Automatic Verification of Hybrid Systems

An arithmetic constraint solving perspective

Martin Fränzle[a]

together with the hybrid systems group of the

Transregional Collaborative Research Center "AVACS"[abcdef]

[a] Carl von Ossietzky Universität Oldenburg, Germany

[b] Albert-Ludwigs-Universität Freiburg, Germany

[c] Universität des Saarlandes, Saarbrücken, Germany

[d] MPII, Saarbrücken, Germany

[e] Academy of Sciences of the Czech Republic, Prague

[f] ETH Zurich, Switzerland

Apologies

Due to serious health problems last week induced by a relapse, I haven't been able to prepare and print handouts. Pls. drop me an email under

`fraenzle@informatik.uni-oldenburg.de`

and I will supply you with an electronic version asap.

Sorry for the inconvenience caused!

What is a hybrid system?

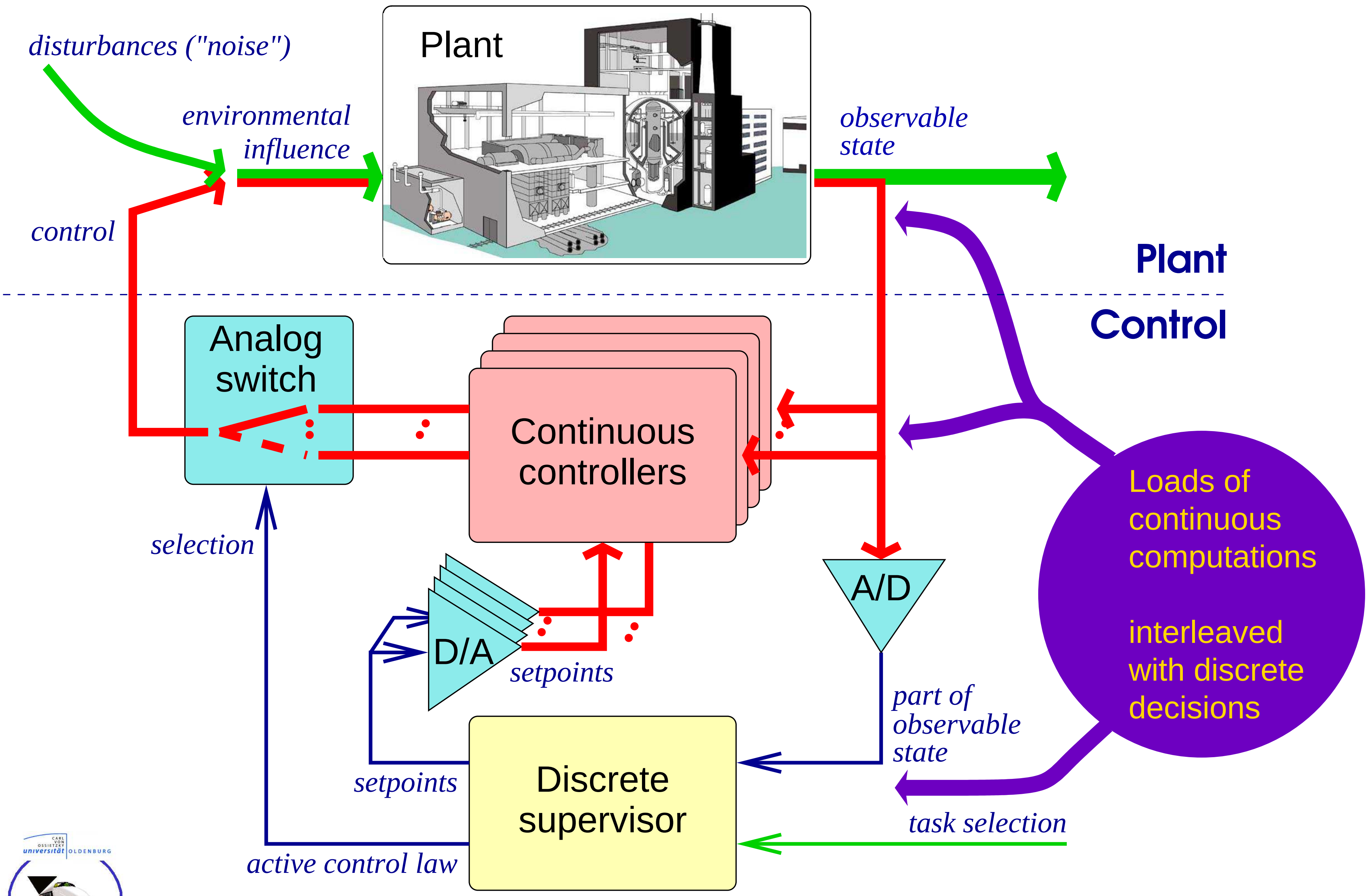
Hybrid (griech.) bedeutet überheblich, hochmütig, vermessen.
Weitere Inhalte [insbes. im wiss. Sprachgebrauch] sind später hinein interpretiert worden.

Hybrid (from Greece) means arrogant, presumptuous.
Other interpretations [in particular, in scientifi c jargon] have been added later.

After H. Menge: Griechisch/Deutsch, Langenscheidt 1984

$\Rightarrow$ when you try to verify hybrid systems,
be prepared that they may act like a prima donna!

Hybrid Systems



Hybrid systems

are ensembles of interacting discrete and continuous subsystems:

- **Technical systems:**
  - physical plant + multi-modal control
  - physical plant + embedded digital system
  - mixed-signal circuits
  - multi-objective scheduling problems (computers / distrib. energy management / traffi c managemant / ...)
- **Biological systems:**
  - Delta-Notch signaling in cell differentiation
  - Blood clotting
  - ...
- **Economy:**
  - cash/good flows + decisions
  - ...
- **Medicine/health/epidemiology:**
  - infectious diseases + vaccination strategies
  - ...

Discrete vs. continuous

A discrete system

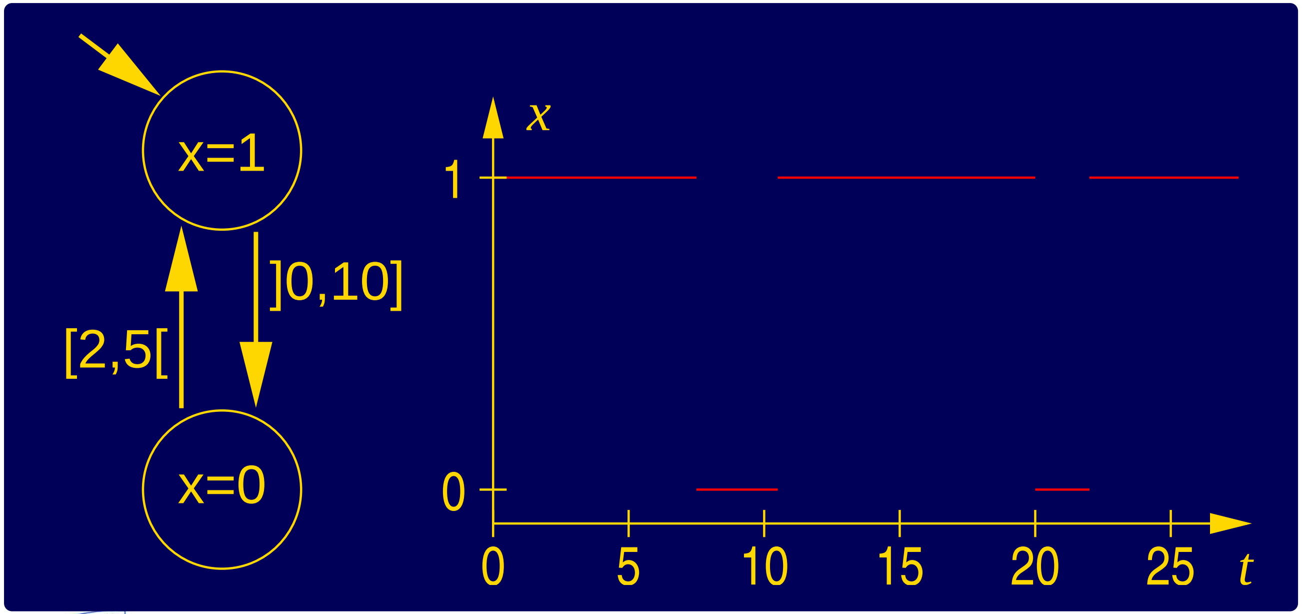
- operates on a state,
- performs discontinuous state changes at discrete time points,
- state is constant in between

E.g., a program

Prog. variables, position

Computation steps: assignments, ctrl. flow

Stable states



Validation by

- **Program verification**
- **State exploration**

Discrete vs. continuous

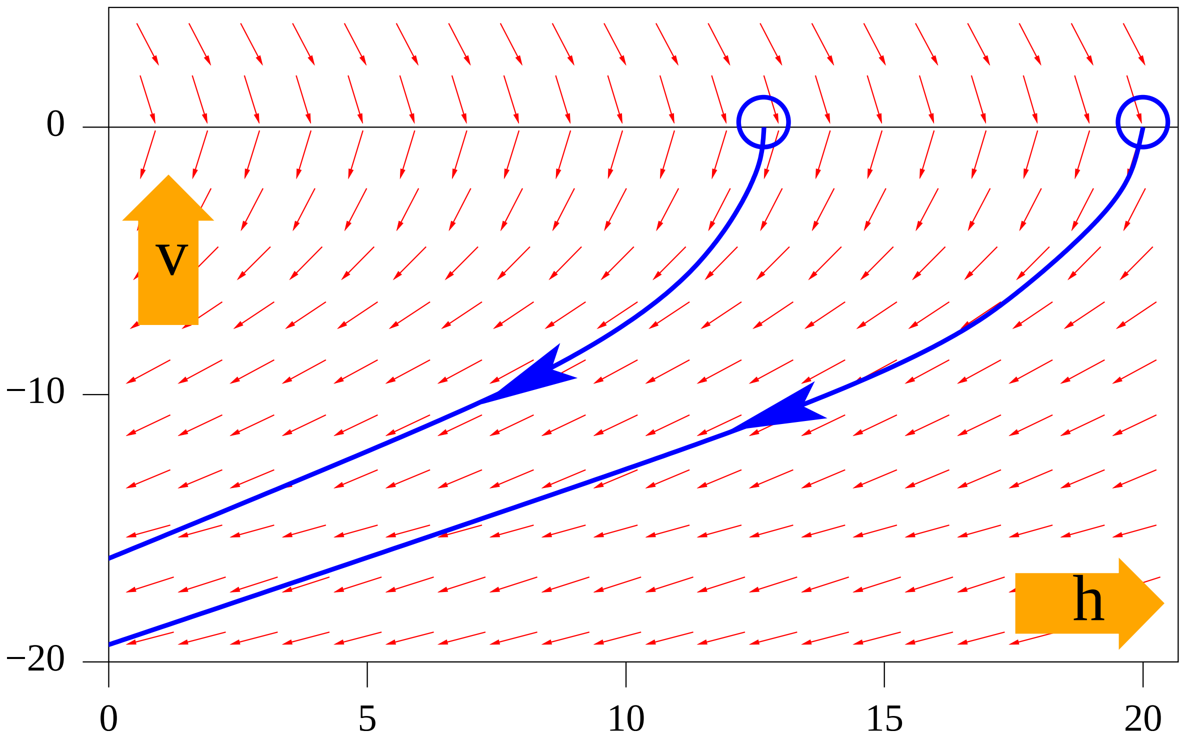
a continuous system

- operates on a continuous state,
- which evolves continuously.

