Reachability analysis for hybrid automata & introduction to abstraction in MC

Martin Fränzle

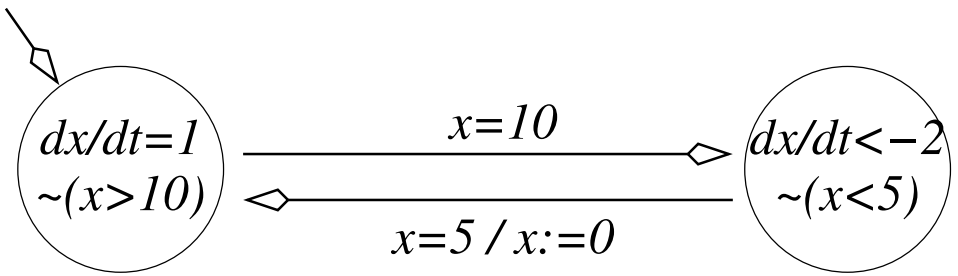
Carl von Ossietzky Universität

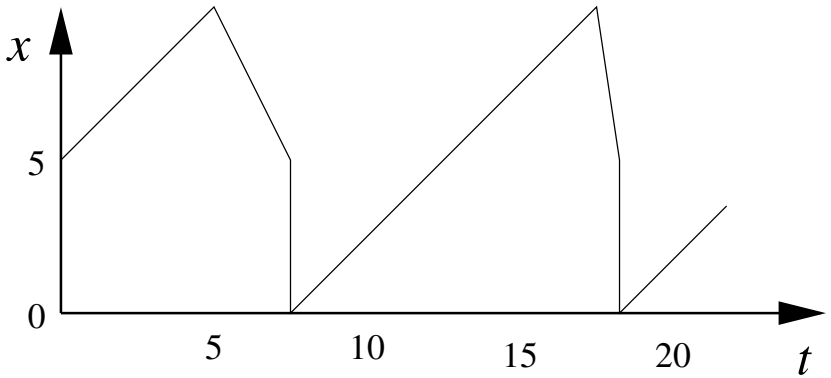
Dpt. of Computing Science

Res. Grp. Hybrid Systems

Oldenburg, Germany

Hybrid automata





Hybrid features are used for

1. formalizing interaction of discrete & continuous components,
2. over-approximating complex differential equations by simple differential inclusions.

Hybrid Automaton

Def: a hybrid automaton H is a tuple $H = (V, X, f, Init, Inv, Jump)$, where :

- V is a *finite* set of discrete modes.
  The elements of V represent the discrete states.
- $X = \{x_1, \ldots, x_n\}$ is an (ordered) finite set of continuous variables.
  A real-valued valuation $z \in \mathbb{R}^n$ of $x_1, \ldots, x_n$ represent a continuous state.
- $f \in V \times \mathbb{R}^n \to \mathbb{R}^n$ assigns a vector field to each mode.
  The dynamics in mode m is $\frac{d\vec{x}}{dt} = f(m, \vec{x})$.
- $Init \subseteq V \times \mathbb{R}^n$ is the initial condition.
  $Init$ defines the admissible initial states of H.
- $Inv \subseteq V \times \mathbb{R}^n$ specifies the mode invariants.
  Inv defines the admissible states of H.
- $Jump \in V \times \mathbb{R}^n \to \mathcal{P}(V \times \mathbb{R}^n)$ is the jump relation.
  $Jump$ defines the possible discrete actions of H. The jump relation may be non-deterministic and entails both discrete modes and continuous variables.

