Modeling and Analysis of Hybrid Systems

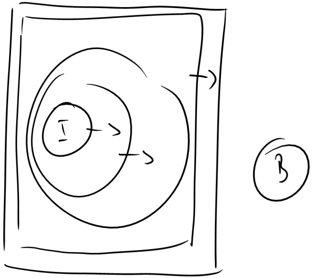
Approximate analysis and minimization

Prof. Dr. Erika Ábrahám

Informatik 2 - Theory of Hybrid Systems
RWTH Aachen University

SS 2013

F :



$x=0$

$\rightarrow \bullet \circlearrowleft \; x := x+1$

Literature

Alur et al.: The algorithmic analysis of hybrid systems

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

Approximate analysis

If the (forward or backward) iterative technique does not terminate, we can compute over-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)

If the (forward or backward) iterative technique does not terminate, we can compute over-approximations of the sets

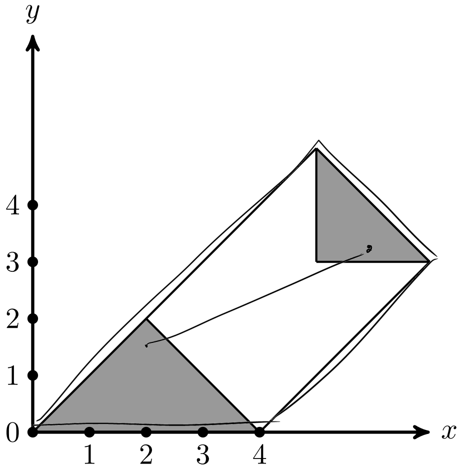
- $(I \mapsto^*)$ of states which are reachable from the initial states I (forward analysis)
- $(\mapsto^* R)$ of states from which the region R is reachable (backward analysis)

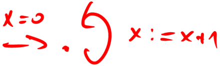
Two approaches:

- 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.





$x \in \mathbb{R}$

1) $R_0 = x = 0$

$R_1 = x = 1$

$R_0 \cup R_1 = x = 0 \lor x = 1$

$conv(R_0 \cup R_1) = 0 \leq x \leq 1$

2) $R_i = x = i$

$\overline{R_i} = 0 \leq x$

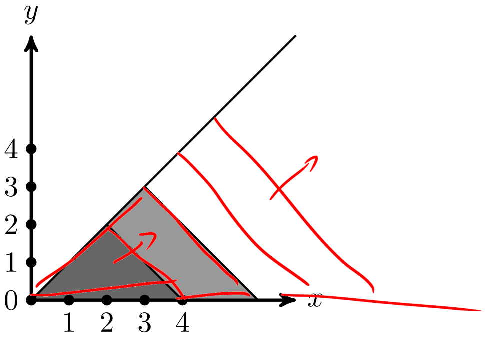
$R_{i+1} = 0 \leq x$

Widening

To enforce the convergence of iterations, we can apply a widening technique.

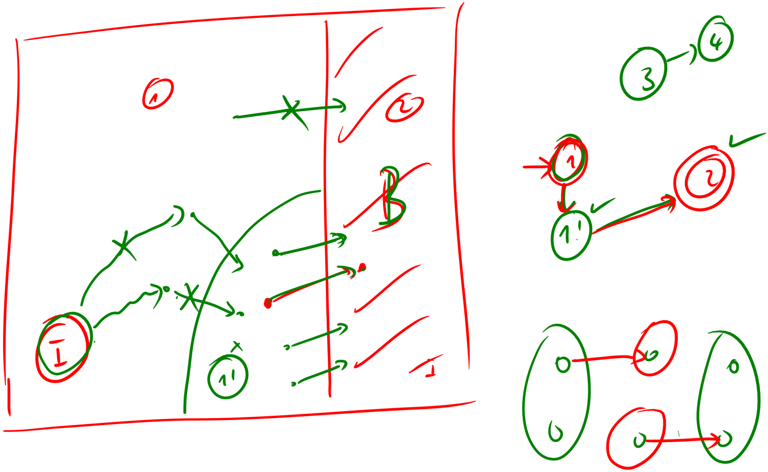
Basic idea: extrapolate the limit of a sequence of polyhedra (occurring in the non-terminating fixpoint computation), in such a way that an upper limit be always reached within a finite number of iterations.

Apply the widening for at least one location in each loop of the graph of the hybrid automaton.

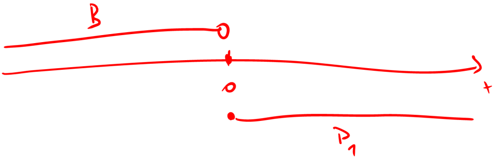


Minimization

The basic idea of abstraction



$I = x=0$



$B : x < 0$

$P_1 : x \geq 0$

$I \wedge P_1$

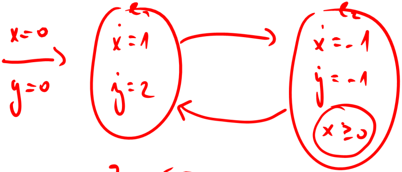
$I \wedge B$

$D(\tau(P_1))$

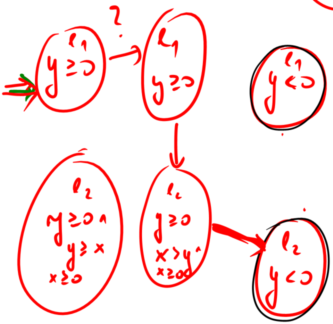
$\tau(P_1) = \exists x', t .\ P_1(x') \wedge t \geq 0 \wedge x = x' + t$