E.g., a ball

Height, speed

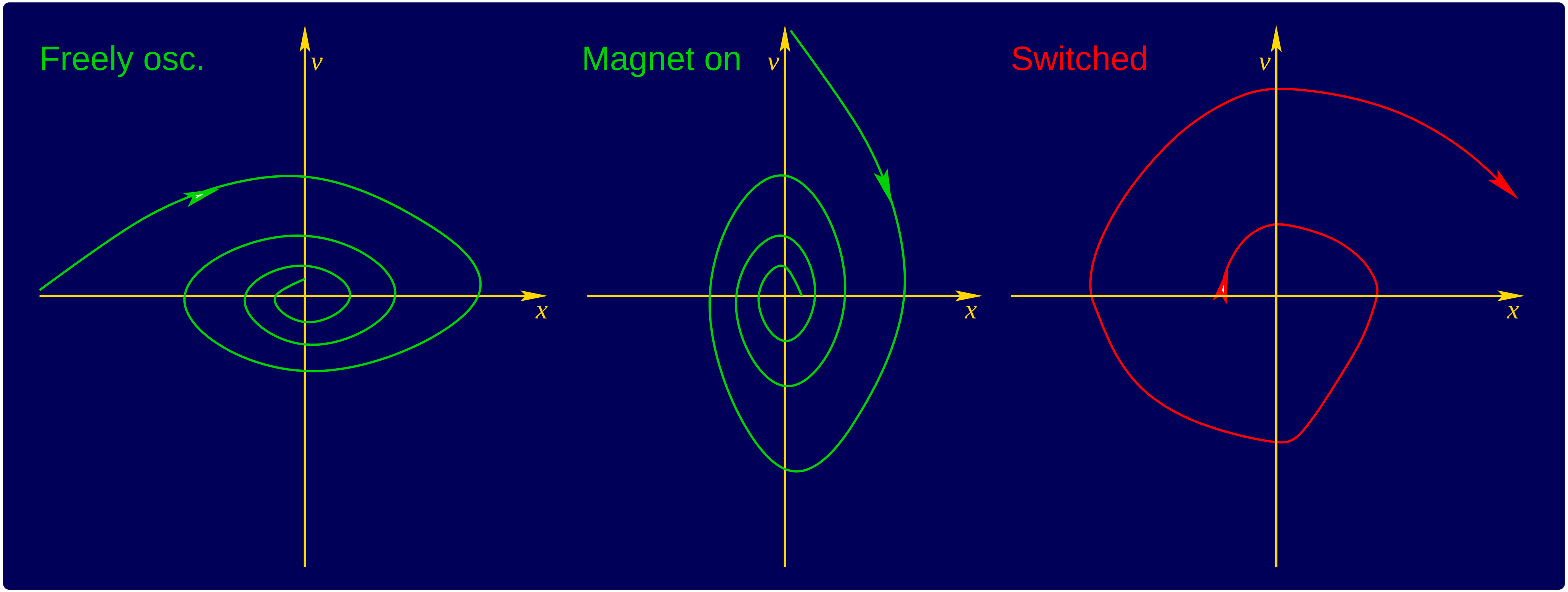
Newtonian mechanics



Validation:

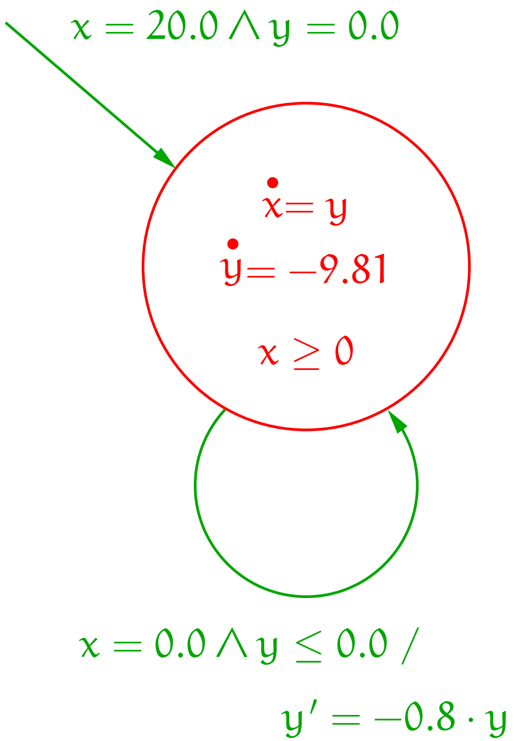
- **Analytically**
- **Simulation + continuity**

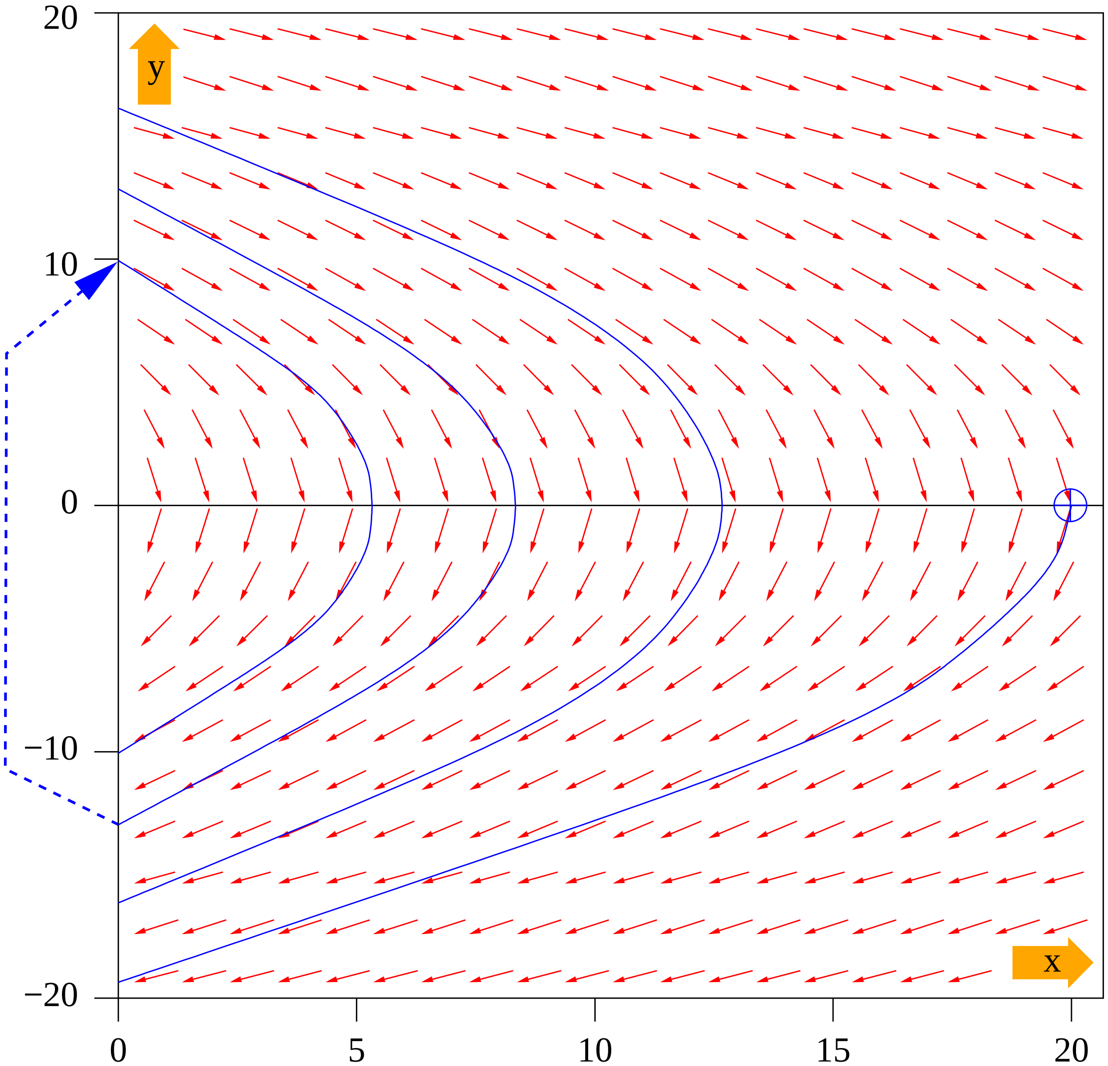
Coupled Dynamics: Forced Pendulum



Interaction of continuous dynamics and discrete mode switch destroys global convergence!

A Formal Model: Hybrid Automata



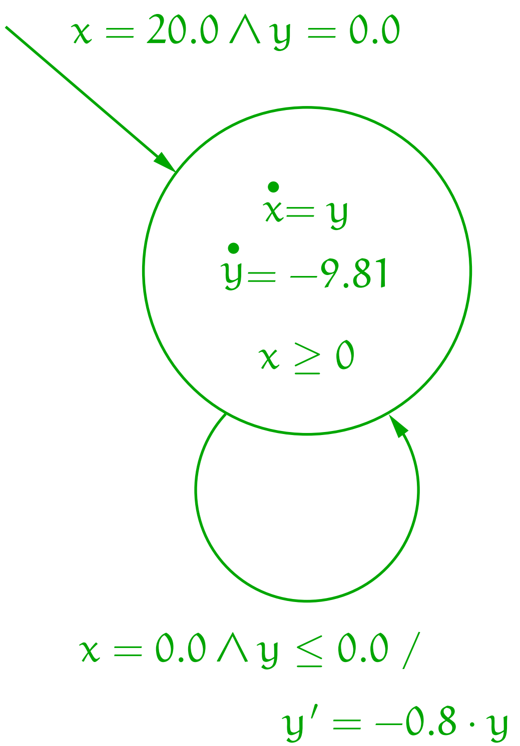


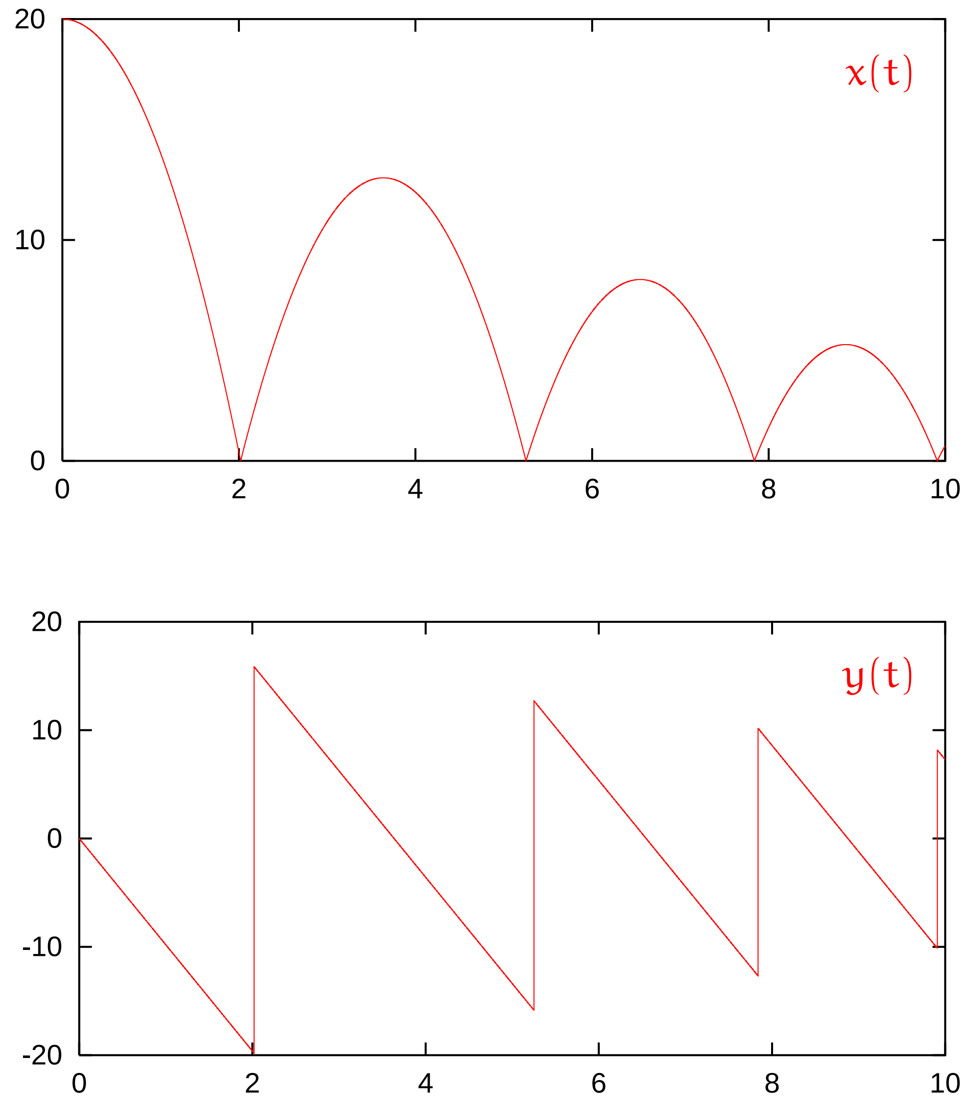
x : vertical position of the ball

y : velocity

- $y > 0$ ball is moving up
- $y < 0$ ball is moving down

A Formal Model: Hybrid Automata





x : vertical position of the ball

y : velocity

$y > 0$ ball is moving up

$y < 0$ ball is moving down

Hybrid automata

Hybrid systems	=	Coupled digital & analog systems

↓

Hybrid automata = *Finite automata* with

- *immediate transitions* that are
- *triggered by predicates on the (continuous) plant state*

+ *evolution of the continuous plant*

- *real-valued variables* governed by
- a set of (restricted) *differential equations* that are
- *selected by the current automaton state*

Hybrid Automata

The formal model

Hybrid Automaton (w/o input) [after K.H. Johansson]