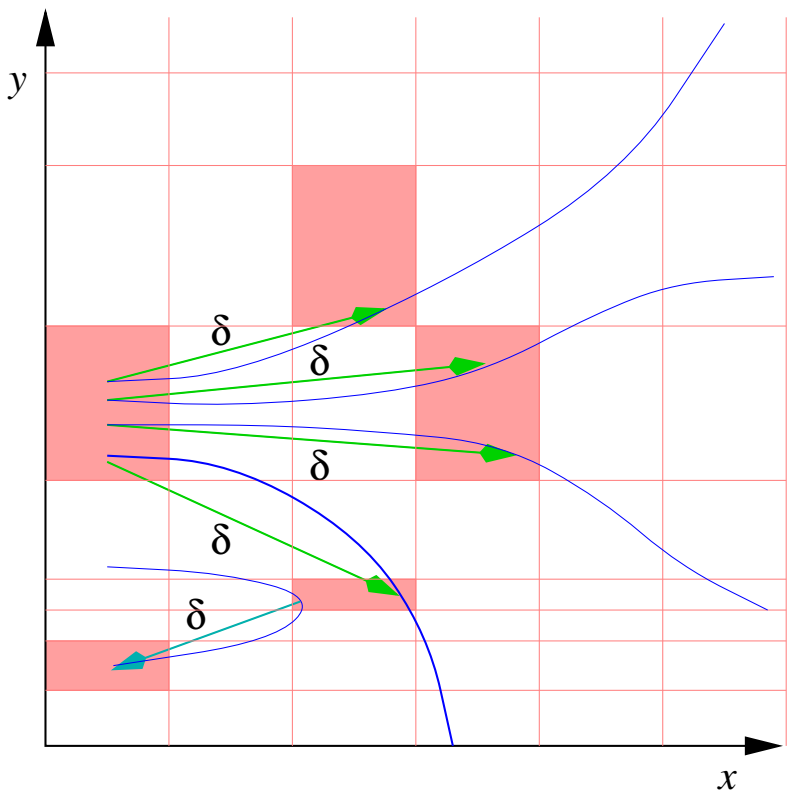
Model-checking through discretization

Idea:

Hybrid automata are mapped to finite state through overapproximation, then subjected to finite-state symbolic model-checking

Problems:

- inexact (“false paths”)
- find *appropriate discretization* (“false negatives”)



Checking safety

...in a finite Kripke structure:

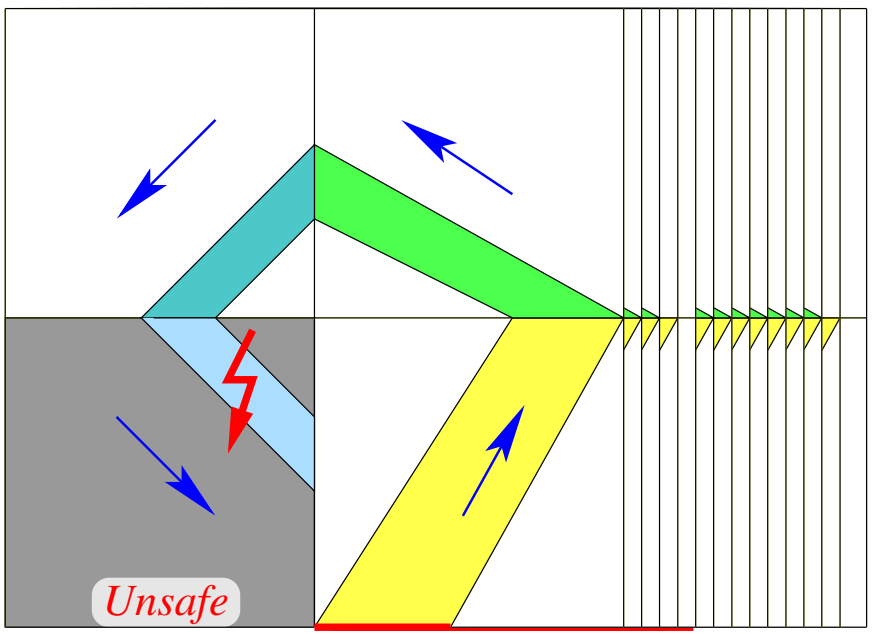
1. For increasing n, calculate the set $\mathit{Reach}^{\leq n}$ of states reachable in at most n steps.
2. Chain $\mathit{Reach}^{\leq 1} \subseteq \mathit{Reach}^{\leq 2} \subseteq \ldots$ has only a finite ascending subchain due to finiteness of state-space.

$\Rightarrow$ Set $\bigcup_{n \in \mathbb{N}} \mathit{Reach}^{\leq n}$ of reachable states can be constructed in finitely many steps.

3. Check for intersection with set of unsafe states.

...in a hybrid automaton:

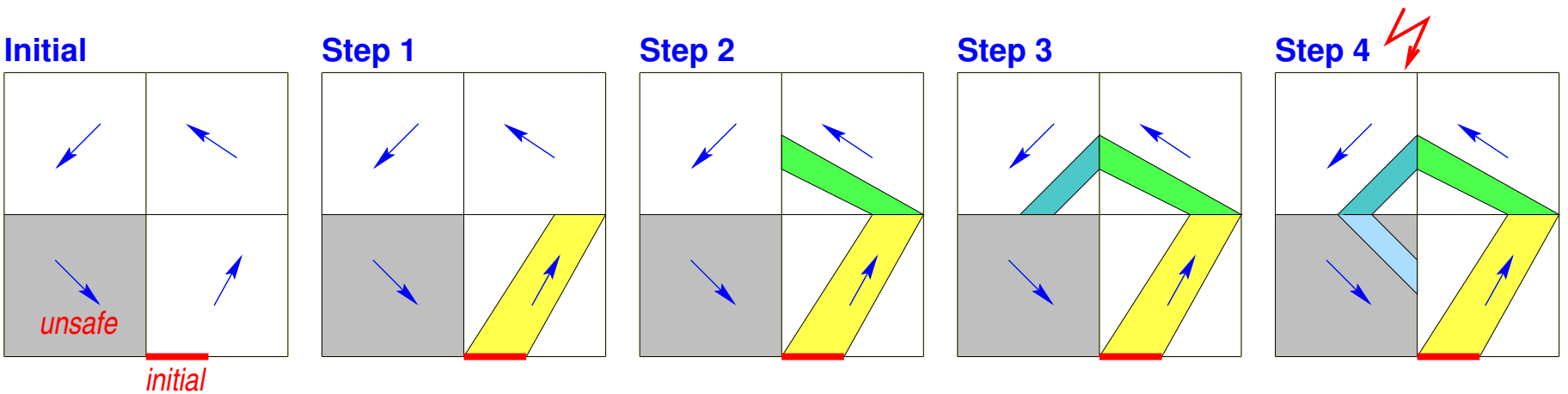
Similar fixpoint construction



need not terminate, but yields an effective procedure for falsification.

Making the idea operational: the ingredients

Idea: Iterate transition relation and continuous dynamics until an unsafe state is hit:



Result: Terminates iff HA is unsafe.

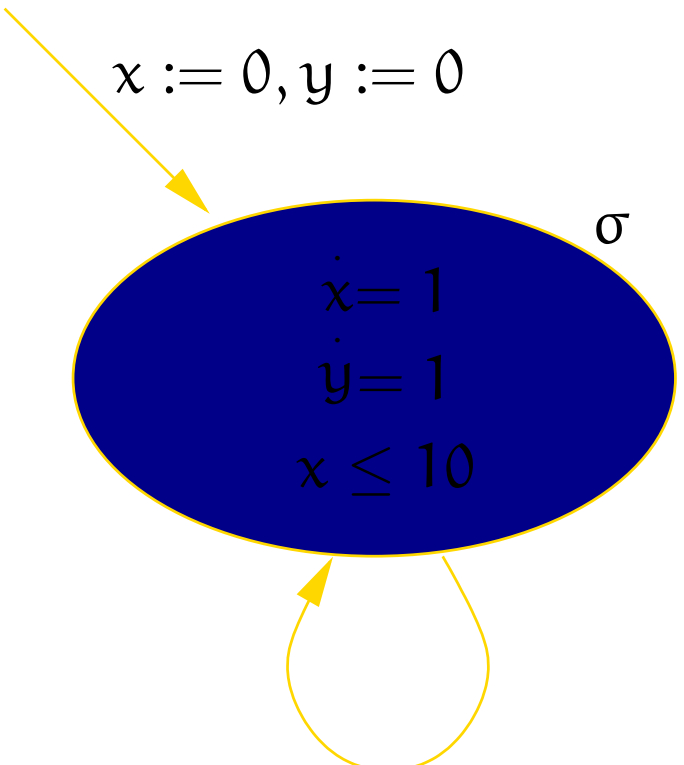
Requires: Effective representations of transition relation, continuous dynamics, and initial, intermediate, and unsafe state sets s.t.

1. Calculation of the state set reachable within $n \in \mathbb{N}$ steps is effective,
2. Emptiness of intersection of unsafe state set with the state set reachable in n steps is decidable.

(implemented in e.g. HyTech [Henzinger, Ho, Wong-Toi, 1995–])

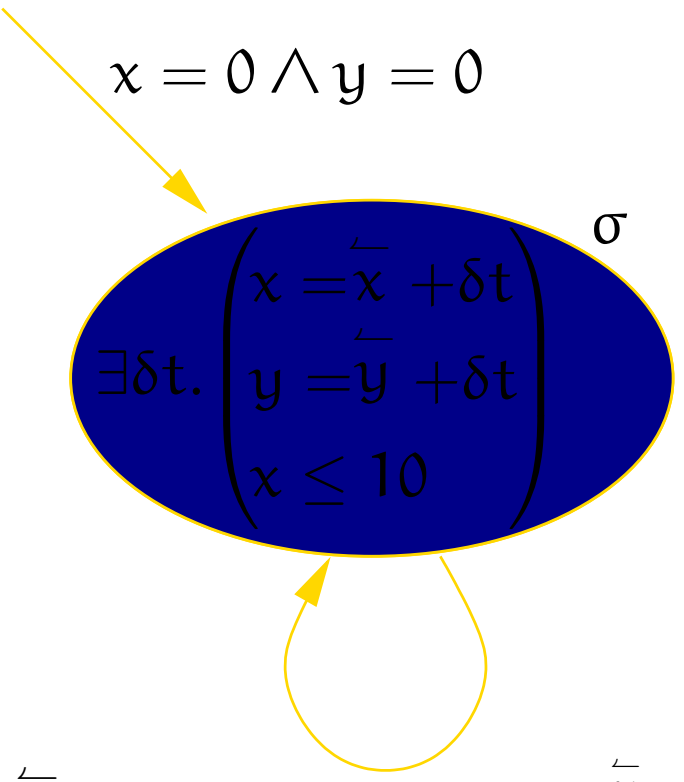
From hybrid automata to logic

A:



$x = 10 \rightarrow x := 0, y := \frac{y}{2} - 1$

A:



$\overleftarrow{x} = 10 \wedge x = 0 \wedge y = \frac{\overleftarrow{y}}{2} - 1$

Convexity of behaviors required, continuity is not FO-expressible!