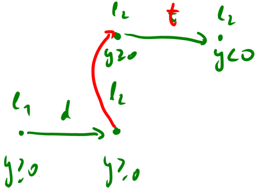
$D(P_1) = \exists x' .\ \underline{x' \geq 0} \wedge \underline{x = x' + 1} \wedge \underline{x < 0}$

$x \geq 1 \wedge x < 0 \equiv \text{false}$


x = 0
y = 0
ℓ1
ẋ = 1
ẏ = 2
ℓ2
ẋ = -1
ẏ = -1
x ≥ 0

$B: y < 0$


ℓ1
y ≥ 0
ℓ1
y ≥ 0
ℓ1
y < 0
ℓ2
y ≥ 0 ∧
y ≥ x
x ≥ 0
ℓ2
y ≥ 0
x > y ∧
x ≥ 0
ℓ2
y < 0


ℓ2
t
ℓ2
y ≥ 0
y < 0
ℓ2
ℓ1
d
y ≥ 0
y ≥ 0

Abstract refinement

- Sometimes it is possible to define an abstraction which is a bisimulation, e.g., for TCTL model checking of timed automata.
- For harder (or undecidable) problems we can try to define an initial abstraction and refine it as long as it leads to spurious counterexamples.
- Some abstraction refinement techniques:
  - Predicate abstraction
  - Counterexample-guided abstraction refinement (CEGAR)
  - Minimization

Abstraction refinement

Using abstraction refinement, there are three main issues that must be resolved:

1. The algorithm should terminate after a finite number of iterations.
2. The resulting partition should consist of a finite number of equivalence classes.
3. The steps of the algorithm should be constructive.

Resolving all three issues results in a decidable problem.

$R := Reach(I)$;

$y := e^x$;

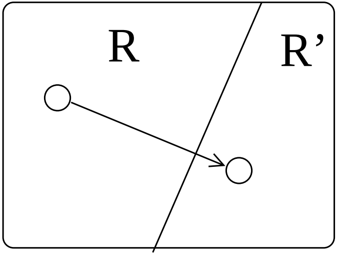
Minimization for linear hybrid automata

- Since the reachability problem for linear hybrid automata is undecidable, we cannot give any 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 algorithm.
- 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

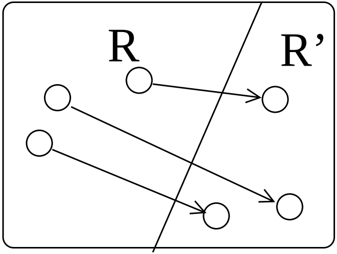
$$R \mapsto R' \quad \text{iff} \quad \exists \sigma \in R.\ \exists \sigma' \in R.\ \sigma \to \sigma'.$$



Definition

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

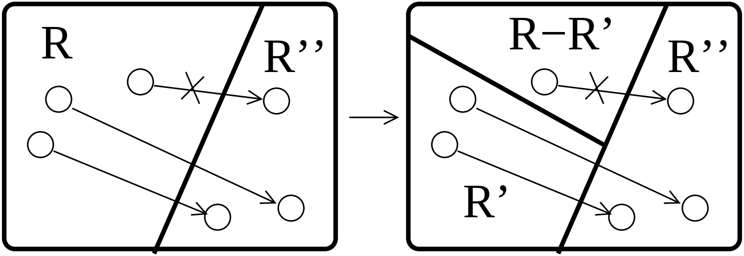
$$R \mapsto R' \quad \text{implies} \quad \forall \sigma \in R.\ \{\sigma\} \mapsto R'.$$



Definition

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

$$\begin{cases} \{R', R \setminus R'\} & \text{if } \exists R'' \in \pi.\ R' = \mathcal{D}^{-}(\mathcal{T}^{-}(R'')) \cap R \wedge R' \neq R, \\ \{R\} & \text{otherwise.} \end{cases}$$



- A partition π is a bisimulation iff every region $R \in \pi$ is stable.
- The partition π respects the region R_{bad} iff for every region $R \in \pi$, either $R \subseteq R_{bad}$ or $R \cap R_{bad} = \emptyset$.
- 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 being reachable from a region containing some initial state.
- In the following let I be the initial states and R_{bad} be the bad states.

$\pi := \{R_{bad}, \Sigma \setminus R_{bad}\}$; *reach* $:= \{R \in \pi \mid R \cap I \neq \emptyset\}$; *stable* $:= \emptyset$;

while *reach* $\neq$ *stable* **do**

  choose $R \in$ (*reach* $\setminus$ *stable*); *reach*$'$:= split$[\pi](R)$;

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

    stable := *stable* $\cup \{R\}$;

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

  else

    reach := (*reach* $\setminus \{R\}) \cup \{R' \mid R' \in$ *reach*$' \wedge R' \cap I \neq \emptyset\}$;

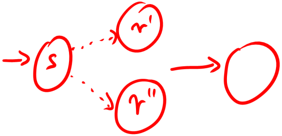
    stable := *stable* $\setminus \{R' \in \pi \mid R' \mapsto R\}$;

    $\pi := (\pi \setminus \{R\}) \cup$ *reach*$'$;

  fi

od

return there is $R \in$ *reach* **such that** $R \subseteq R_{bad}$;



Lemma

The procedure returns TRUE iff $I \mapsto^* R_{bad}$.

- If the regions R_{bad} 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.