terms over Var.
- A hybrid system is *time-deterministic* iff for every location $l \in Loc$ and every valuation $\nu \in V$ there is at most one activity $f \in Act(l)$ with $f(0) = \nu$. The activity f, then, is denoted by $\varphi_l[\nu]$, its component for $x \in Var$ by $\varphi_l^x[\nu]$.

Linear hybrid automata

Linear hybrid automata are time-deterministic hybrid automata whose definitions contain linear terms, only.

- Activities $Act(l)$ are given as sets of differential equations $\dot{x} = k_x$, one for each variable $x \in Var$, with k_x an integer (rational) constant:

$$\varphi_l^x[\nu](t) = \nu(x) + k_x \cdot t.$$

- Invariants $Inv(l)$ are defined by linear formulae ψ over Var:

$$\nu \in Inv(l) \quad \textit{iff} \quad \nu(\psi)$$

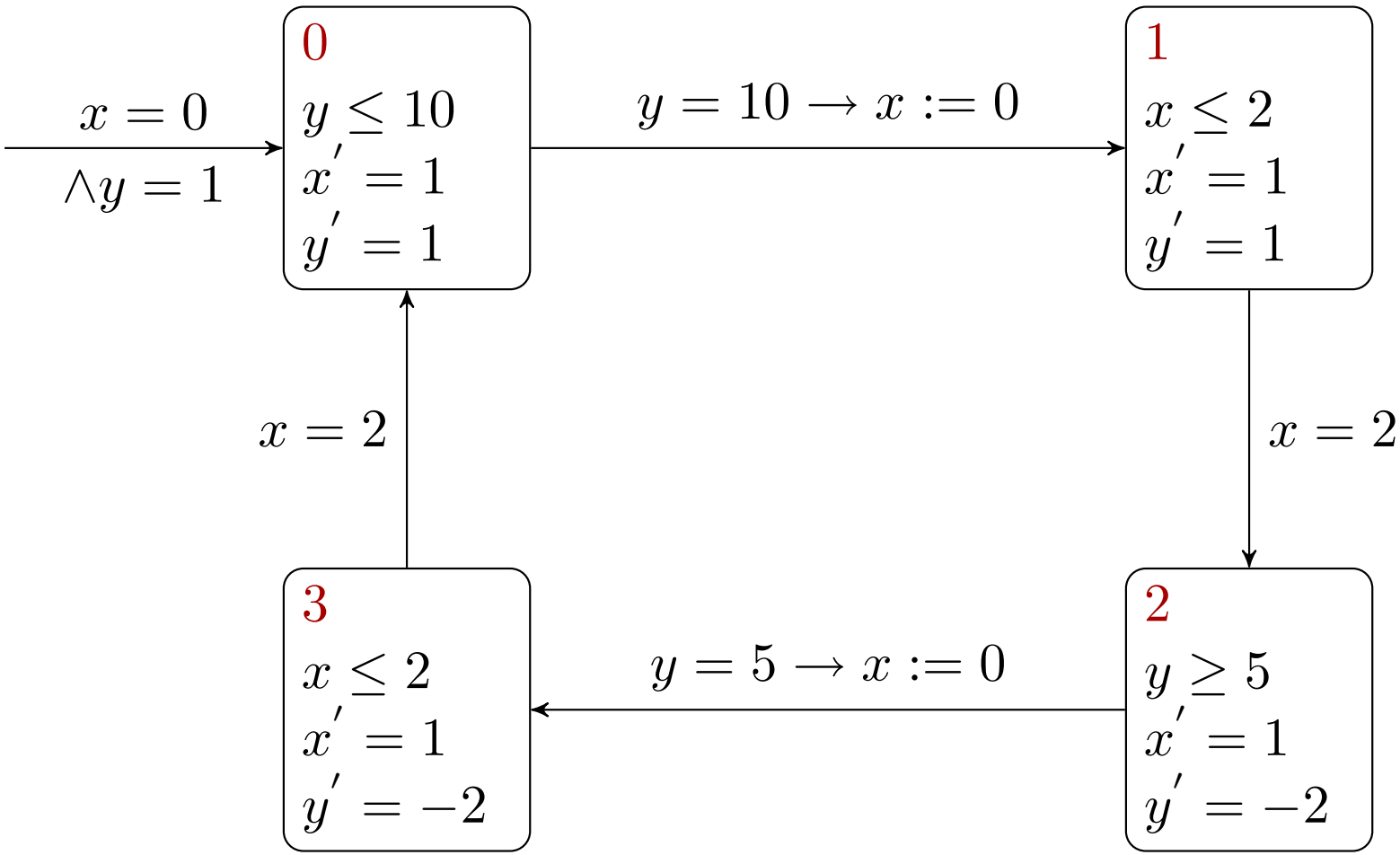
- For all edges, the transision relation is defined by a guarded set of nondeterministic assignments:

$$\psi \Rightarrow \{x := [\alpha_x, \beta_x] \mid x \in Var\},$$

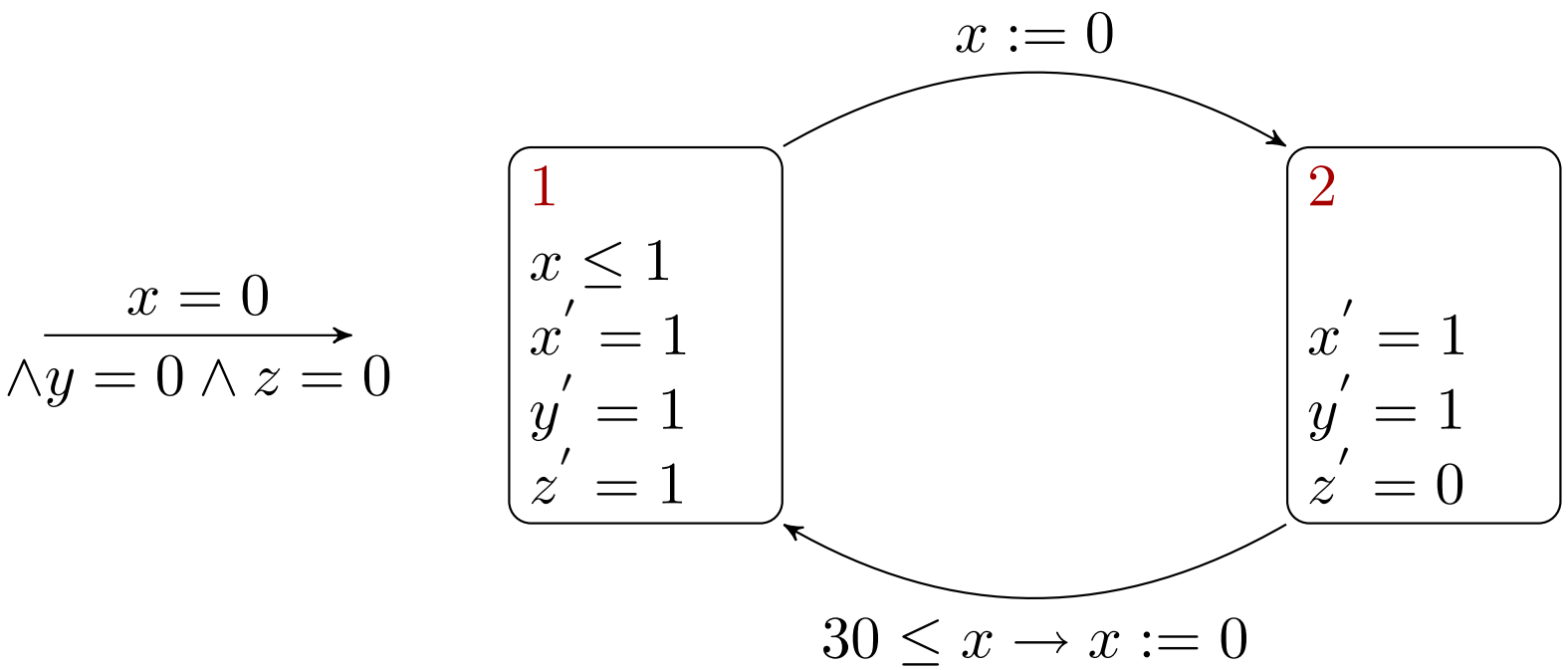
where the quard ψ is a linear formula and α_x, β_x are linear terms:

$$(\nu, \nu') \in \mu \quad \textit{iff} \quad \nu(\psi) \wedge \forall x \in Var.\ \nu(\alpha_x) \leq \nu'(x) \leq \nu(\beta_x).$$

Water-level monitor



Leaking gas burner



Reminder: Semantics of hybrid systems

$$\frac{(l, a, \mu, l') \in Edge \quad (\nu, \nu') \in \mu \quad \nu' \in Inv(l')}{(l, \nu) \xrightarrow{a} (l', \nu')} \text{ Rule } \texttt{Discrete}$$

$$\frac{\begin{array}{l} f \in Act(l) \quad f(0) = \nu \quad f(t) = \nu' \\ t \geq 0 \quad \forall 0 \leq t' \leq t . f(t') \in Inv(l) \end{array}}{(l, \nu) \xrightarrow{t} (l, \nu')} \text{ Rule } \texttt{Time}$$

Time-deterministic systems

Definition

For time-deterministic hybrid systems we define the "time can progress" predicate:

$$tcp_l[\nu](t) \quad iff \quad \forall 0 \leq t' \leq t.\ \varphi_l[\nu](t') \in Inv(l).$$

For time-deterministic systems we can rewrite the time-step rule to:

$$\frac{tcp_l[\nu](t)}{(l,\nu) \xrightarrow{t} (l, \varphi_l[\nu](t))}\ \texttt{Rule}_{\texttt{Time}}$$

Forward analysis

- We define the *forward time closure* $\langle P \rangle_l^{\nearrow}$ of $P \subseteq V$ at $l \in Loc$ as the set of valuations reachable from P by letting time progress:

$$\nu' \in \langle P \rangle_l^{\nearrow} \quad \textit{iff} \quad \exists \nu \in P.\ \exists t \in \mathbb{R}^{\geq 0}.\ tcp_l[\nu](t) \wedge \nu' = \phi_l[\nu](t).$$

- Extension to regions $R = \cup_{l \in Loc}(l, R_l)$:

$$\langle R \rangle^{\nearrow} = \cup_{l \in Loc}(l, \langle R_l \rangle_l^{\nearrow}).$$

- We define the *postcondition* $post_e[P]$ of P with respect to an edge $e = (l, a, \mu, l')$ as the set of valuations reachable from P by e:

$$\nu' \in post_e[P] \quad \textit{iff} \quad \exists \nu \in P.\ (\nu, \nu') \in \mu.$$

- Extension to regions $R = \cup_{l \in Loc}(l, R_l)$:

$$post[R] = \cup_{e=(l,a,\mu,l') \in Edge}(l', post_e[R_l]).$$

- A *symbolic run* of the linear hybrid automaton A is a finite or infinite sequence

$$\pi : \ (l_0, P_0)\ (l_1, P_1) \quad \ldots (l_i, P_i) \quad \ldots$$

of regions such that for all $i \geq 0$ there is an edge e_i from l_i to l_{i+1} and

$$P_{i+1} = post_{e_i}[\langle P_i \rangle_{l_i}^{\nearrow}].$$

- Correspondence between runs and symbolic runs?
- Given a region $I \subseteq \Sigma$, the *reachable region* $(I \mapsto^*) \subseteq \Sigma$ of I is the set of all states that are reachable from states in I:

$$\sigma \in (I \mapsto^*) \quad \textit{iff} \quad \exists \sigma' \in I.\ \sigma' \rightarrow^* \sigma.$$

Lemma

Let $I = \cup_{l \in Loc}(l, I_l)$ be a region of the linear hybrid automaton A. The reachable region $(I, \mapsto^*) = \cup_{l \in Loc}(l, R_l)$ is the least fixpoint of the equation

$$X = \langle I \cup post[X] \rangle^{\nearrow}$$

or, equivalently, for all locations $l \in Loc$, the set R_l of valuations is the least fixpoint of the set of equations

$$X_l = \langle I_l \cup \bigcup_{e=(l',a,\mu,l) \in Edge} post_e[X_{l'}] \rangle_l^{\nearrow}.$$

Lemma

For all linear hybrid automata, if $P \subseteq V$ is a linear set of valuations, then for all $l \in Loc$ and $e \in Edge$, both $\langle P \rangle_l^{\nearrow}$ and $post_e[P]$ are linear sets of valuations.