Essentials of polynomial HA

- Finite set Σ of discrete states, finite vector $\vec{x}$ of cont. variables
- An activity predicate $act_\sigma \in \mathrm{FOL}(\mathbb{R}, =, +, \times)$ defines the possible continuous evolution while residing in discrete state σ
- A transition predicate $trans_{\sigma \to \sigma'} \in \mathrm{FOL}(\mathbb{R}, =, +, \times)$ defines guard and effect of transition from discrete state σ to discrete state σ'
- An initiation predicate $initial_\sigma \in \mathrm{FOL}(\mathbb{R}, =, +, \times)$ defines the discrete/continuous state pairs the system can start in
- A path is a sequence $\langle (\sigma_0, \vec{y}_0), (\sigma_1, \vec{y}_1), \ldots \rangle \in (\Sigma \times \mathbb{R}^d)^{\star|\omega}$ entailing an alternation of transitions and activities:
  - $(\overleftarrow{\vec{x}} := \vec{y}_i, \vec{x} := \vec{y}_{i+1}) \models trans_{\sigma_i \to \sigma_{i+1}}$ if i is odd
  - $(\overleftarrow{\vec{x}} := \vec{y}_i, \vec{x} := \vec{y}_{i+1}) \models act_{\sigma_i}$ and $\sigma_i = \sigma_{i+1}$ if i is even
  - $(\vec{x} := \vec{y}_0) \models initial_{\sigma_0}$

Decidability of $\mathrm{FOL}(\mathbb{R}, =, +, \times)$ yields decision procedures for temporal properties of paths of *finitely fixed length*

Reachability

of a final discrete state σ' from an initial discrete state σ and through an execution containing n transitions can be formalized through the inductively defined predicate $\phi^{n}_{\sigma\to\sigma'}$, where

$$\phi^{0}_{\sigma\to\sigma'} = \begin{cases} \text{false}\,, \text{ if } \sigma \neq \sigma' \,, \\ \mathit{act}_{\sigma}\,, \text{ if } \sigma = \sigma' \,, \end{cases}$$

$$\phi^{n+1}_{\sigma\to\sigma'} = \bigvee_{\tilde{\sigma}\in\Sigma} \exists\, \vec{x}_1, \vec{x}_2 \,.\, \begin{pmatrix} \phi^{n}_{\sigma\to\tilde{\sigma}}[\vec{x}_1/\vec{x}] \wedge \\ \mathit{trans}_{\tilde{\sigma}\to\sigma'}[\vec{x}_1, \vec{x}_2/ \overleftarrow{\vec{x}}, \vec{x}] \wedge \\ \mathit{act}_{\sigma'}[\vec{x}_2/ \overleftarrow{\vec{x}}] \end{pmatrix}$$

Safety of hybrid automata

$\Rightarrow$ An unsafe state is reachable within n steps iff

$$\mathit{unsafe}_n = \bigvee_{\sigma' \in \Sigma} \mathit{Reach}_{\sigma'}^{\leq n} \wedge \neg \mathit{safe}_{\sigma'}$$

is satisfiable, where

$$\mathit{Reach}_{\sigma'}^{\leq n} = \bigvee_{i \in \mathbb{N}_{\leq n}} \bigvee_{\sigma \in \Sigma} \phi^i_{\sigma \to \sigma'} \wedge \mathit{initial}_\sigma[\overleftarrow{\vec{x}} / \vec{x}]$$

characterizes the continuous states reachable in at most n steps within discrete state σ'.

$\Rightarrow$ An unsafe state is reachable iff there is some $n \in \mathbb{N}$ for which unsafe_n is satisfiable.

The semi-decision procedure

1. FOL$(\mathbb{R}, =, +, \times)$ is decidable. [Tarski 1948]
2. *unsafe*$_{\mathfrak{n}}$ is a formula of FOL$(\mathbb{R}, =, +, \times)$.

$\Rightarrow$ For arbitrary $\mathfrak{n} \in \mathbb{N}$ it is *decidable whether an unsafe state is reachable within* $\mathfrak{n}$ *steps*.

3. By successively testing increasing $\mathfrak{n}$, this yields a *semi-decision procedure for reachability of unsafe states*:
   (a) Select some $\mathfrak{n} \in \mathbb{N}$,
   (b) check *unsafe*$_{\mathfrak{n}}$.
   (c) If this yields true then an unsafe state is reachable. Report this and terminate.
   (d) Otherwise select strictly larger $\mathfrak{n} \in \mathbb{N}$ and redo from step (b).