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

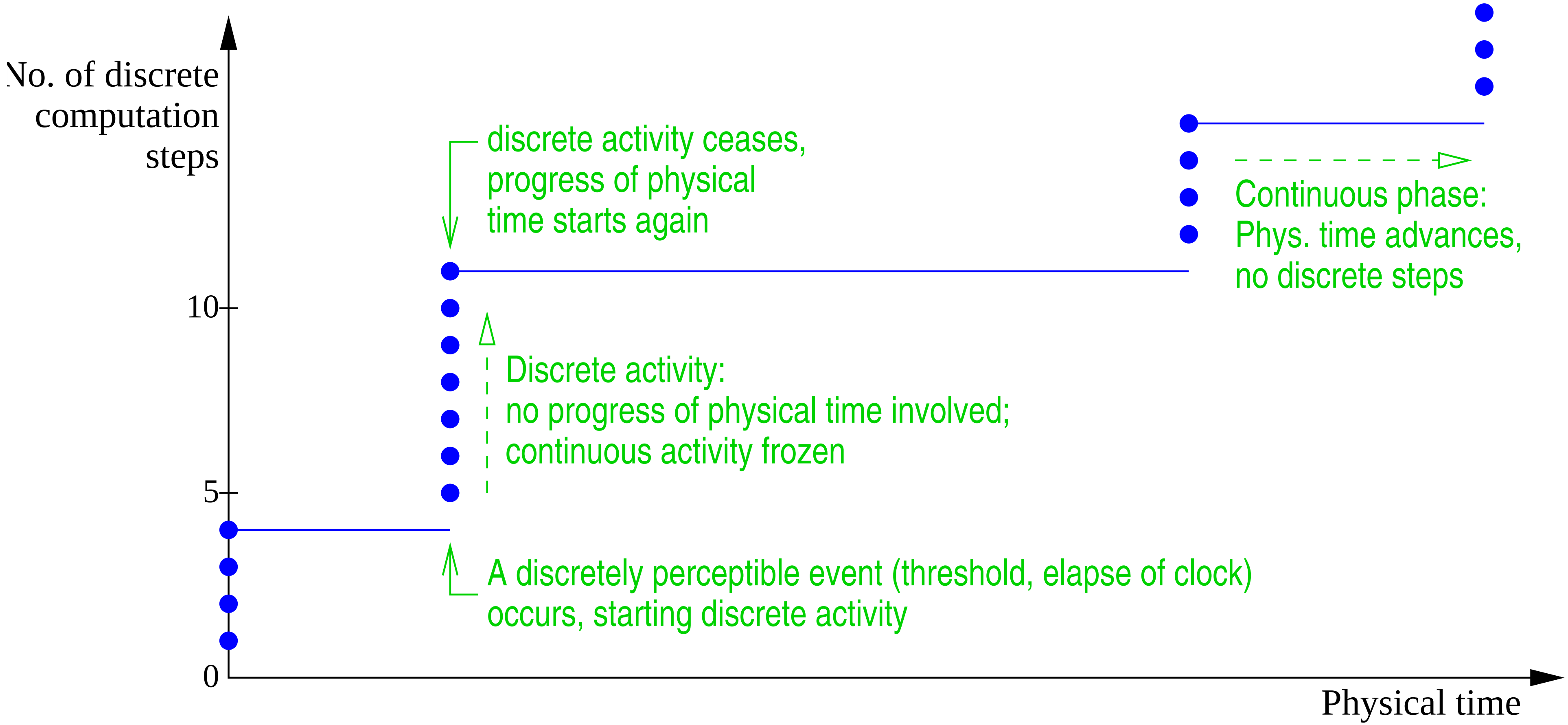
- V is a *finite* set of discrete modes.
  The elements of V represent the discrete states.
- $X = \{x_1, \ldots, x_n\}$ is an (ordered) finite set of continuous variables.
  A real-valued valuation $z \in \mathbb{R}^n$ of $x_1, \ldots, x_n$ represent a continuous state.
- $f \in V \times \mathbb{R}^n \to \mathbb{R}^n$ assigns a vector field to each mode.
  The dynamics in mode m is $\frac{dx}{dt} = f(m, \mathbf{x})$.
- $Init \subseteq V \times \mathbb{R}^n$ 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.

Generalizations

This defi nition of a HA is *not* the most general one. Obvious extensions include

- Input / disturbances in the vector fi eld.
- Labeled jumps.
- Nondeterministic continuous evolutions.
- Stochastic effects.

Semantics: Two-Dimensional Time

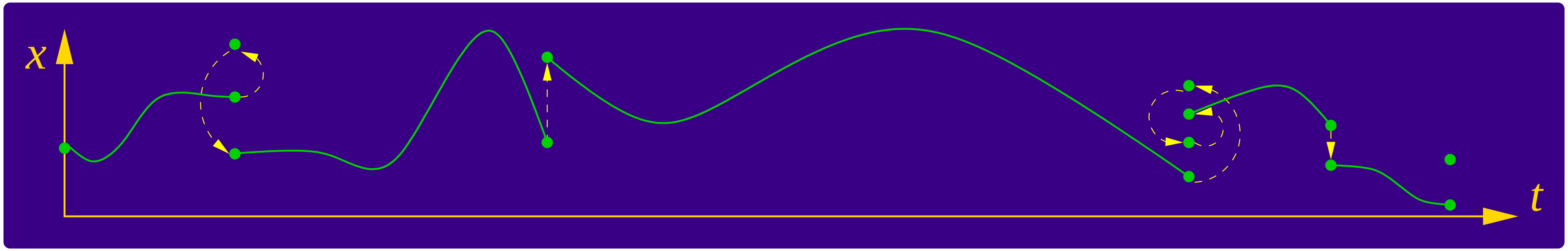


An idealization partially justifi ed by differing speeds of ES and environment!

Hybrid time [after K.H. Johansson]

Def: A hybrid time frame is a fi nite or infi nite sequence $\tau = \langle I_1, I_1, \ldots \rangle$ of time intervals I_i, where

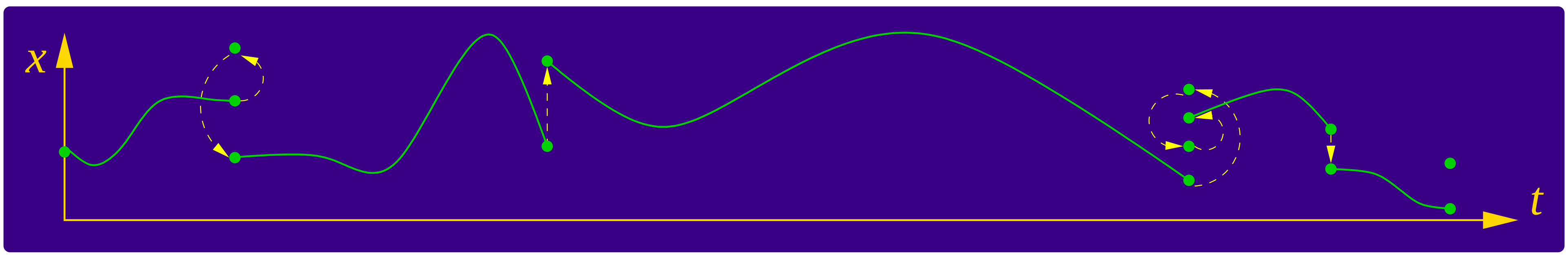
- each I_i is a non-empty convex subset of $\mathbb{R}_{\geq 0}$, i.e. a non-empty interval in $\mathbb{R}_{\geq 0}$,
- $\inf I_i \in I_i$ for each i, i.e. the intervals are left-closed,
- $\sup I_i \in I_i$ for each $i < \text{len}\ \tau$, i.e. all intervals excepts perhaps the rightmost are right-closed,
- $\max I_i = \min I_{i+1}$ for each $i < \text{len}\ \tau$, i.e. the intervals are adjacent and overlap exactely in one point.



Hybrid trajectories [after K.H. Johansson]

Def: A hybrid trajectory E is a tuple $E = (\tau, \nu, x)$ such that

- τ is a hybrid time frame,
- $\nu \in V^* \cup V^\omega$ with $\text{len}\ \nu = \text{len}\ \tau$ is a sequence of discrete modes,
- $x \in (\mathbb{R}_{\geq 0} \xrightarrow{\text{part.,cont.}} \mathbb{R}^n)^* \cup (\mathbb{R}_{\geq 0} \xrightarrow{\text{part.,cont.}} \mathbb{R}^n)^\omega$ with $\text{len}\ x = \text{len}\ \tau$ and $\text{dom}\ x_i = \tau_i$ is a sequence of continuous flows of the variables in X.

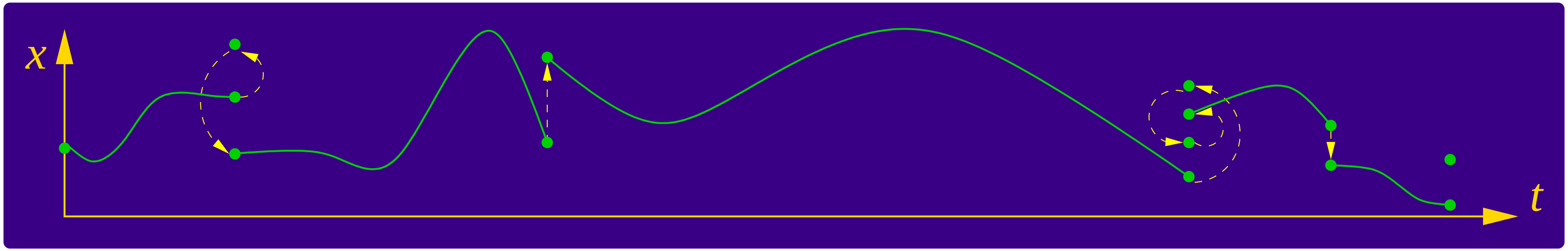


Executions of a HA [after K.H. Johansson]

Def: A run $E = (\tau, \nu, x)$ is an *execution* of the hybrid automaton $H = (V, X, f, Init, Inv, Jump)$ iff

- **Initiation:** $(\nu_1, x_1(\min \tau_1)) \in Init$,
- **Consecution:** $Jump((\nu_i, x_i(\max \tau_i)) \ni (\nu_{i+1}, x_{i+1}(\min \tau_{i+1}))$ holds for all $i < \text{len}\ \tau$,
- **Continuous evolution:** x_i is a solution of $\frac{dx}{dt} = f(\nu_i, \mathbf{x})$ for each $i \leq \text{len}\ \tau$,
- **State consistency:** $(\nu_i, x_i(t)) \in Inv$ for each $t \in \text{dom}\ \tau_i$ and each $i \leq \text{len}\ \tau$

hold.



Hybrid systems

- **Proof obligation:** Can the system be guaranteed to show desired behaviour, even under disturbances? E.g.,
  - remains in safe states?
  - eventually reaches a desired operational mode?
  - stabilizes, i.e., converges against a setpoint / stable orbit / region of phase space?

! involves co-verifi cation of controller and *continuous* environment.

State and Dimension Explosion

Number of continuous variables linear in number of cars

- Positions, speeds, accelerations,
- torque, slip, ...

Number of discrete states exponential in number of cars

- Operational modes, control modes,
- state of communication subsystem, ...

Size-dependent dynamics

- Latency in ctrl. loop depends on number of cars due to communication subsystem.
- Coupled dynamics yields long hidden chan-nels chaining signal transducers.

- Need a scalable approach
- Let’s try to achieve this through strictly symbolic methods.

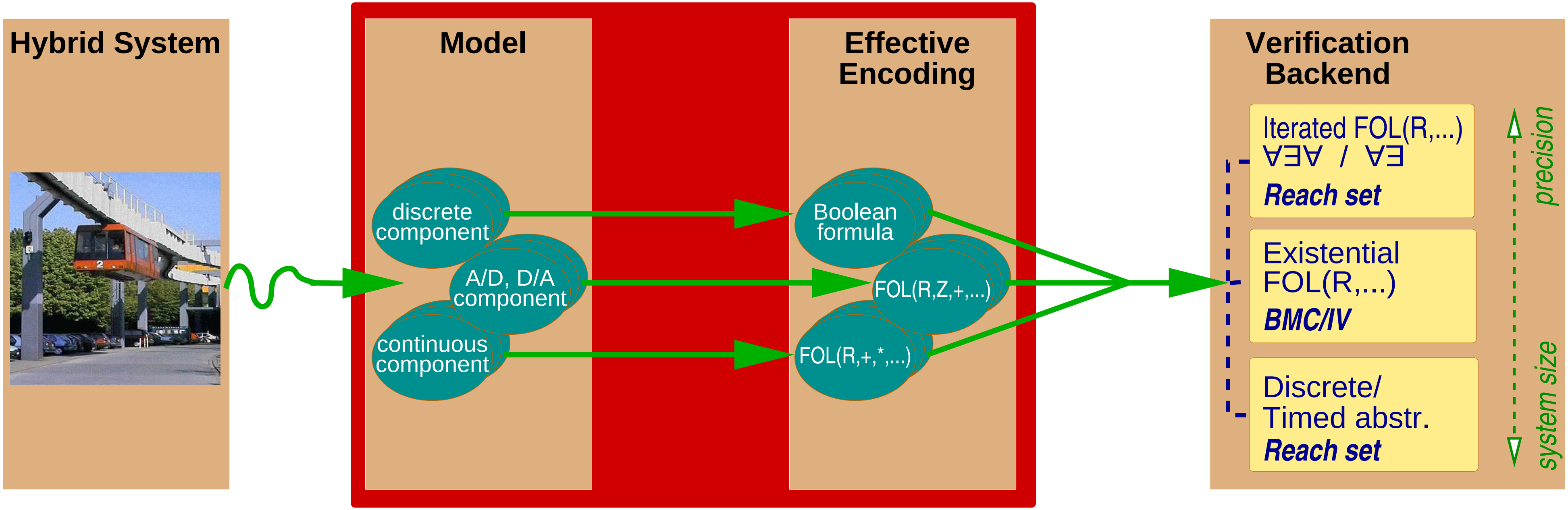
Outline

1. Translation of high-level models
   - Simulink + Stateflow
   - Compositional translation
   - based on predicative encoding of block invariants
2. Basic principles of state-exploratory analysis of HA
   - Finite-state abstraction vs. hybridisation vs. image computation of ODEs
   - iterating a FO-defi nable map
3. A sample tool set
   - SAT-modulo-theory based
   - three (increasingly experimental) levels:
     - linear hybrid automata vs. LinSAT
     - non-linear assignments
     - non-linear differential equations
   - under development in AVACS subprojects H1 and H2