Example: Leaking gas burner

Backward analysis

- We define the *backward time closure* $\langle P \rangle_l^{\swarrow}$ of $P \subseteq V$ at $l \in Loc$ as the set of valuations from which it is possible to reach a valuation in P by letting time progress:

$$\nu' \in \langle P \rangle_l^{\swarrow} \quad iff \quad \exists \nu \in P.\ \exists t \in \mathbb{R}^{\geq 0}.\ tcp_l[\nu'](t) \wedge \nu = \phi_l[\nu'](t).$$

- Extension to regions $R = \cup_{l \in Loc}(l, R_l)$:

$$\langle R \rangle^{\swarrow} = \cup_{l \in Loc}(l, \langle R_l \rangle_l^{\swarrow}).$$

- We define the *precondition* $pre_e[P]$ of P with respect to an edge $e = (l, a, \mu, l')$ as the set of valuations from which it is possible to reach a valuation from P by e:

$$\nu' \in pre_e[P] \quad iff \quad \exists \nu \in P.\ (\nu', \nu) \in \mu.$$

- Extension to regions $R = \cup_{l \in Loc}(l, R_l)$:

$$pre[R] = \cup_{e=(l',a,\mu,l) \in Edge}(l', post_e[R_l]).$$

- Given a region $R \subseteq \Sigma$, the *initial region* $(\mapsto^* R) \subseteq \Sigma$ of R is the set of all states from which a states in R is reachable:

$$\sigma \in (\mapsto^* R) \quad \textit{iff} \quad \exists \sigma' \in R.\ \sigma \rightarrow^* \sigma'.$$

Lemma

Let $R = \cup_{l \in Loc}(l, R_l)$ be a region of the linear hybrid automaton A. The initial region $I = \cup_{l \in Loc}(l, I_l)$ is the least fixpoint of the equation

$$X = \langle R \cup pre[X] \rangle^{\swarrow}$$

or, equivalently, for all locations $l \in Loc$, the set I_l of valuations is the least fixpoint of the set of equations

$$X_l = \langle R_l \cup \bigcup_{e=(l,a,\mu,l') \in Edge} pre_e[X_{l'}] \rangle_l^{\swarrow} .$$

Lemma

For all linear hybrid automata, if $P \subseteq V$ is a linear set of valuations, then for all $l \in Loc$ and $e \in Edge$, both $\langle P \rangle_l^{\swarrow}$ and $pre_e[P]$ are linear sets of valuations.

Example: Leaking gas burner

Approximate analysis

If the (forward or backward) iterative techniques does not converge, we can compute *upper approximations* of the sets

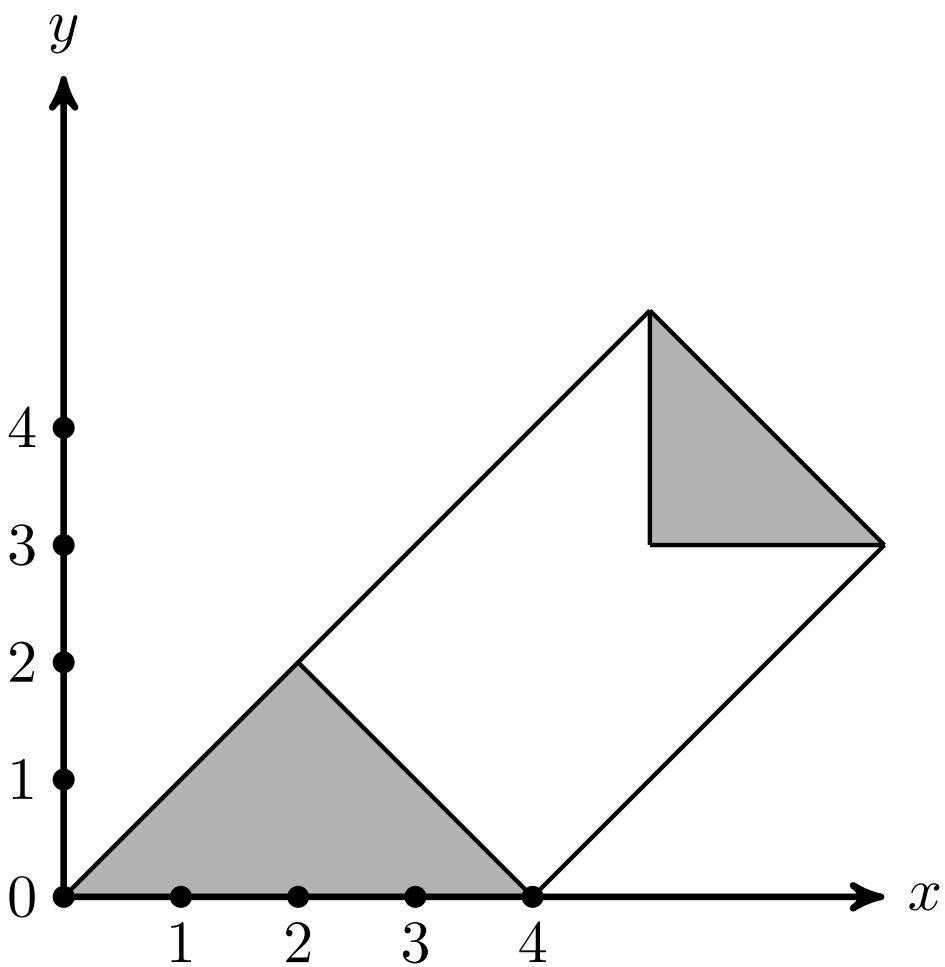
- $(I \mapsto^*)$ of states which are reachable from the initial states I (forward analysis)
- $(\mapsto^* R)$ of states from which the region R is reachable (backward analysis)