The semi-decision procedure — contd.

Note that in general the semi-decision procedure can only detect being unsafe, yet does not terminate iff the HA is safe. Hence, it

can be used for falsifying HA,

but not for verifying them.

However, there are cases where $\mathit{Reach}^{\leq n+1}_{\sigma'} \Rightarrow \mathit{Reach}^{\leq n}_{\sigma'}$ holds for some $n \in \mathbb{N}$ s.t. the reachable state set can be calculated in a finite number of steps.

Appropriate back-end decision procedures

Type of analysis	Falsification only	Falsific. + verification if reach set stabilizes
Property to be checked	unsafe state reachable	unsafe state reachable + reach set stabilization
Formal property to be checked	exists $n \in \mathbb{N}$ s.t. $\bigvee_\sigma \mathit{Reach}_\sigma^{\leq n} \wedge \neg \mathit{safe}_\sigma$ is satisfiable	dito + $\bigwedge_{\sigma'} \mathit{Reach}_{\sigma'}^{\leq n+1} \Rightarrow \mathit{Reach}_{\sigma'}^{\leq n}$ is valid
Formula class	existential FOL$(\mathbb{R}, =, +, \times)$	FOL$(\mathbb{R}, =, +, \times)$ with one quantifier alternation

Abstraction in Modeling & Model Chcecking

An insight

- What you model in model-based analysis and design need not be an exact image of the system you have or anticipate to build.
- It may well be an *approximation*,
- provided validity of properties evaluated on the approximation wrt. the real system can be guaranteed.

Motivation

- High-level modeling can provide an executable prototype of the ES and its environment.
  But what if there is no computable representation of the environment dynamics?
- High-level modeling supports automatic verification through model checking and automatic test case generation.
  But what if the model checking /test generation problem
  - is computationally intractable due to very large state spaces?
  - is not mechanizable due to undecidability, e.g. due to undecidable arithmetic problems involved?

Can we still take advantage of the above-mentioned mechanic procedures?

What you'll learn

1. That abstraction can solve the problem
   - can reduce very large state spaces while still allowing to safely deduce properties that do also hold for the full system
   - can provide decidable problem representation whose validity implies validity of the original problem (soundness) but not necessarily vice versa (incompleteness)
2. The formal notion of abstraction of a Kripke structure
3. The relation between abstraction types and properties preserved

Abstraction: basic idea

- Up to now, we assumed the model to be a faithful representation of our system (ES and its environment).
- Now, we are prepared to eliminate or modify detail that is irrelevant to our problem
  - eliminate state variables
  - eliminate difficult operations on variables
- We are willing to accept a loss of precision iff we have clear criteria for whether an analysis result is an artifact of the abstraction or not

Basic forms of state abstraction

1. Remove detail about the update of (some) state variables
   - size of state space remains unchanged!
   - yet still helpful:
     - undecidable arithmetic problems may disappear because of removal of the corresponding updates,
     - descriptional complexity of the model may be reduced substantially.
2. Replace the domains of some state variables (or even the whole state space) by smaller domains
   - size of state space shrinks:
     - usually yields speedup of state-exploratory methods (model checking, test vector generation, ...)
     - can replace infinite state spaces by finite ones
     - can replace undecidable infinite-state problems by decidable infinite-state problems (e.g., hybrid automata $\rightarrow$ timed automata)

Example 1: Replacing operations

- Replace all occurrences of variable $x \in \mathbb{Z}$ in terms by a non-deterministic value:

$$y := abs(x) \quad \rightsquigarrow \quad y := \text{ one of } abs(\mathbb{Z})$$

N.B. This does essentially remove state variable x.

- or replace all assignments to $x \in \mathbb{Z}$ by a fully non-deterministic assignment:

$$x := abs(y) \quad \rightsquigarrow \quad x := \text{ one of } \mathbb{Z}$$

Both versions: Original program halts if replacement halts, but not necessarily vice versa.

Example 2: From infinite state to finite state

- Replace state variables $x \in \mathbb{Z}$ by $\tilde{x} \in \{+, 0, -\}$.
- Replace updates of x in program/model by updates of $\tilde{x}$ s.t.

$$\begin{array}{lll} x > 0 & \text{if} & \tilde{x} = + \\ x = 0 & \text{if} & \tilde{x} = 0 \\ x < 0 & \text{if} & \tilde{x} = - \end{array}$$

  holds for corresponding places in a run
- Thus, replace
  - $x := 3$ by $\tilde{x} := +$,
  - $x := 0$ by $\tilde{x} := 0$,
  - $y := abs(x)$ by $\tilde{y} :=$ if $\tilde{x} = 0$ then 0 else $+$
  - $x := x - 1$ by $\tilde{y} :=$ if $\tilde{x} \in \{-, 0\}$ then $-$ else one of $\{0, +\}$

- this replaces a deterministic program by a non-deterministic one,
- replaces an infinite-state program by a program over finite state,
- replaces an undecidable halting problem by a decidable one
  - halting of a program operating over integers is in general undecidable,
  - halting of non-deterministic programs operating over finitely many variables ranging over finite domain is decidable.