Verification Frontend

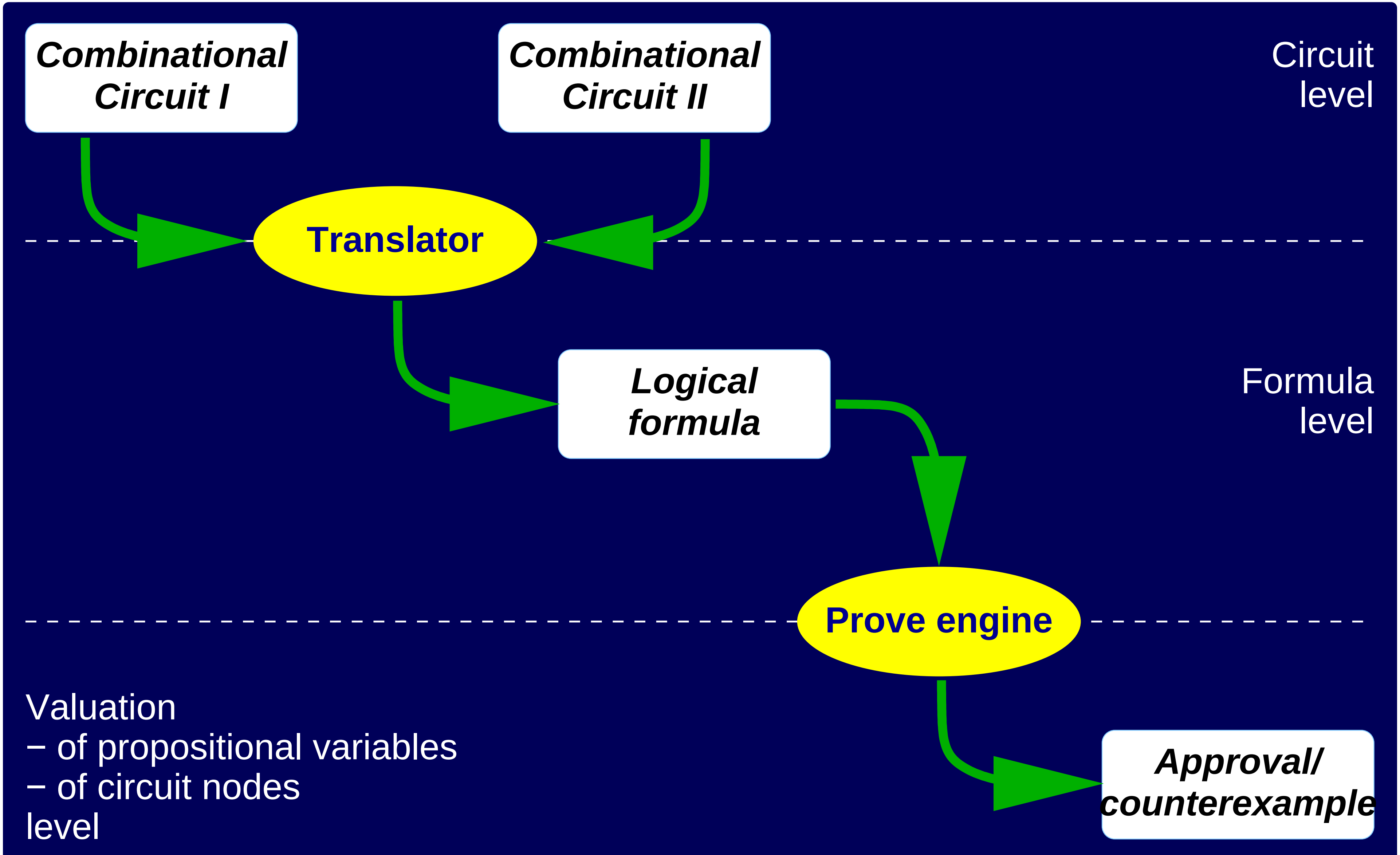
Translation of hybrid systems to arithmetic constraints

Translation



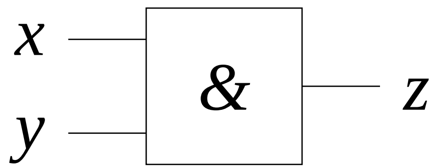
- Compositional translation into many-sorted logics

Analogy: Combinatorial Circuits

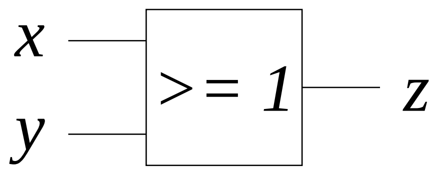


Mapping circuits to formulae

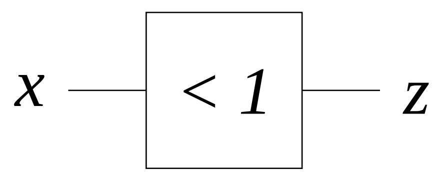
A gate is mapped to a propositional formula formalizing its invariant:



$\mapsto \quad x \wedge y \Leftrightarrow z$



$\mapsto \quad x \vee y \Leftrightarrow z$

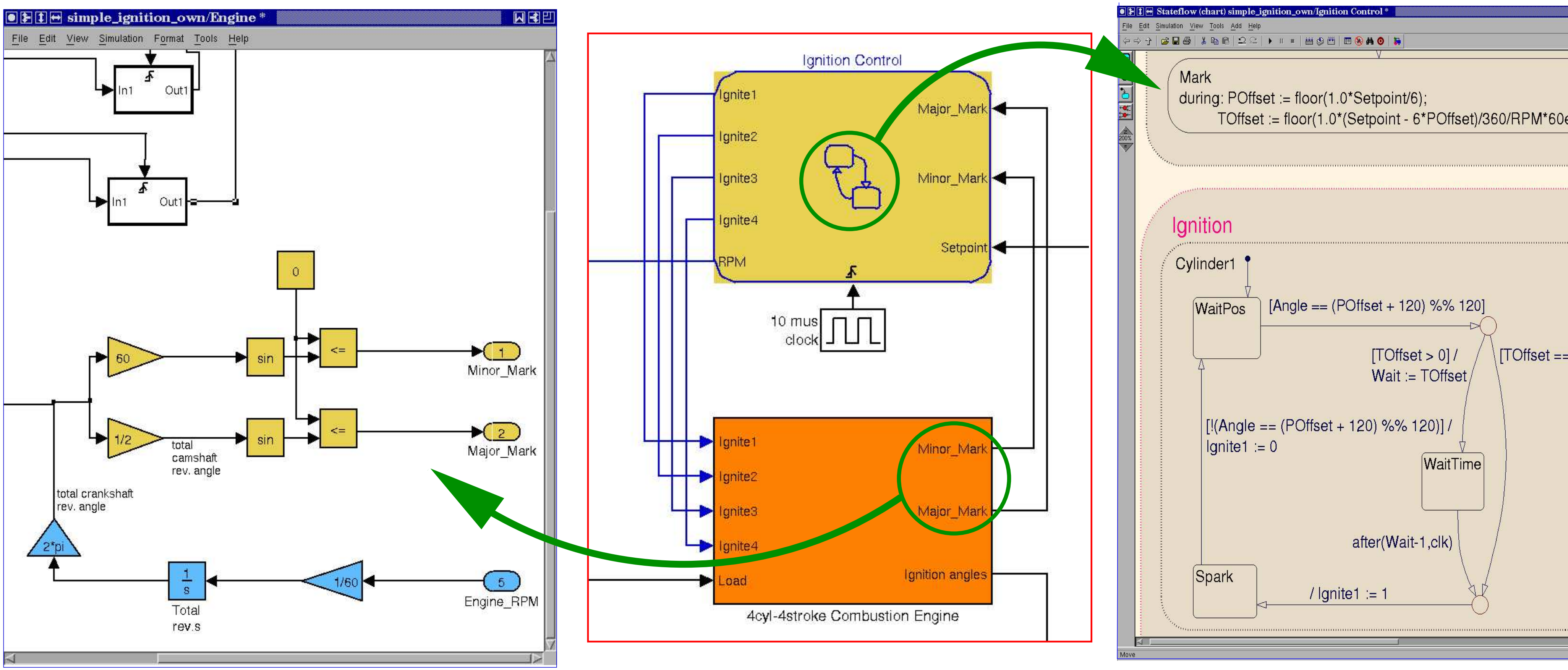


$\mapsto \quad \neg x \Leftrightarrow z$

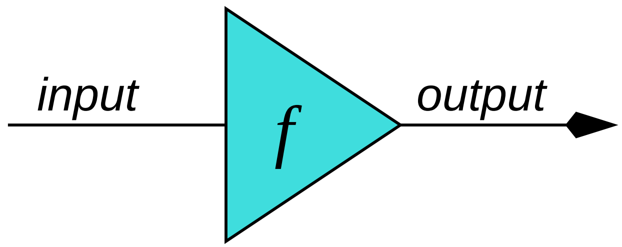
$\vdots \quad \mapsto$ combinations thereof.

Circuit behavior corresponds to conjunction of all its gate formulae.

Generalizing the concept: Simulink+Stateflow



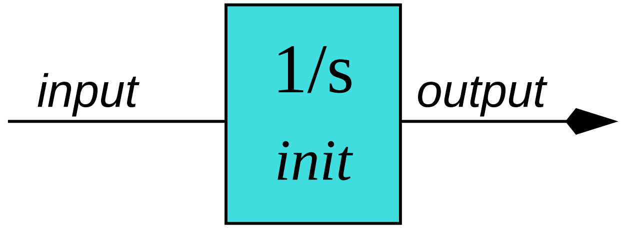
'Algebraic' blocks



- time-invariant transfer function $\mathit{output}(t) = f(\mathit{input}(t))$
- made 1st-order by making time implicit: $\mathrm{Flow} \equiv \mathit{output} = f(\mathit{input})$
- no constraints on initial value: $\mathrm{Init} \equiv \mathrm{true}$,
- discontinuous jumps always admissible $\mathrm{Jump} \equiv \mathrm{true}$,

> All the formulae are elements of a suitably rich 1st-order logics over $\mathbb{R}$.

Integrators



- integrates its input over time: $\mathit{output}(t) = \mathit{init} + \int_0^t \mathit{input}(u)\,du$.
- made semi-1st-order by using derivatives: $\mathrm{Flow} \equiv \frac{d\mathit{output}}{dt} = \mathit{input}$
- initial value is rest value: $\mathrm{Init} \equiv \mathit{output} = \mathit{init}$,
- discontinuous jumps don't affect output $\mathrm{Jump} \equiv \mathit{output} = \overleftarrow{\mathit{output}}$,

Use in Model Exploration

Given: Transition pred. $\mathrm{trans}(x, x')$, initial state pred. $\mathrm{init}(x)$, conj. invar. $\phi(x)$.

E.g., **Bounded Model Checking (BMC) algorithm:**

1. For given $i \in \mathbb{N}$ check for satisfi ability of

   $$\neg \left(\begin{array}{ll} & \mathrm{init}(x_0) \wedge \mathrm{trans}(x_0, x_1) \wedge \ldots \wedge \mathrm{trans}(x_{i-1}, x_i) \\ \Rightarrow & \phi(x_0) \wedge \ldots \wedge \phi(x_i) \end{array} \right).$$

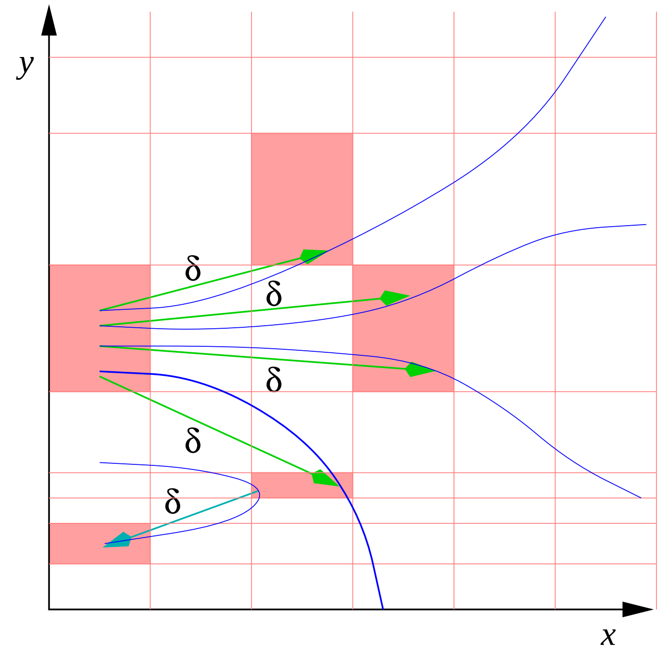
   If test succeeds then report violation of goal.
2. Otherwise repeat with larger i.

Can we use the predicates off-the-shelf?

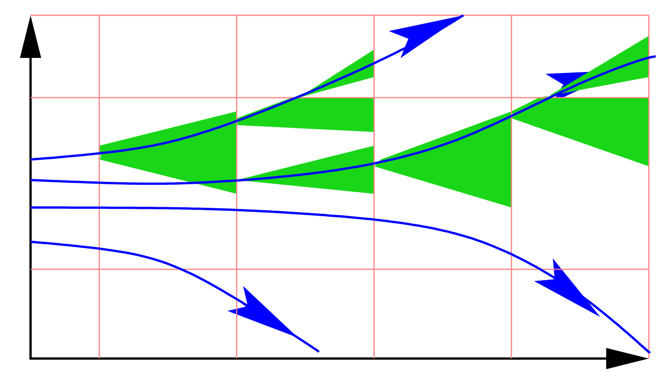
No, as dynamics is not in terms of pure pre-/post-relations.