Two approaches:

- Convex hull
- Widening

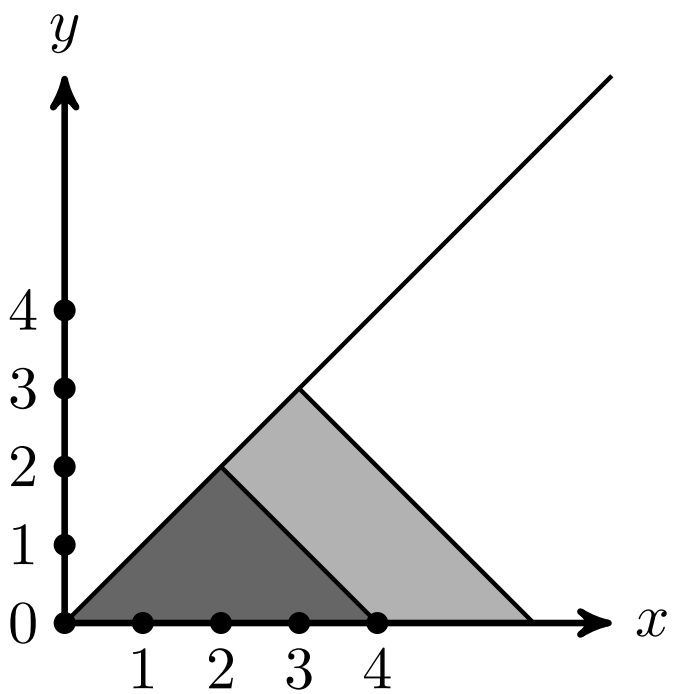
Convex hull

Instead of computing the *union* of sets, compute the *convex hull*, i.e., the least convex polyhedron containing the operands of the union.



Widening

To enforce the convergence of iterations, we can apply a *widening technique*.
Basic idea: extrapolate the limit of a sequence of polyhedra (occurring in the non-terminating fixpoint computation), in such a way that an upper limit be always reached within a finite number of iterations.
Apply the widening for at least one location in each *loop* of the graph of the hybrid system.



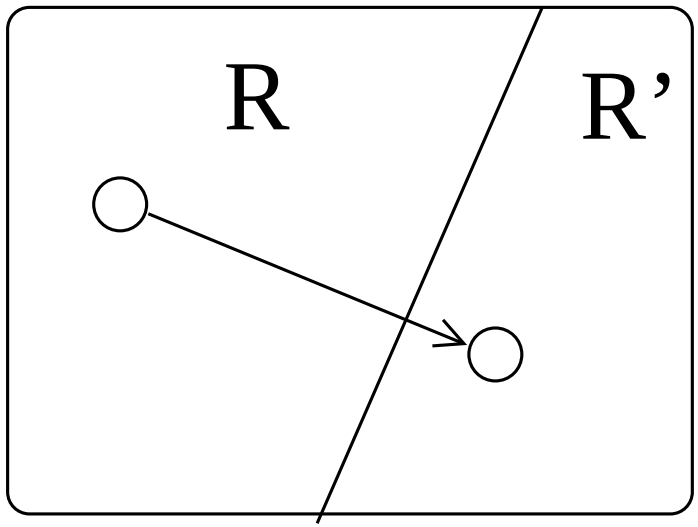
Minimization

- Since the reachability problem for linear hybrid automata is undecidable, we cannot give a complete algorithm for computing a finite abstraction (bisimulation), like in the case of timed automata.
- Thus it is not a surprise, that reachability analysis does not always reach a fixpoint.
- To increase the chance to success, we can extend (e.g., forward) reachability analysis with a minimization analysis.
- Given an initial condition and a safety specification, we could try to construct a partitioning of the state space, by
  - specifying an initial partitioning into "good" and "bad" states (according to the specification), and
  - refining this partitioning according to (forward) reachability) until we can draw conclusions wrt. to the validity of the specification.
- To explain it more exactly, first we need some formalisms...

Definition

The next relation $\mapsto$ on regions is defined by

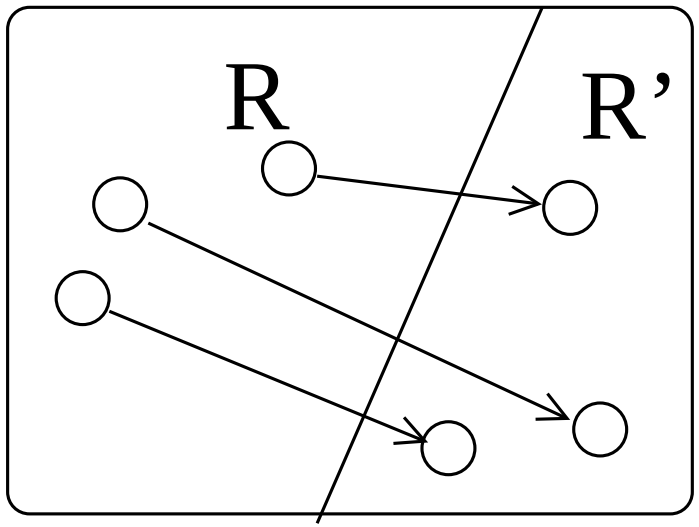
$$R \mapsto R' \quad \text{iff} \quad \exists \sigma \in R.\ \exists \sigma' \in R.\ \sigma \to \sigma'.$$



Definition

Let π be a partition of the state space Σ. A region $R \in \pi$ is called *stable* iff for all $R' \in \pi$,