What is the relation between the two halting problems?

Over- & Underapproximation

State-Space-Preserving Case

Overapproximation

Def.: Given a Kripke structure $K = (V, E, L, I)$, a *state-preserving* overapproximation is a Kripke structure $K' = (V, E', L, I)$ with $E' \supset E$.

Lemma: All paths of K are also paths of K'.

Cor.: If K' satisfies an LTL formula ϕ then so does K.

Def.: $\forall$CTL denotes the set of CTL formulae where universal path quantifiers occur in positive context only and existential path quantifiers occur in negative context only (i.e., the negation normal form contains only universal path quantifiers).

Cor.: If K' satisfies an $\forall$CTL formula ϕ then so does K.

The reverse implications of the corollaries are i.g. not true!

Error paths of K' which are not error paths of K are called spurious counterexamples.

Underapproximation

Def.: Given Kripke structures K and K', K' is a *state-preserving* underapproximation of K iff K is a state-preserving overapproximation of K'.

Lemma: All paths of K' are also paths of K.

Cor.: Only if K' satisfies an LTL formula ϕ then so does K.

Def.: $\exists$CTL denotes the set of CTL formulae where universal path quantifiers occur in negative context only and existential path quantifiers occur in positive context only (i.e., the negation normal form contains only existential path quantifiers).

Cor.: If K' satisfies an $\exists$CTL formula ϕ then so does K.

The reverse implications of the corollaries are i.g. not true!

Can it help?

- Direct approximation does not change the state space, hence, no size benefit.
- Yet it may help, in particular with symbolic MC:
  - Symbolic state-exploration problem may be rdeuced to decidable fragments of the underlying logics due to removal of “undecidable operations”.
  - Predicative encodings of state sets can become more compact.
  - Extreme case: If $\mathtt{E}'$ is the universal relation then $\mathtt{AX}\phi$ is either satisfied by all states or by no state of $\mathtt{K}'$, thus yielding trivial predicative representations.

Simulations

Approximations between different state spaces

Simulations

Def.: Given two Kripke structure $K = (V, E, L, I)$ and $K' = (V', E', L', I')$ and a relation $R \subseteq V \times V'$, we call R a simulation relation between K and K' iff $(s, s') \in R$ implies

1. $L(s) = L'(s')$,
2. $\forall t \in V \bullet [(s,t) \in E \implies \exists t' \in V' \bullet (s',t') \in E' \wedge (t,t') \in R]$,

and

3. $\forall s \in I \bullet \exists s' \in I' \bullet (s,s') \in R$.

Def.: We say that K' simulates K, denoted $K \preceq K'$ iff there exists a simulation relation between K and K'.

Lemma: If $K \preceq K'$ then for each *anchored* path π of K (i.e., for each path with $\pi_0 \in I$) there exists an anchored path π' of K' s.t. $L(\pi) = L'(\pi')$.

Cor.: If K' satisfies an LTL or $\forall$CTL formula ϕ then so does K.

The reverse implication is in general not true!

Computing Abstractions

State morphisms

Abstraction by state morphims

[Clarke e.a., 1994,2000]

Given the concrete Kripke structure $\mathrm{K} = (V, \mathrm{E}, \mathrm{L}, \mathrm{I})$,

1. select an abstract state space S',
2. define a total abstraction function $\mathrm{h} : V \to V'$ (state morphism)
   - if $\mathrm{h}(s) = \mathrm{h}(t)$ then the abstraction doesn't distinguish s and t,
   - $\mathrm{h}(s) = \mathrm{h}(t)$ has to imply $\mathrm{L}(s) = \mathrm{L}(t)$,
   - induces a partition on V of cardinality $|\mathrm{codom}\,(\mathrm{h})|$,
3. compute the abstract transitions E' and the abstract labeling function L'.

Computing E': exist. and univ. abstraction

Existential abstraction: *Existence* of a transition / initial state in the pre-image sufficient for generating the image under h:

1. $E' = \{(s', t') \in V' \times V' \mid \exists (s, t) \in E \bullet h(s) = s' \wedge h(t) = t'\}$,
2. $I' = \{s' \in V' \mid \exists s \in I \bullet h(s) = s'\}$,
3. $L' = L \circ h^{-1}$.

Lemma: $h^{\exists}(K) = (\mathrm{codom}\,(h), E', L', I')$ simulates K.

Universal abstraction: *All* pre-images of the source state have to have a matching transition:

1. $E'' = \{(s', t') \in V' \times V' \mid \forall s \in h^{-1}(s') \exists t \in h^{-1}(t') \bullet (s, t) \in E\}$,
2. ...
3. ...