Images of ODEs: Approaches

1. *Safe finite-state abstraction:*
   - E.g., discretization through quantization (and overapproximation); yields finite-state system.
   - exponential in dimension of system
   - coarse abstractions give many false negatives ⇝ CEGAR



2. *Hybridization:* chop the phase space; do piece-wise safe approximation by tractable dynamics (e.g., maps definable in decidable logics over $\mathbb{R}$)
   - potentially more concise,
   - yet still exponential in dimension of system



3. (Safely approximate) *on-the-fly computation of ODE images.*

Hybridization

Will not elaborate on into this issue here: approaches range from

- approximation by piecewise (i.e., in a grid element) constant differential inclusions obtained via interval-based safe approx. of upper and lower bounds on individual derivatives:

$$\frac{dx}{dt} = x^2 + 2y \wedge x \in [1,2] \wedge y \in [5,7] \quad \rightsquigarrow \quad \frac{dx}{dt} \in [11,18]$$

  a.o. [Henzinger, Kopke, Puri, Varaiya 1998] [Stursberg, Kowalewski 1999]

- to approximation by piecew. affi ne / multi-affi ne vector fi elds [Asarin, Dang, Girard 06]
- and to Taylor approximations [Piazza et al. 05, Lanotte, Tini 05]

For Lipschitz-continuous ODEs, imprecision generally is

- linear in grid width (though with different constants),
- exponential in length of time frame.

e.g., [Girard 2002; Asarin, Dang, Girard 2006]

Impact on decidability

Due to the (worst-case) exponential deviation over time, such hybridizations are not suffi cient for approximate (up to some ε) computation of the reachable state space over unbounded time frames.

Hence, questions like

- "If the distance of the reachable state space from a set of bad states larger than ε then provide a proof of this fact."

for fbws lacking a closed-form solution are i.g. not "decidable" by hybridization and related approximation schemes.

[Platzer, Clarke 2006]

...unless the fbw is attracting such that it cancels the accumulating error.

[Asarin, Dang, Girard 2006]

Principles of hybrid state-space exploration:

Iterating a 1st-order definable map

Checking safety

...in a finite Kripke structure:

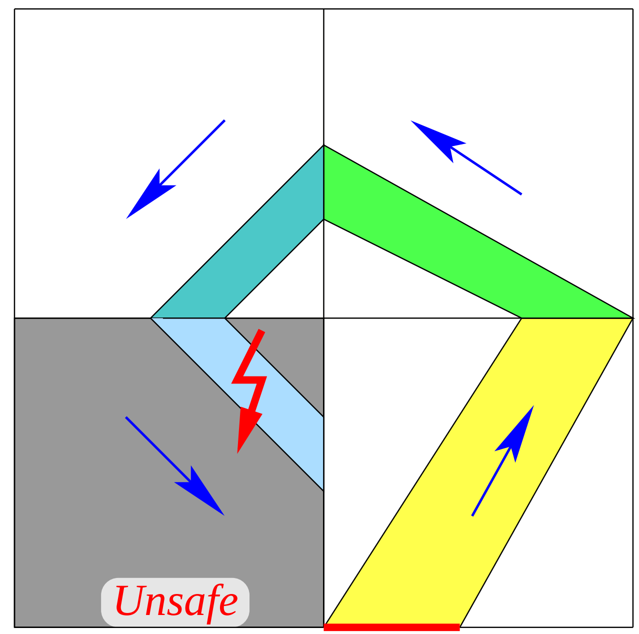
1. For increasing n, calculate the set $\mathit{Reach}^{\leq n}$ of states reachable in at most n steps.
2. Chain $\mathit{Reach}^{\leq 1} \subseteq \mathit{Reach}^{\leq 2} \subseteq \dots$ has only a finite ascending subchain due to finiteness of state-space.

$\Rightarrow$ 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:

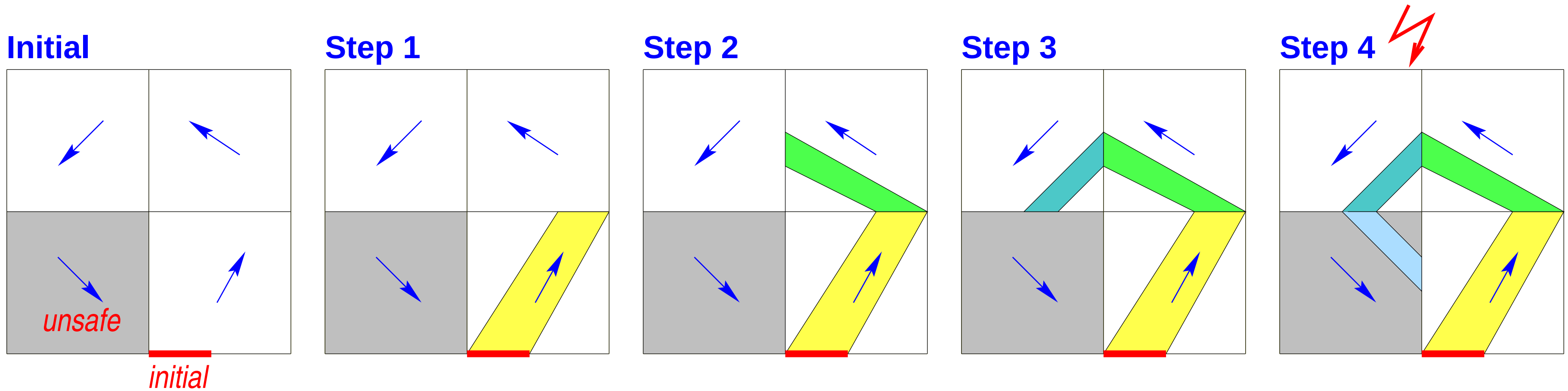
Similar fixpoint construction



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.

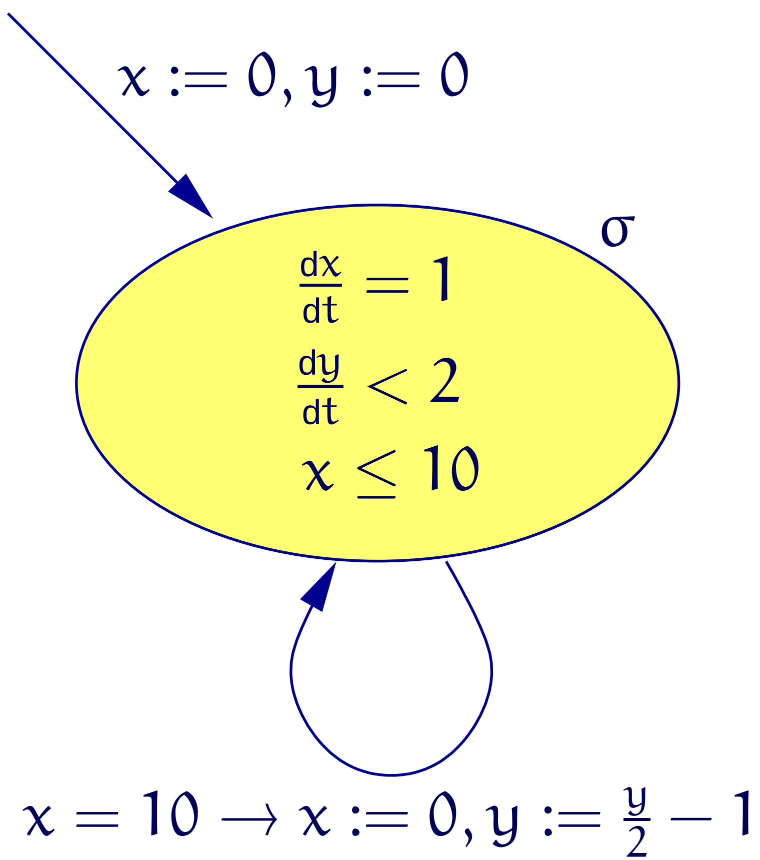
Requires: Effective representations of transition relation, continuous dynamics, and initial, intermediate, and unsafe state sets s.t.

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.

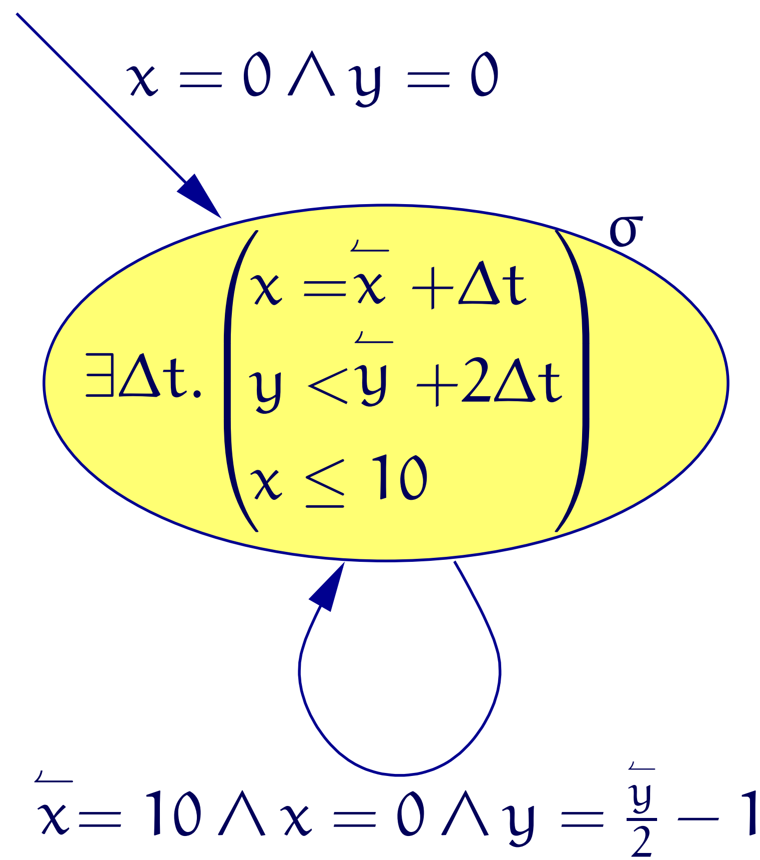
(implemented in, e.g., HyTech [Henzinger, Ho, Wong-Toi, 1995–])

From hybrid automata to logic

A:



A:



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

ALBERT-LUDWIGS UNIVERSITÄT FREIBURG
UNIVERSITÄT DES SAARLANDES

Essentials of polynomial HA

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

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

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

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

Safety of hybrid automata

⇒ An unsafe state is reachable within n steps iff

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

is satisfiable, where

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

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

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

ALBERT-LUDWIGS UNIVERSITÄT FREIBURG
CARL VON OSSIETZKY universität OLDENBURG
UNIVERSITÄT DES SAARLANDES

The semi-decision procedure

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

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

3. By successively testing increasing n, this yields a *semi-decision procedure for reachability of unsafe states*:
   (a) Select some $n \in \mathbb{N}$,
   (b) check *Unsafe*$_n$.
   (c) If this yields `true` then an unsafe state is reachable. Report this and terminate.
   (d) Otherwise select strictly larger $n \in \mathbb{N}$ 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 $\mathit{Reach}_{\sigma'}^{\leq n+1} \Rightarrow \mathit{Reach}_{\sigma'}^{\leq n}$ holds for some $n \in \mathbb{N}$ s.t. the reachable state set can be calculated in a finite number of steps.

But the reachability problem is undecidable in general!

Decidability

The problem is undecidable already for very restricted subclasses of hybrid automata:

- Stopwatch automata [Čerāns 1992; Wilke 1994; Henzinger, Kopke, Puri, Varaiya 1995]
- 3-dimensional piecewise constant derivative systems [Asarin, Maler, Pnueli 1995]
- ...

Decidable subclasses tend to abandon interplay between changes in continuous dynamics and transition selection/effect, or the dimensionality is extremely low:

- Timed automata [Alur, Dill 1994] and initialized rectangular automata [Henzinger, Kopke, Puri, Varaiya 1995]
- multi-priced timed automata [Larsen, Rasmussen 2005], priced timed automata with pos. and neg. rates [Boyer, Brihaye, Bruyère, Raskin 2007]
- 2-dimensional piecewise constant derivative systems [Maler, Pnueli 1994], also non-deterministic [Asarin, Schneider, Yovine 2001]
- ...

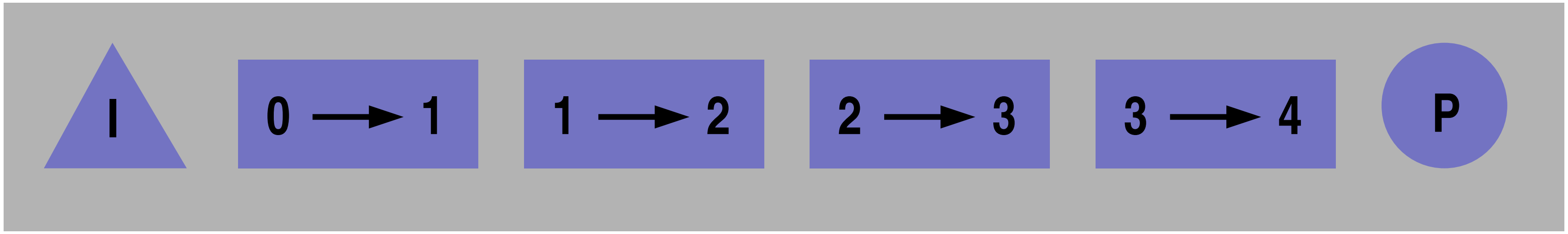
Iterating over the state-space

...how do we do this in practice

- on very large state spaces, both continuous and discrete?
- for non-polynomial assignments / pre-post-relations?
- for non-linear differential equations?