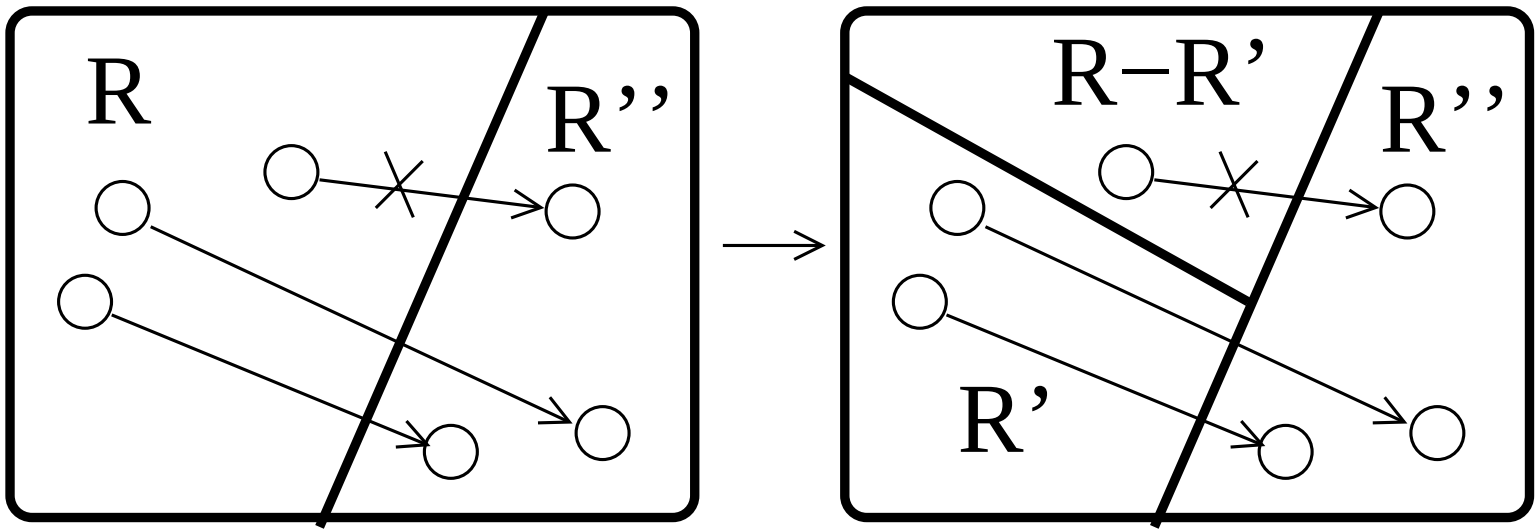
$$R \mapsto R' \quad \text{implies} \quad \forall \sigma \in R.\ \{\sigma\} \mapsto R'.$$



Definition

$\mathsf{split}[\pi](R) :=$

$$\begin{cases} \{R', R - R'\} & \text{if } \exists R'' \in \pi.\ R' = \mathsf{pre}[< R'' >^{\swarrow}] \cap R \wedge R' \subset R, \\ \{R\} & \text{otherwise.} \end{cases}$$



- A partition π is a *bisimulation* iff every region $R \in \pi$ is stable.
- The partition π *respects* the region R_F iff for every region $R \in \pi$, either $R \subseteq R_F$ or $R \cap R_F = \emptyset$.
- Idea: The partitioning must respect the specification, and must be stable for the regions reachable from regions containing some initial states.
- The specification holds iff in this abstraction there is no region containing a "bad" state and reachable from a regions containing some initial state.
- In the following let I be the initial states and R_F be the "bad" states.

$\pi := \{R_F, \Sigma - R_F\};\ \alpha := \{R | R \cap I \neq \emptyset\};\ \beta := \emptyset;$

while $\alpha \neq \beta$ **do**

choose $R \in (\alpha - \beta)$; $\quad \alpha' := \mathsf{split}[\pi](R)$;

if $\alpha' = \{R\}$ **then**

$\beta := \beta \cup \{R\}$;

$\alpha := \alpha \cup \{R' \in \pi \mid R \mapsto R'\}$;

else

$\alpha := \alpha - \{R\}$;

if $\exists R' \in \alpha'$ such that $R' \cap I \neq \emptyset$ **then** $\alpha := \alpha \cup \{R`\}$ **fi**

$\beta := \beta - \{R' \in \pi \mid R' \mapsto R\}$;

$\pi := (\pi - \{R\}) \cup \alpha'$;