Lemma: K simulates $h^{\forall}(K) = (\mathrm{codom}\,(h), E'', L', I')$.

Projection

- Usually, the concrete state space is a Cartesian product $V = \Pi_1^n V_i$ spanned by multiple state variables.
- Then, any projection, e.g. $h(v_1, \ldots, v_n) = (v_3, v_4, v_7, v_9, \ldots, v_n)$ is a state homomorphism
- provided $L(s) = L(s')$ for all s, s' with $h(s) = h(s')$!

This is the most frequently used type of abstraction function!
It is often applied despite violation of the labeling cond.!

Soundness

Observation: Labels not referenced by the formula don't affect its satisfaction.

Conseq.: K' need not actually simulate K in order to have preservation $(\mathrm{K}' \models \phi) \implies (\mathrm{K} \models \phi)$ of the model property. It suffices that

$$\mathrm{K}|_{\mathrm{F}} \stackrel{\text{def.}}{=} (\mathrm{V}, \mathrm{E}, \nu \mapsto \mathrm{L}(\nu) \cap \mathrm{F}, \mathrm{I}) \preceq \mathrm{K}'|_{\mathrm{F}} \stackrel{\text{def.}}{=} (\mathrm{V}', \mathrm{E}', \nu' \mapsto \mathrm{L}'(\nu') \cap \mathrm{F}, \mathrm{I}')$$

where F is the set of atomic propositions used in ϕ.

Conseq.: If the atomic propositions are predicates over state variables (as usual) then we can safely project to any superset of the variables referenced in the atomic propositions in ϕ, thus removing arbitrarily many unreferenced variables:

- $(\mathrm{K}' \models \phi) \implies (\mathrm{K} \models \phi)$ holds, i.e. no false positives.
- False negatives may, however, arise.

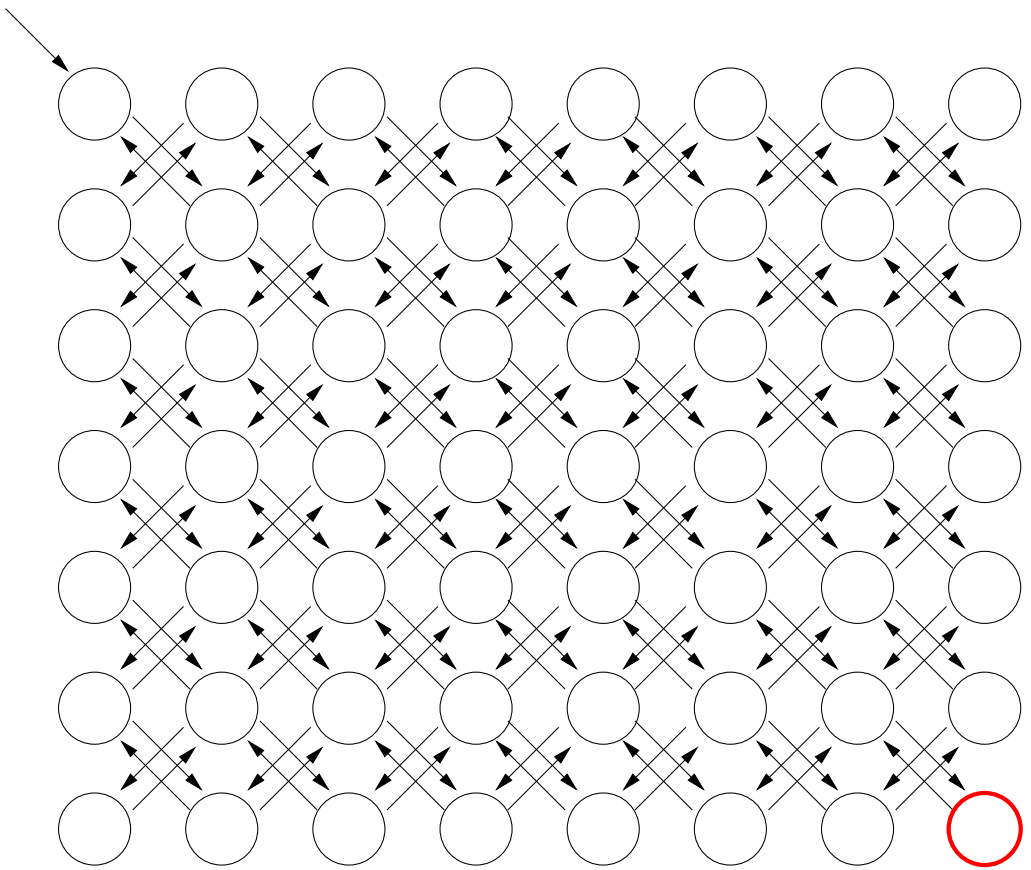
Combinations

Many practical abstractions can be understood as

1. first pursuing a homomorphic existential abstraction wrt. a partial inverse of a projection, i.e. a function $\mathrm{h} : V \to V'$ such that there is a projection $\mathrm{p} : V' \to V$ with $\mathrm{p} \circ \mathrm{h} = \mathrm{id}$
   - introduces additional state components due to inverse projection,
   - provides a simulation of the original Kripke structure due to existential abstraction
     — despite the larger state space delivered;
2. then performing a homomorphic existential abstraction wrt. a projection
   - removes part of the state space (if done reasonably then not of the freshly introduced one),
   - provides a simulation of the previous and thus of the original Kripke structure due to existential abstraction.