SAT modulo theory
as an engine for
bounded model checking of
linear hybrid automata

Bounded Model Checking (BMC)



Method:

- construct formula that is satisfi able iff error trace of length k exists
- formula is a k–fold unrolling of the systems transition relation, concatenated with a characterization of the initial state(s) and the (unsafe) state to be reached
- use appropriate decision procedure to decide satisfi ability of the formula
- usually BMC is carried out incrementally for $k = 0, 1, 2, \ldots$ until an error trace is found or tired

Bounded Model Checking (BMC) algorithm

1. For given $i \in \mathbb{N}$ check for satisfi ability of

   $$\neg \left(\begin{array}{ll} & init(x_0) \wedge trans(x_0, x_1) \wedge \ldots \wedge trans(x_{i-1}, x_i) \\ \Rightarrow & \phi(x_0) \wedge \ldots \wedge \phi(x_i) \end{array} \right).$$

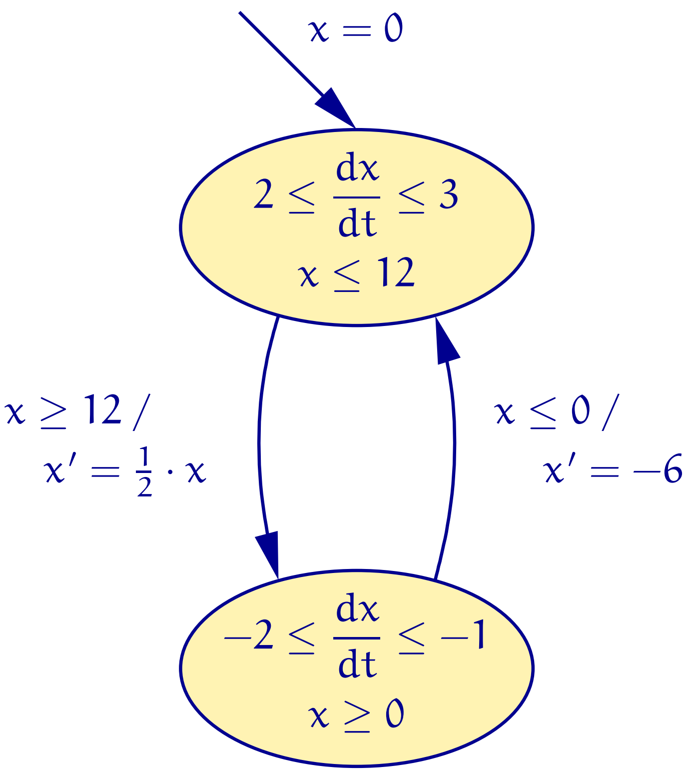
   If test succeeds then report violation of goal.
2. Otherwise repeat with larger i.

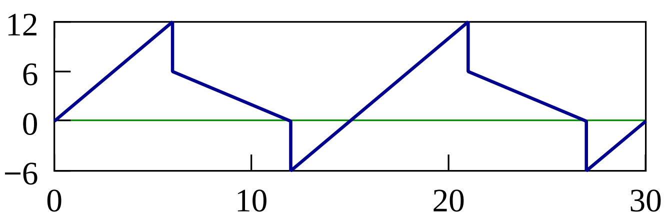
Linear hybrid automata

- In this part, we will concentrate on hybrid automata where the initiation and transition predicates are linear and the activities give rise to polyhedral pre-post-relations:
  - $\mathit{initial}_\sigma \in \mathrm{FOL}(\mathbb{R}, +, \leq)$ with $\mathrm{free}(\mathit{initial}_\sigma) \subseteq \{x_1, \ldots, x_d\}$ for each σ,
  - $\mathit{act}_\sigma = \mathit{diff}_\sigma \wedge \mathit{inv}_\sigma \in \mathrm{FOL}(\mathbb{R}, +, \leq)$ for each σ, where
    - diff_σ is purely conjunctive and $\mathrm{free}(\mathrm{diff}_\sigma) \subseteq \{\frac{dx_1}{dt}, \ldots, \frac{dx_d}{dt}\}$,
    - inv_σ is conjunctive and $\mathrm{free}(\mathrm{inv}_\sigma) \subseteq \{x_1, \ldots, x_d\} \cup \{\overleftarrow{x_1}, \ldots, \overleftarrow{x_d}\}$,
  - $\mathit{trans}_{\sigma \to \sigma'} \in \mathrm{FOL}(\mathbb{R}, +, \leq)$ with $\mathrm{free}(\mathit{trans}_{\sigma \to \sigma'}) \subseteq \{x_1, \ldots, x_d\} \cup \{\overleftarrow{x_1}, \ldots, \overleftarrow{x_d}\}$ for each σ, σ'.

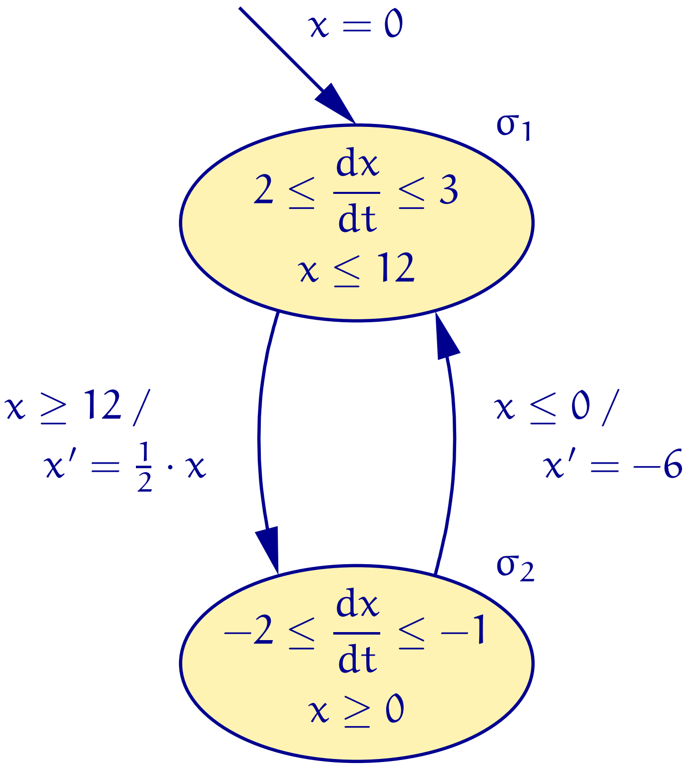
- **N.B.:** Such continuous activities give rise to linear pre-/post-relations.

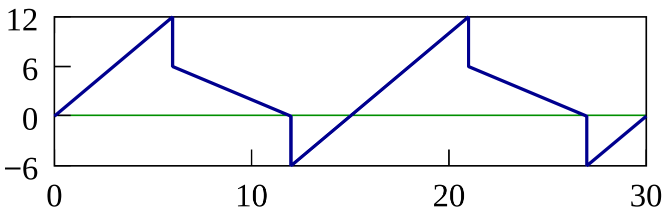
Linear Hybrid Automata (LHA)





BMC of Linear Hybrid Automata





Initial state:

$$\sigma_1^0 \ \wedge\ \neg\sigma_2^0 \ \wedge\ x^0 = 0.0$$

Jumps:

$$\sigma_1^i \wedge \sigma_2^{i+1} \ \rightarrow (x^i \geq 12) \ \wedge\ (x^{i+1} = 0.5 \cdot x^i) \ \wedge\ t^i = 0$$

Flows:

$$\sigma_1^i \wedge \sigma_1^{i+1} \ \rightarrow \begin{cases} & (x^i + 2\,t^i) \ \leq\ x^{i+1} \ \leq\ (x^i + 3\,t^i) \\ \wedge & (x^{i+1} \leq 12) \\ \wedge & (t^i > 0) \end{cases}$$

Quantifier−free Boolean combinations of linear arithmetic constraints over the reals

Parallel composition corresponds to conjunction of formulae

$\longrightarrow$ No need to build product automaton

Ingredients of a Solver for BMC of LHA

BMC of LHA yields very large boolean combination of linear arithmetic facts.

Davis Putnam based SAT-Solver:

- ☺ tackle instances with $\gg 10.000$ variables
- ☺ effi cient handling of disjunctions
- ☹ Boolean variables only

Linear Programming Solver:

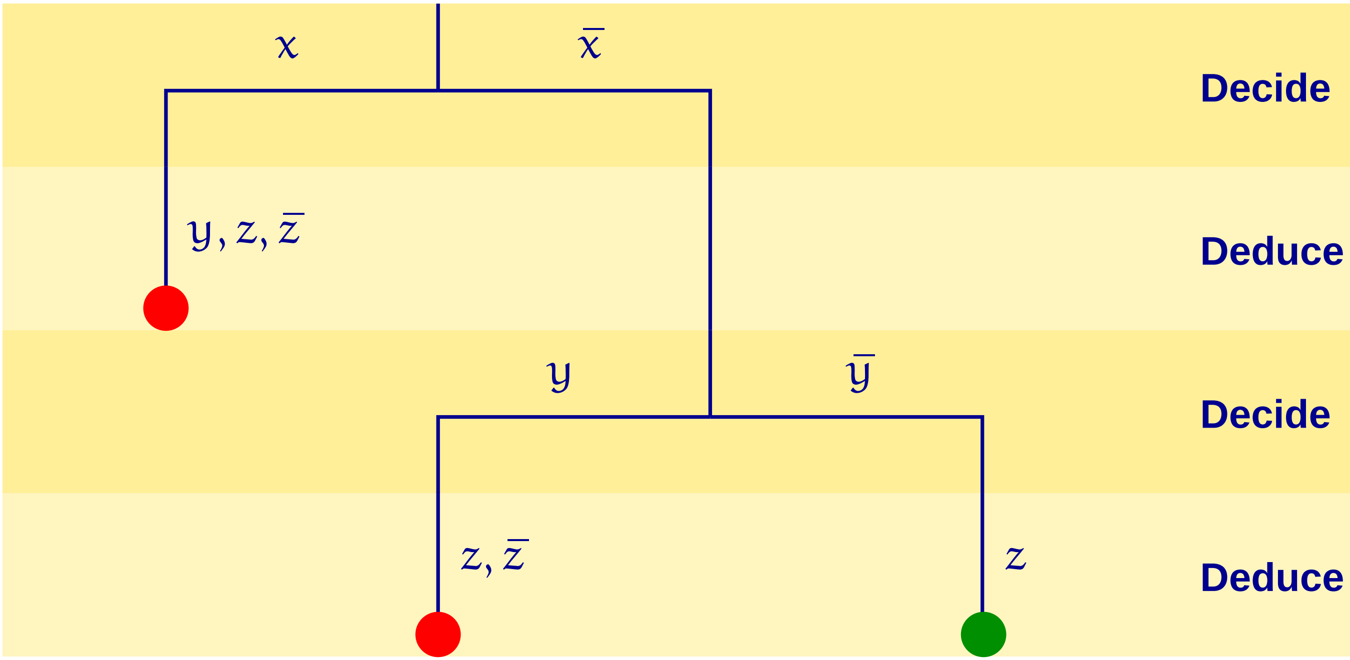
- ☺ solves large conjunctions of linear arithmetic inequations
- ☺ effi cient handling of continuous variables ($> 10^6$)
- ☹ no disjunctions

Idea: Combine both methods to overcome shortcomings.