fi

od

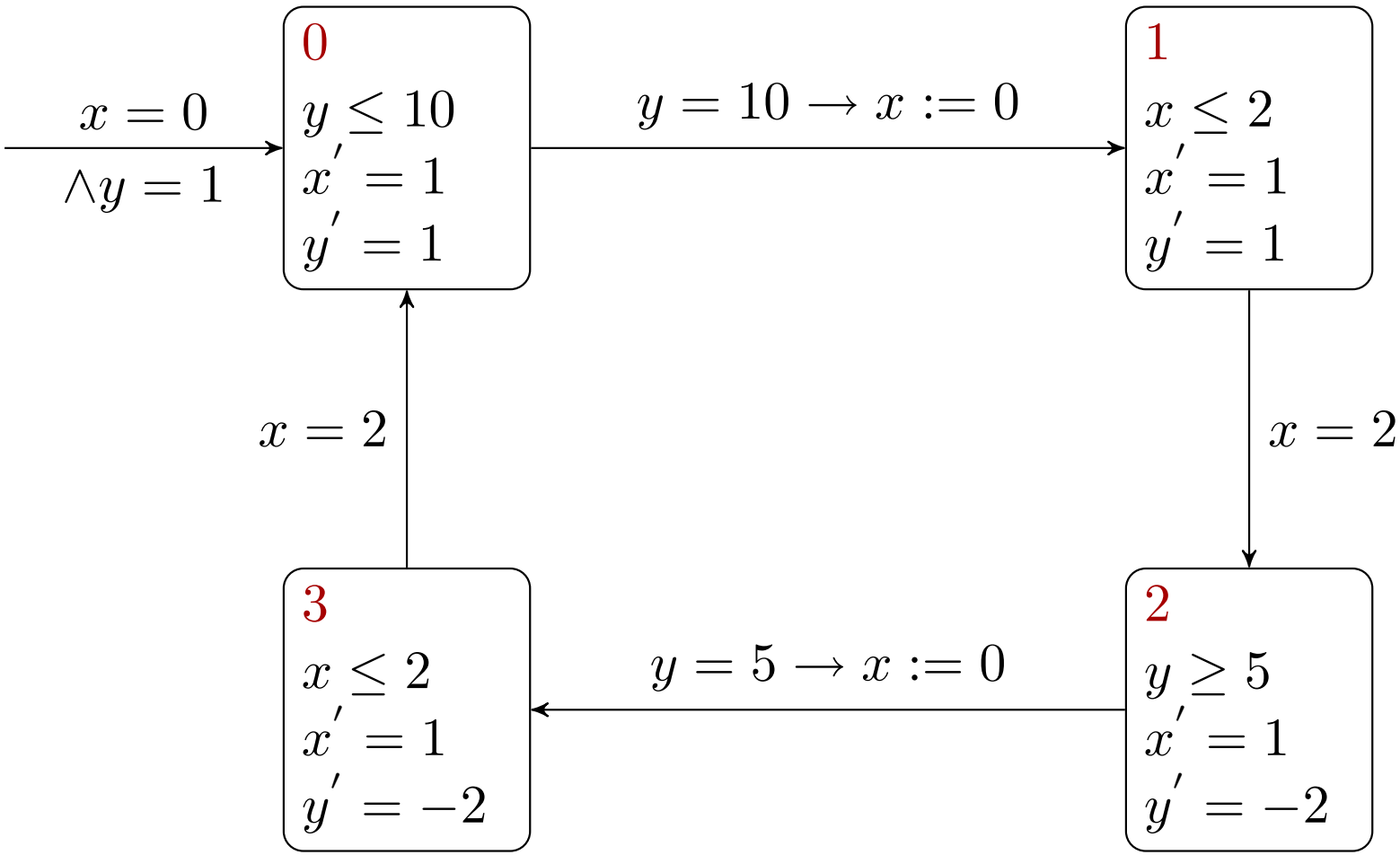
return there is $R \in \alpha$ **such that** $R \subseteq R_F$;

Lemma

The procedure returns TRUE iff $I \mapsto^* R_F$.

- If the regions R_F and I are linear, all regions that are constructed by the procedure are linear.
- The algorithm terminates iff the coarsest bisimulation has only a finite number of equivalence classes.

Water-level monitor



$$\begin{aligned} I &= (ps = 0 \wedge x = 0 \wedge y = 1) \\ R_F &= (y < 1 \wedge y > 12) \end{aligned}$$

Initial partitioning: $\pi_1 = \{$

$$\begin{aligned} R_{00} &= (pc = 0 \wedge 1 \leq y \leq 12), \quad R_{01} = (pc = 0 \wedge (y < 1 \vee y > 12)), \\ R_{10} &= (pc = 1 \wedge 1 \leq y \leq 12), \quad R_{11} = (pc = 1 \wedge (y < 1 \vee y > 12)), \\ R_{20} &= (pc = 2 \wedge 1 \leq y \leq 12), \quad R_{21} = (pc = 2 \wedge (y < 1 \vee y > 12)), \\ R_{30} &= (pc = 3 \wedge 1 \leq y \leq 12), \quad R_{31} = (pc = 3 \wedge (y < 1 \vee y > 12))\} \end{aligned}$$

$\alpha = \{R_{00}\}, \ \beta = \emptyset.$

Initial partitioning: $\pi_1 = \{$

$$\mathbf{R_{00}} = (\mathbf{pc = 0 \wedge 1 \leq y \leq 12}), \quad R_{01} = (pc = 0 \wedge (y < 1 \vee y > 12)),$$
$$R_{10} = (pc = 1 \wedge 1 \leq y \leq 12), \quad R_{11} = (pc = 1 \wedge (y < 1 \vee y > 12)),$$
$$R_{20} = (pc = 2 \wedge 1 \leq y \leq 12), \quad R_{21} = (pc = 2 \wedge (y < 1 \vee y > 12)),$$
$$R_{30} = (pc = 3 \wedge 1 \leq y \leq 12), \quad R_{31} = (pc = 3 \wedge (y < 1 \vee y > 12))\}$$

$\alpha = \{R_{00}\},\ \beta = \emptyset.$

Choose $R = R_{00}$:
split$[\pi_1](R_{00}) = \{(pc = 0 \wedge 1 \leq y \leq 10), (pc = 0 \wedge 10 < y \leq 12)\}.$

New partitioning: $\pi_2 = \{$

$$\mathbf{R_{000}} = (\mathbf{pc = 0 \wedge 1 \leq y \leq 10}),$$
$$\mathbf{R_{001}} = (\mathbf{pc = 0 \wedge 10 < y \leq 12}), \quad R_{01} = (pc = 0 \wedge (y < 1 \vee y > 12)),$$
$$R_{10} = (pc = 1 \wedge 1 \leq y \leq 12), \quad R_{11} = (pc = 1 \wedge (y < 1 \vee y > 12)),$$
$$R_{20} = (pc = 2 \wedge 1 \leq y \leq 12), \quad R_{21} = (pc = 2 \wedge (y < 1 \vee y > 12)),$$
$$R_{30} = (pc = 3 \wedge 1 \leq y \leq 12), \quad R_{31} = (pc = 3 \wedge (y < 1 \vee y > 12))\}$$

$\alpha = \{R_{000}\},\ \beta = \emptyset.$

Current partitioning: $\pi_2 = \{$

$\mathbf{R_{000} = (pc = 0 \wedge 1 \leq y \leq 10)}$,
$R_{001} = (pc = 0 \wedge 10 < y \leq 12)$, $R_{01} = (pc = 0 \wedge (y < 1 \vee y > 12))$,
$R_{10} = (pc = 1 \wedge 1 \leq y \leq 12)$, $R_{11} = (pc = 1 \wedge (y < 1 \vee y > 12))$,
$R_{20} = (pc = 2 \wedge 1 \leq y \leq 12)$, $R_{21} = (pc = 2 \wedge (y < 1 \vee y > 12))$,
$R_{30} = (pc = 3 \wedge 1 \leq y \leq 12)$, $R_{31} = (pc = 3 \wedge (y < 1 \vee y > 12))\}$

$\alpha = \{R_{000}\},\ \beta = \emptyset$.