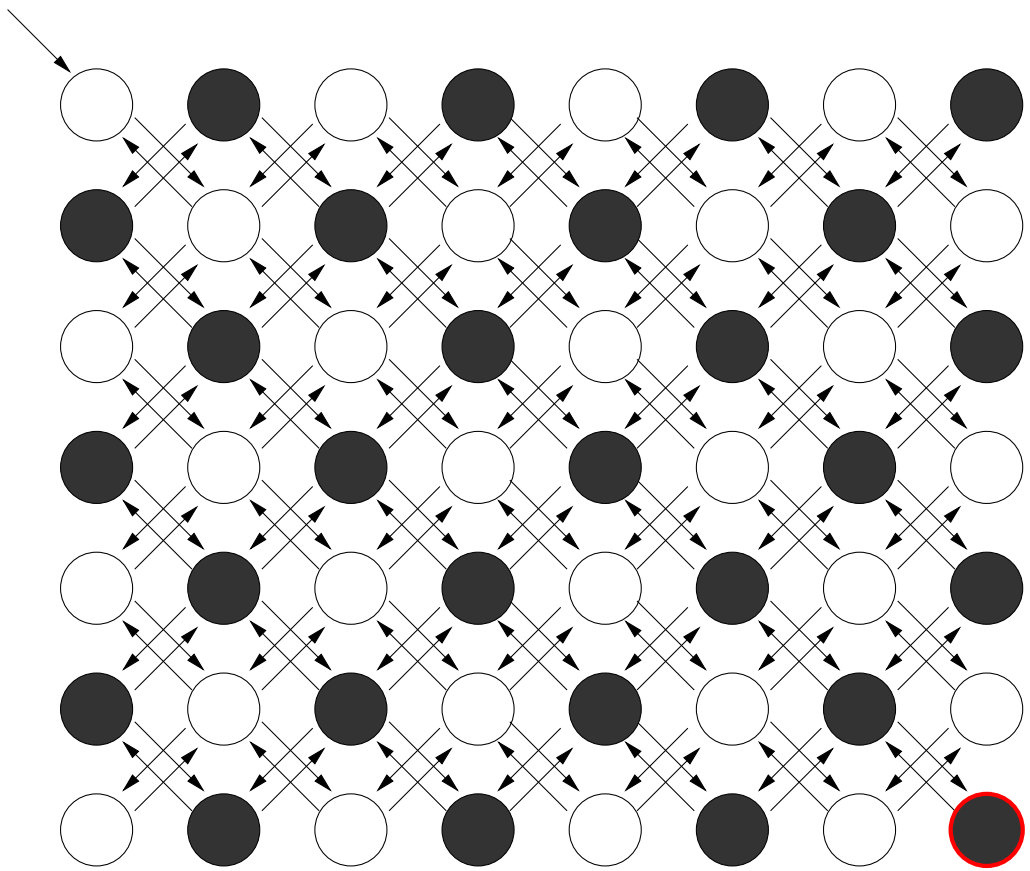
Example

$$
\begin{aligned}
V &= \{0,\dots,7\}^2 \qquad I = \{(0,0)\} \\
E &= \left\{ ((x,y),(x',y')) \;\middle|\; \begin{array}{l} \mathrm{odd}(x+y) \iff \mathrm{odd}(x'+y') \\ \wedge |x-x'| \le 1 \wedge |y-y'| \le 1 \end{array} \right\} \\
L(x,y) &= \begin{cases} \{\mathrm{red}\} & x = y = 7 \\ \emptyset & \text{otherwise.} \end{cases}
\end{aligned}
$$



Example: inverse projection

$$
\begin{aligned}
V &= \{0,\ldots,7\}^2 \\
V' &= \{0,\ldots,7\}^2 \times \{\mathrm{black}, \mathit{white}\} \\
h(x,y) &= \begin{cases} (x,y,\mathrm{black}) & \text{if } x+y \text{ is odd,} \\ (x,y,\mathit{white}) & \text{otherwise.} \end{cases}
\end{aligned}
$$



Example: projection

$$
\begin{aligned}
V' &= \{0,\ldots,7\}^2 \times \{\text{black}, \text{white}\} \\
V'' &= \{\text{black}, \text{white}\} \\
h(x,y,c) &= c
\end{aligned}
$$