$\rightsquigarrow$ **SAT modulo theory**

Davis–Putnam Procedure

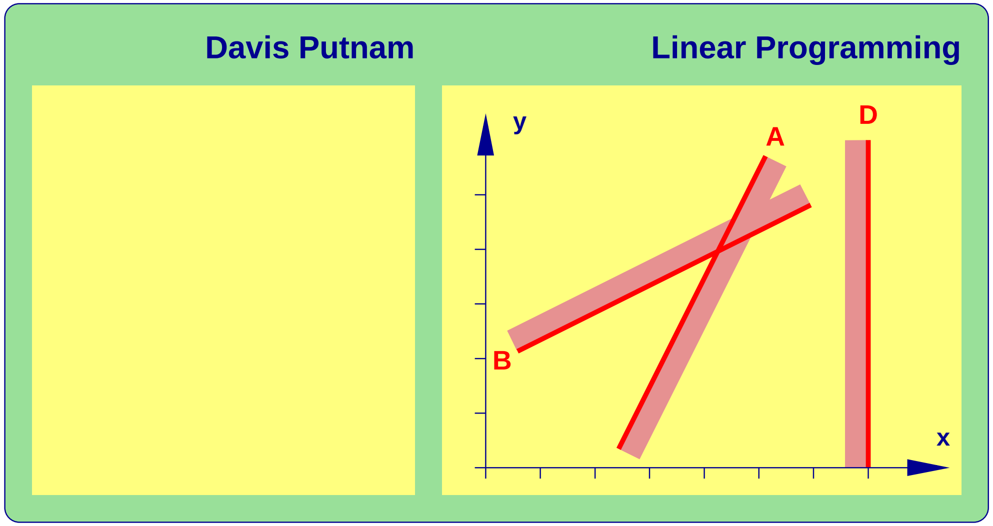
$$(x \vee y \vee z)$$
$$\wedge\ (\bar{x} \vee y)$$
$$\wedge\ (\bar{y} \vee z)$$
$$\wedge\ (\bar{x} \vee \bar{y} \vee \bar{z})$$
$$\wedge\ (x \vee \bar{y} \vee \bar{z})$$



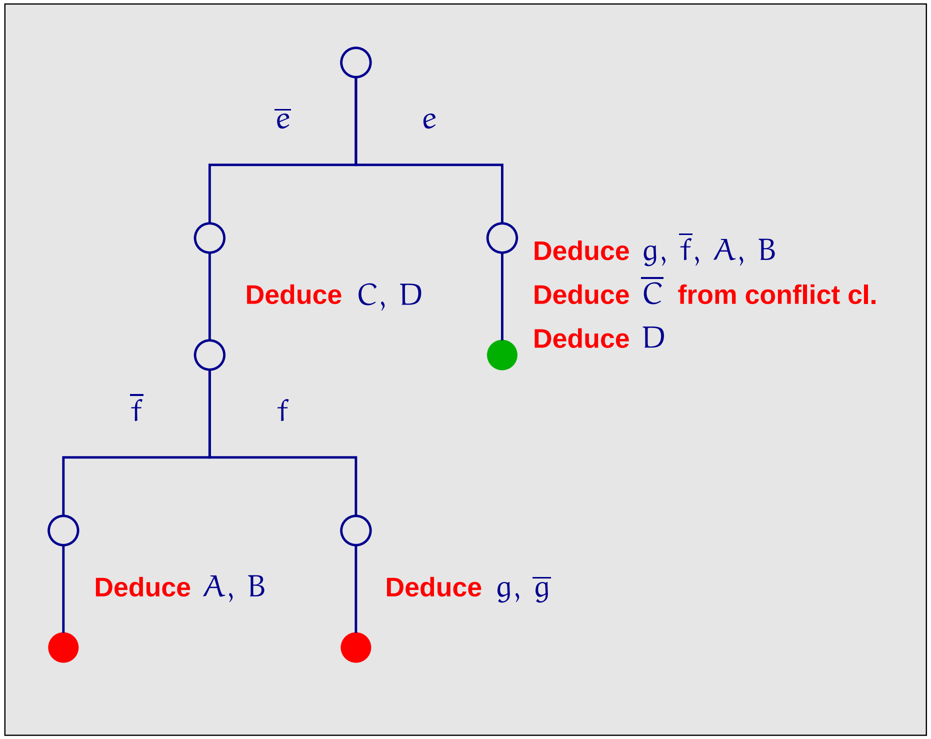
Satisfiability solving for decidable theories:

Lazy theorem proving & DPLL(T)

The Lazy TP Scheme: LinSAT



Learned conflict clause: $\overline{A} + \overline{B} + \overline{C} \geq 1$



DPLL search

1. traversing possible truth-value assignments of Boolean part
2. incrementally (de-)constructing a *conjunctive* arithmetic constraint system
3. querying external solver to determine consistency of arithm. constr. syst.

Deciding the conjunctive $\mathcal{T}$-problems

For $\mathcal{T}$ being linear arithmetic over $\mathbb{R}$, this can be done by linear programming:

$$\bigwedge_{i=1}^{n} \sum_{j=1}^{m} A_{i,j} x_j \leq b_j \quad \text{iff} \quad A\mathbf{x} \leq \mathbf{b}$$

$\rightsquigarrow$ Solving LP

$$\begin{aligned} \text{maximize} \quad & \mathbf{c}^T \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \end{aligned}$$

with arbitrary $\mathbf{c}$ provides consistency information.

Deciding the conjunctive T-problems (cntd.)

To cope with systems C containing *strict* inequations $\sum_{j=1}^{m} A_{i,j}x_j < b_j$, one

classically: introduces a slack variable ε,

- then replaces $\sum_{j=1}^{m} A_{i,j}x_j < b_j$ by $\sum_{j=1}^{m} A_{i,j}x_j + \varepsilon \leq b_j$,
- solves the resultant LP L, maximizing the objective function ε

$\leadsto$ C is satisfi able iff L is satisfi able with optimum solution > 0.

more elegantly: treat ε symbolically:

- use 1 and ε as fundamental units of the number system,
- represent all numbers and coeffi cients in inequations as linear combinations of 1 and ε

[Dutertre, de Moura 2006: Yices]

Extracting reasons for $\mathcal{T}$-conflicts

Goal: In case that the original constraint system

$$C = \left(\begin{array}{ccc} & \bigwedge_{i=1}^{k} & \sum_{j=1}^{n} \mathbf{A}_{i,j}\mathbf{x}_j \leq \mathbf{b}_i \\ \wedge & \bigwedge_{i=k+1}^{n} & \sum_{j=1}^{n} \mathbf{A}_{i,j}\mathbf{x}_j < \mathbf{b}_i \end{array} \right)$$

is infeasible, we want a subset $I \subseteq \{1, \ldots, n\}$ such that

- the subsystem $C|_I$ of the constraint system containing only the conjuncts from I also is infeasible,
- yet the subsystem is *irreducible* in the sense that any proper subset J of I designates a feasible system $C|_J$.

Such an irreducible infeasible subsystem (IIS) is a prime implicant of all the possible reasons for failure of the constraint system C.

Extracting IIS

Provided constraint system C contains only non-strict inequations,

- extraction of IIS can be reduced to finding extremal solutions of a dual system of linear inequations, similar to Farkas' Lemma (Gleeson & Ryan 1990; Pfetsch, 2002)
- to keep the objective function bounded, one can use dual LP

$$
\begin{array}{lrcl}
\text{maximize} & \mathbf{w}^T\mathbf{y} & & \\
\text{subject to} & \mathbf{A}^T\mathbf{y} & = & 0 \\
 & \mathbf{b}^T\mathbf{y} & = & 1 \\
 & \mathbf{y} & \geq & 0 \\
\text{where} & \mathbf{w}_i & = & \begin{cases} -1 & \text{if } b_i \leq 0, \\ 0 & \text{if } b_i > 0 \end{cases}
\end{array}
$$

- choice of $\mathbf{w}$ guarantees boundedness of objective function

$\Longrightarrow$ optimal solution exists whenever the LP is feasible.

! For such a solution, $I = \{i \mid \mathbf{y}_i \neq 0\}$ is an IIS.

Extensions & Optimizations

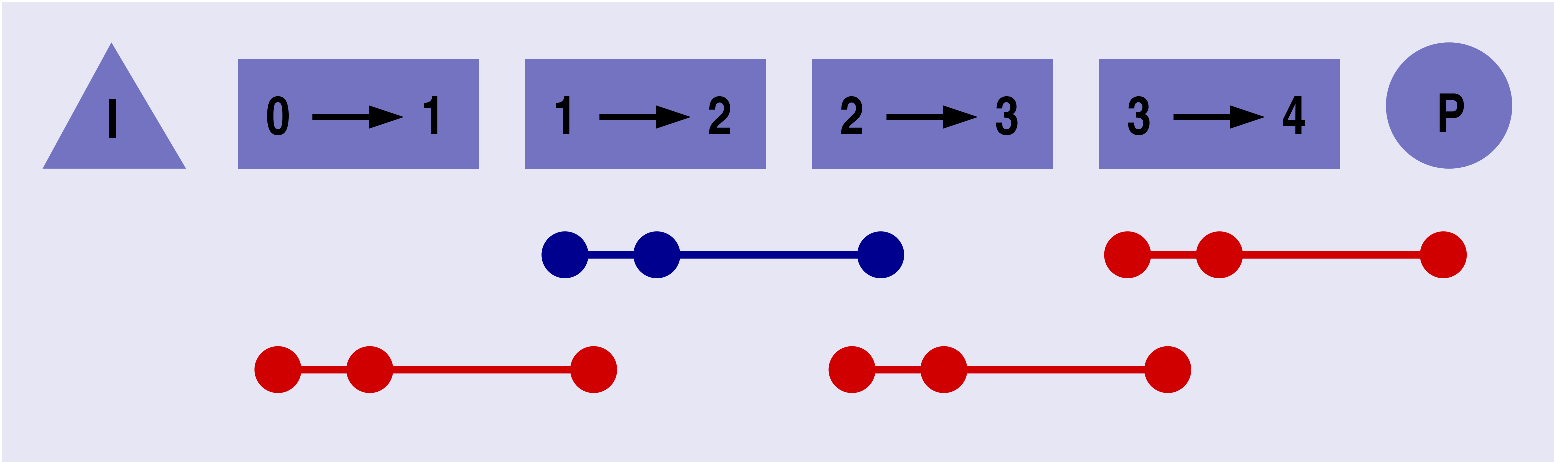
DPLL(T): If the T solver can itself do fwd. inference, it cannot only prune the search tree through conflict detection, but also through constraint propagation:

1. SAT solver assigns truth values to subset $\mathrm{C} \subset A$ of the set A of constraints occurring in the input formula,
2. T solver finds them to be consistent *and* to imply a truth value assignment to further T constraints $\mathrm{D} \subseteq A \setminus \mathrm{C}$,
3. these truth-value assignments are performed in the SAT solver store before resuming SAT solving.

SAT modulo theory for LinSAT

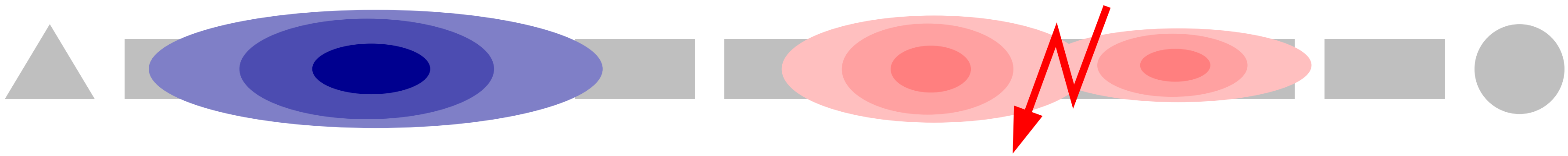
- SAT modulo theory solvers reasoning over linear arithmetic as a theory are readily available: E.g.,
  - LPSAT [Wolfman & Weld, 1999]
  - ICS [Filliatre, Owre, Rueß, Shankar 2001], Simplics [de Moura, Dutertre 2005], Yices [Dutertre, de Moura 2006]
  - MathSAT [Audemard, Bertoli, Cimatti, Kornilowicz, Sebastiani, Bozzano, Juntilla, van Rossum, Schulz 2002–]
  - SVC [Barrett, Dill, Levitt 1996], CVC [Stump, Barrett, Dill 2002], CVC Lite [Barrett, Berezin 2004], CVC3 [Barrett, Fuchs, Ge, Hagen, Jovanovic 2006]
  - HySAT [Herde & Fränzle, 2004]
  - ...
- Their use for analyzing linear hybrid automata has been advocated a number of times (e.g. in [Audemard, Bozzano, Cimatti, Sebastiani 2004]).
- They combine symbolic handling of discrete state components (via SAT solving) with symbolic handling of continuous state components.
- Formulae arising in BMC have a specifi c structure, which can be exploited for accelerating SAT search [Strichman 2004]

Pimp my SMT Solver: Isomorphy Inference



- learning schemes employed in SAT solvers account for a major fraction of the running time
- creation of a conflict clause is even more expensive in a combined solver as it entails the extraction of an IIS
- idea: exploit symmetric structure to add isomorphic copies of a conflict clause to the problem
- thus multiplying the benefi ttaken from the time–consuming reasoning process

Pimp my SMT Solver: Decision Strategies

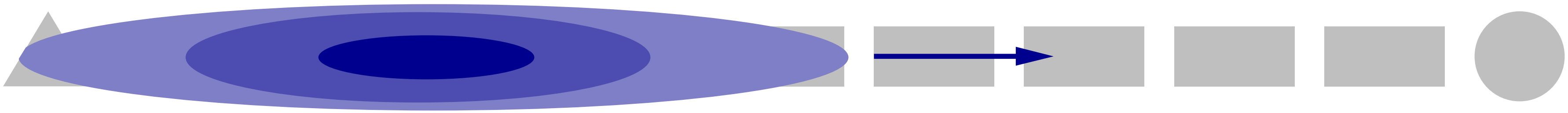


General–Purpose Decision Heuristics:

- distant cycles of the transition relation are being satisfied independently
- until they finally turn out to be incompatible, often entailing the need to backtrack over long distances

For BMC we can use smarter decision strategies !