Choose $R = R_{000}$: stable.

New partitioning: π_2, $\alpha = \{R_{000}, R_{001}, R_{10}\},\ \beta = \{R_{000}\}$.

Choose $R = R_{001}$: stable.

New partitioning: π_2, $\alpha = \{R_{000}, R_{001}, R_{10}\},\ \beta = \{R_{000}, R_{001}\}$.

Current partitioning: $\pi_2 = \{$

$$
\begin{array}{ll}
R_{000} = (pc = 0 \wedge 1 \leq y \leq 10), & \\
R_{001} = (pc = 0 \wedge 10 < y \leq 12), & R_{01} = (pc = 0 \wedge (y < 1 \vee y > 12)), \\
\mathbf{R_{10} = (pc = 1 \wedge 1 \leq y \leq 12)}, & R_{11} = (pc = 1 \wedge (y < 1 \vee y > 12)), \\
R_{20} = (pc = 2 \wedge 1 \leq y \leq 12), & R_{21} = (pc = 2 \wedge (y < 1 \vee y > 12)), \\
R_{30} = (pc = 3 \wedge 1 \leq y \leq 12), & R_{31} = (pc = 3 \wedge (y < 1 \vee y > 12))\}
\end{array}
$$

$\alpha = \{R_{000}, R_{001}, R_{10}\},\ \beta = \{R_{000}, R_{001}\}.$

Choose $R = R_{10}$: $\quad \mathsf{split}[\pi_2](R_{10}) =$
$\{(pc = 1 \wedge 0 \leq x \leq 2 \wedge 1 \leq y \leq 12), (pc = 1 \wedge x > 2 \wedge 1 \leq y \leq 12)\}$

New partitioning: $\pi_3 = \{$

$$
\begin{array}{ll}
\ldots & \\
\mathbf{R_{100} = (pc = 1 \wedge 0 \leq x \leq 2 \wedge 1 \leq y \leq 12)}, & \\
\mathbf{R_{101} = (pc = 1 \wedge x > 2 \wedge 1 \leq y \leq 12)}, & \ldots \\
\ldots &
\end{array}
$$

$\alpha = \{R_{000}, R_{001}\},\ \beta = \{R_{001}\}.$

Current partitioning: $\pi_3 = \{$

$$
\begin{aligned}
&\mathbf{R_{000} = (pc = 0 \wedge 1 \leq y \leq 10)},\\
&R_{001} = (pc = 0 \wedge 10 < y \leq 12),\\
&R_{100} = (pc = 1 \wedge 0 \leq x \leq 2 \wedge 1 \leq y \leq 12),\\
&R_{101} = (pc = 1 \wedge x > 2 \wedge 1 \leq y \leq 12),\\
&R_{20} = (pc = 2 \wedge 1 \leq y \leq 12),\\
&R_{30} = (pc = 3 \wedge 1 \leq y \leq 12),\\
&R_{01} = (pc = 0 \wedge (y < 1 \vee y > 12)),\\
&R_{11} = (pc = 1 \wedge (y < 1 \vee y > 12)),\\
&R_{21} = (pc = 2 \wedge (y < 1 \vee y > 12)),\\
&R_{31} = (pc = 3 \wedge (y < 1 \vee y > 12))\}
\end{aligned}
$$

$\alpha = \{R_{000}, R_{001}\},\ \beta = \{R_{001}\}.$
Choose $R = R_{000}$: stable
New partitioning: π_3, $\alpha = \{R_{000}, R_{001}, R_{100}\},\ \beta = \{R_{001}, R_{000}\}.$

Current partitioning: $\pi_3 = \{$

$$
\begin{aligned}
&R_{000} = (pc = 0 \wedge 1 \leq y \leq 10),\\
&R_{001} = (pc = 0 \wedge 10 < y \leq 12),\\
&\mathbf{R_{100} = (pc = 1 \wedge 0 \leq x \leq 2 \wedge 1 \leq y \leq 12)},\\
&R_{101} = (pc = 1 \wedge x > 2 \wedge 1 \leq y \leq 12),\\
&R_{20} = (pc = 2 \wedge 1 \leq y \leq 12),\\
&R_{30} = (pc = 3 \wedge 1 \leq y \leq 12),\\
&R_{01} = (pc = 0 \wedge (y < 1 \vee y > 12)),\\
&R_{11} = (pc = 1 \wedge (y < 1 \vee y > 12)),\\
&R_{21} = (pc = 2 \wedge (y < 1 \vee y > 12)),\\
&R_{31} = (pc = 3 \wedge (y < 1 \vee y > 12))\}
\end{aligned}
$$

$\alpha = \{R_{000}, R_{001}, R_{100}\},\ \beta = \{R_{001}, R_{000}\}.$
Choose $R = R_{100}$: stable
New partitioning: π_3,
$\alpha = \{R_{000}, R_{001}, R_{100}, R_{101}, R_{20}\},\ \beta = \{R_{001}, R_{000}, R_{100}\}.$

Current partitioning: $\pi_3 = \{$

$$
\begin{aligned}
&R_{000} = (pc = 0 \wedge 1 \leq y \leq 10),\\
&R_{001} = (pc = 0 \wedge 10 < y \leq 12),\\
&R_{100} = (pc = 1 \wedge 0 \leq x \leq 2 \wedge 1 \leq y \leq 12),\\
&\mathbf{R_{101} = (pc = 1 \wedge x > 2 \wedge 1 \leq y \leq 12)},\\
&R_{20} = (pc = 2 \wedge 1 \leq y \leq 12),\\
&R_{30} = (pc = 3 \wedge 1 \leq y \leq 12),\\
&R_{01} = (pc = 0 \wedge (y < 1 \vee y > 12)),\\
&R_{11} = (pc = 1 \wedge (y < 1 \vee y > 12)),\\
&R_{21} = (pc = 2 \wedge (y < 1 \vee y > 12)),\\
&R_{31} = (pc = 3 \wedge (y < 1 \vee y > 12))\}
\end{aligned}
$$

$\alpha = \{R_{000}, R_{001}, R_{100}, R_{101}, R_{20}\},\ \beta = \{R_{001}, R_{000}, R_{100}\}.$
Choose $R = R_{101}$: stable
New partitioning: π_3,
$\alpha = \{R_{000}, R_{001}, R_{100}, R_{101}, R_{20}\},\ \beta = \{R_{001}, R_{000}, R_{100}, R_{101}\}.$

Current partitioning: $\pi_3 = \{$

$$
\begin{aligned}
&R_{000} = (pc = 0 \wedge 1 \leq y \leq 10),\\
&R_{001} = (pc = 0 \wedge 10 < y \leq 12),\\
&R_{100} = (pc = 1 \wedge 0 \leq x \leq 2 \wedge 1 \leq y \leq 12),\\
&R_{101} = (pc = 1 \wedge x > 2 \wedge 1 \leq y \leq 12),\\
&\mathbf{R_{20} = (pc = 2 \wedge 1 \leq y \leq 12)},\\
&R_{30} = (pc = 3 \wedge 1 \leq y \leq 12),\\
&R_{01} = (pc = 0 \wedge (y < 1 \vee y > 12)),\\
&R_{11} = (pc = 1 \wedge (y < 1 \vee y > 12)),\\
&R_{21} = (pc = 2 \wedge (y < 1 \vee y > 12)),\\
&R_{31} = (pc = 3 \wedge (y < 1 \vee y > 12))\}
\end{aligned}
$$

$\alpha = \{R_{000}, R_{001}, R_{100}, R_{101}, R_{20}\},\ \beta = \{R_{001}, R_{000}, R_{100}, R_{101}\}.$
Choose $R = R_{20}$:
split$[\pi_3](R_{20}) = \{(pc = 2 \wedge 5 \leq y \leq 12), (pc = 2 \wedge 1 \leq y < 5)\}$

New partitioning: $\pi_4 =$
$\{\ldots \quad \mathbf{R_{200} = (pc = 2 \wedge 5 \leq y \leq 12), R_{201} = (pc = 2 \wedge 1 \leq y < 5)}, \ldots\}$
$\alpha = \{R_{000}, R_{001}, R_{100}, R_{101}\},\ \beta = \{R_{001}, R_{000}, R_{101}\}.$

Current partitioning: $\pi_4 = \{$

$$R_{000} = (pc = 0 \wedge 1 \leq y \leq 10),\ R_{001} = (pc = 0 \wedge 10 < y \leq 12),$$
$$\mathbf{R_{100} = (pc = 1 \wedge 0 \leq x \leq 2 \wedge 1 \leq y \leq 12)},$$
$$R_{101} = (pc = 1 \wedge x > 2 \wedge 1 \leq y \leq 12),$$
$$R_{200} = (pc = 2 \wedge 5 \leq y \leq 12),\ R_{201} = (pc = 2 \wedge 1 \leq y < 5),$$
$$R_{30} = (pc = 3 \wedge 1 \leq y \leq 12),$$
$$R_{01} = (pc = 0 \wedge (y < 1 \vee y > 12)),$$
$$R_{11} = (pc = 1 \wedge (y < 1 \vee y > 12)),$$
$$R_{21} = (pc = 2 \wedge (y < 1 \vee y > 12)),$$
$$R_{31} = (pc = 3 \wedge (y < 1 \vee y > 12))\}$$

$\alpha = \{R_{000}, R_{001}, R_{100}, R_{101}\},\ \beta = \{R_{001}, R_{000}, R_{101}\}.$
Choose $R = R_{100}$: $\mathsf{split}[\pi_4](R_{100}) = \{(pc = 1 \wedge 0 \leq x \leq 2 \wedge 3 \leq y \leq 12 \wedge 3 \leq y - x \leq 12), (pc = 1 \wedge 0 \leq x \leq 2 \wedge 1 \leq y < 3 \wedge 1 \leq y - x < 3)\}$

New partitioning:
$\pi_5 = \{\ldots R_{1000} = (pc = 1 \wedge 0 \leq x \leq 2 \wedge 3 \leq y \leq 12 \wedge 3 \leq y - x \leq 12), R_{1001} = (pc = 1 \wedge 0 \leq x \leq 2 \wedge 1 \leq y < 3 \wedge 1 \leq y - x < 3), \ldots\}$

$R_{000}, R_{1000}, R_{200}, R_{201}$ are stable $\rightarrow$
$\alpha = \{R_{000}, R_{001}, R_{1000}, R_{200}, R_{201}, R_{30}\}$,
$\beta = \{R_{000}, R_{001}, R_{1000}, R_{200}, R_{201}\}$.

Just 5 steps more and we are ready, with no bad states in α!

Aside from linear hybrid automata, a common approach to obtaining bisimulations has been to utilize an algorithm which refines an initial partition of the state space until it becomes compatible with the system dynamics and the property to be preserved.
Using this approach, there are three main issues that must be resolved:

1. When does the algorithm terminate after a finite number of iterations?
2. When does the resulting partition consists of a finite number of equivalence classes?
3. Are all the steps of the algorithm constructive?

Resolving all three issues results in a decidable problem.

1. When does the algorithm terminate after a finite number of iterations?
2. When does the resulting partition consists of a finite number of equivalence classes?
3. Are all the steps of the algorithm constructive?

Attacking the first two issues has been solved either by providing a bisimulation (timed automata), or by transforming the problem to one for which a bisimulation is known to exist (rectangular automata).
The third issue is typically tackled using quantifier elimination techniques from mathematical logic.