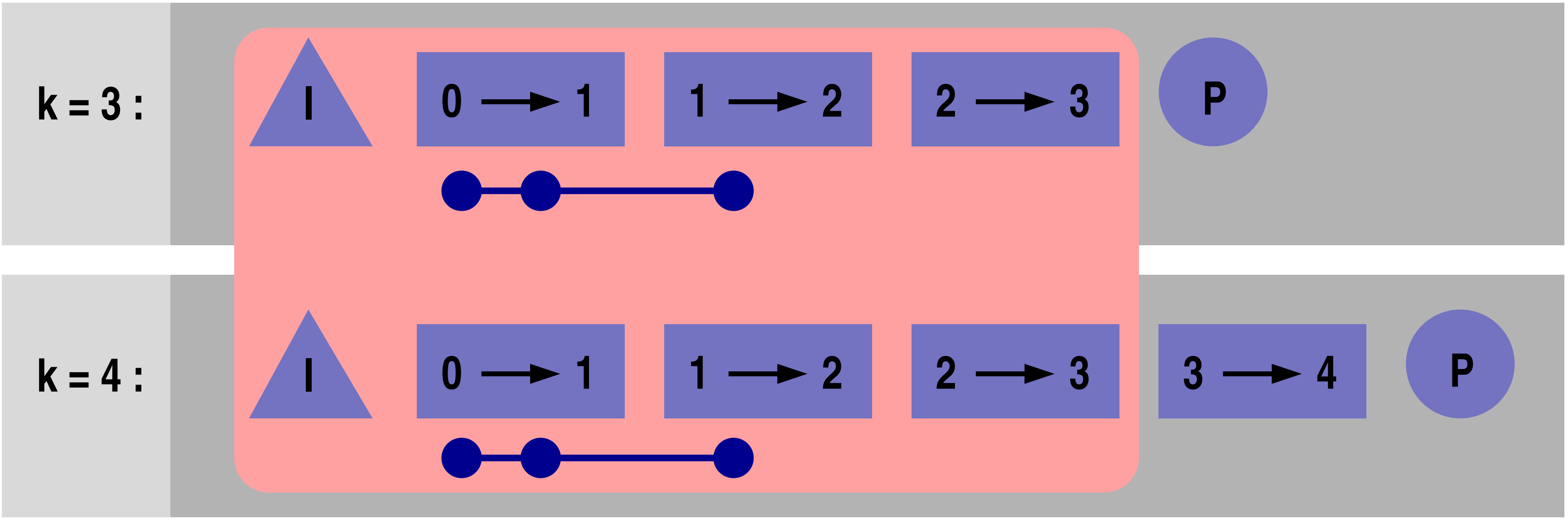
Pimp my SMT Solver: Decision Strategies



Forward–Heuristics:

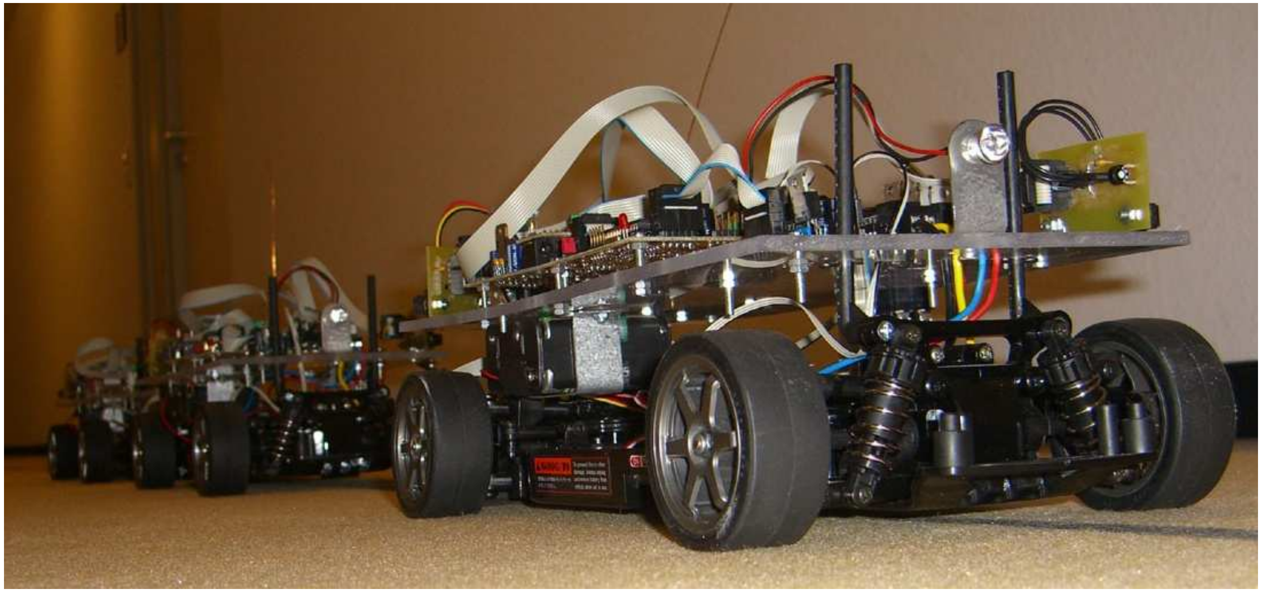
- select decision variables in the natural order induced by the linear structure of the BMC formula
- e.g. starting with variables from cycle 0, then from cycle 1, 2, etc.
- thereby extending prefi xes of legal runs of the system
- allows conficts to be detected and resolved more locally

Pimp my SMT Solver: Knowledge Reuse



- when carrying out BMC incrementally the consecutive formulas share a large number of clauses
- thus, when moving from instance k to $k+1$ (or doing them in parallel), we can conjoin the conflict clauses derived when solving the k–instance to the $k+1$–instance (and vice versa)
- only sound for conflict clauses inferred from clauses which are common to both instances

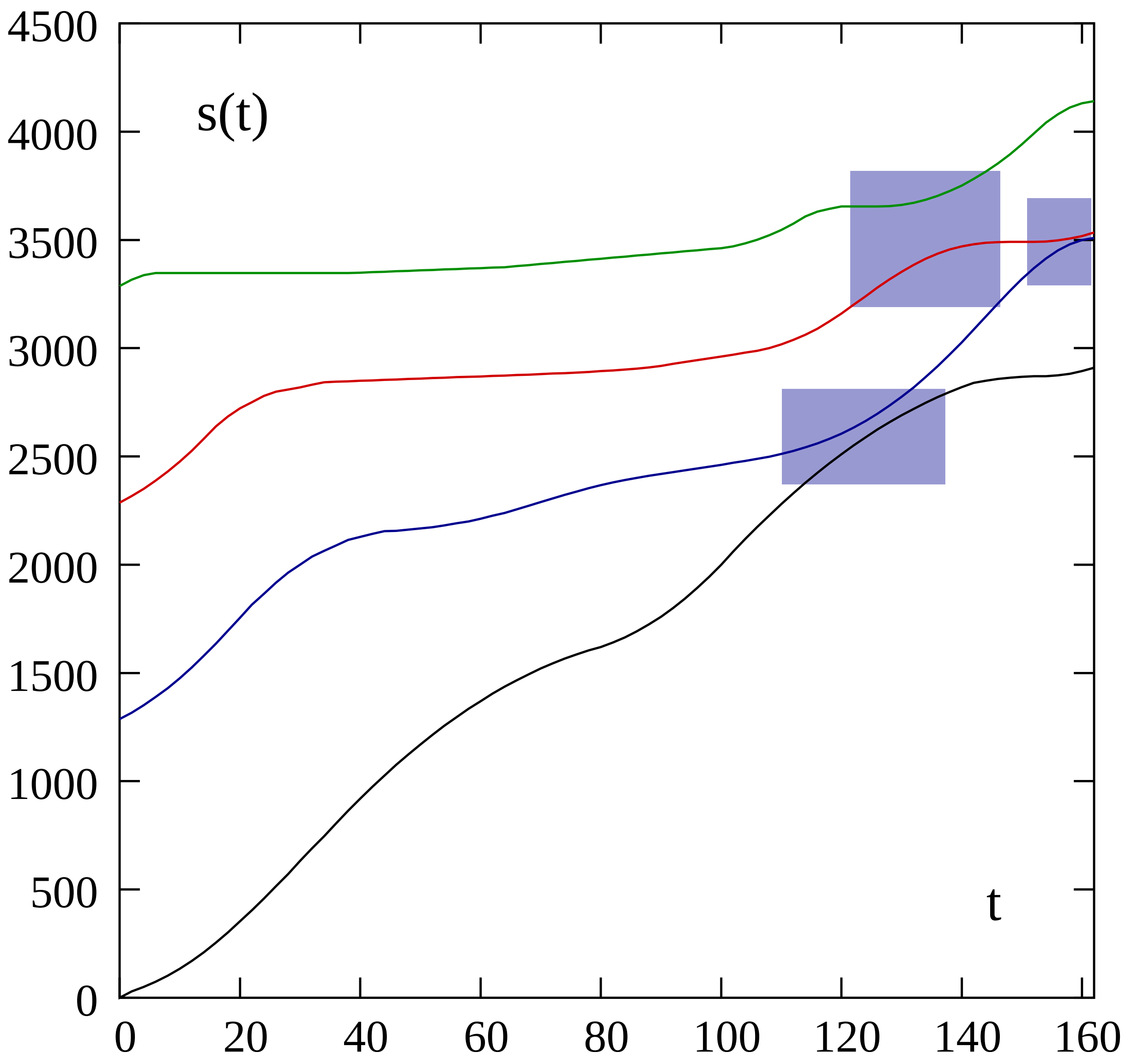
Case Study: Elastic Distance Control



System Overview:

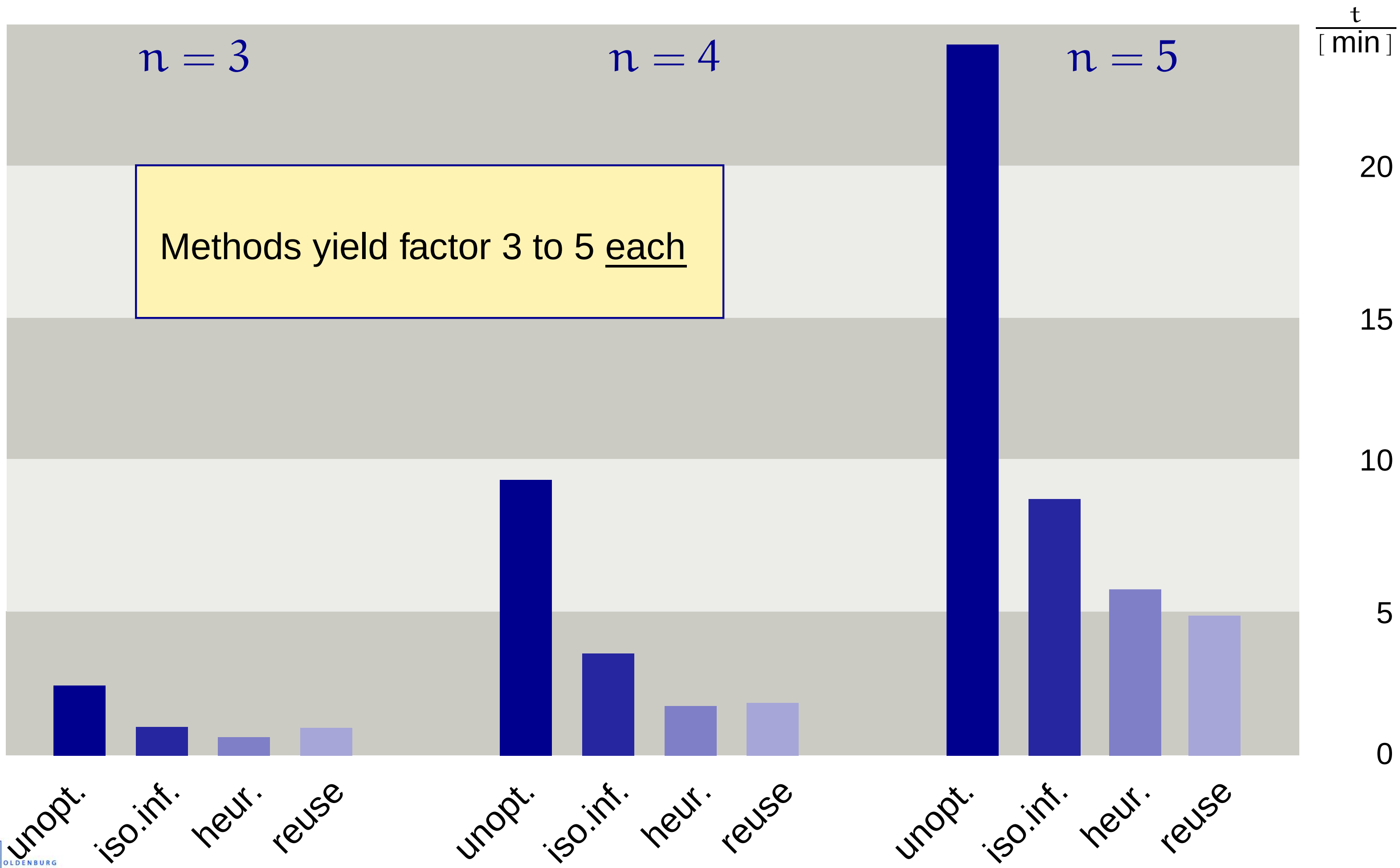
- n cars running on the same lane
- each car has a collision avoidance controller
- controller has four control modes:
  - free running $\leftrightarrow$ front or/and back intrusion into safety envelope
  - elastic coupling in case of intrusion

Sample Trace



Case Study: Elastic Distance Control

Results: (total time needed to solve all 22+1 instances until error trace is found)



but...

- what to do if assignments are non-linear?

$$x := \sin y + e^x$$

- what to do if continuous behavior is more general:
  - linear differential equations?

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x} + \mathbf{b}$$

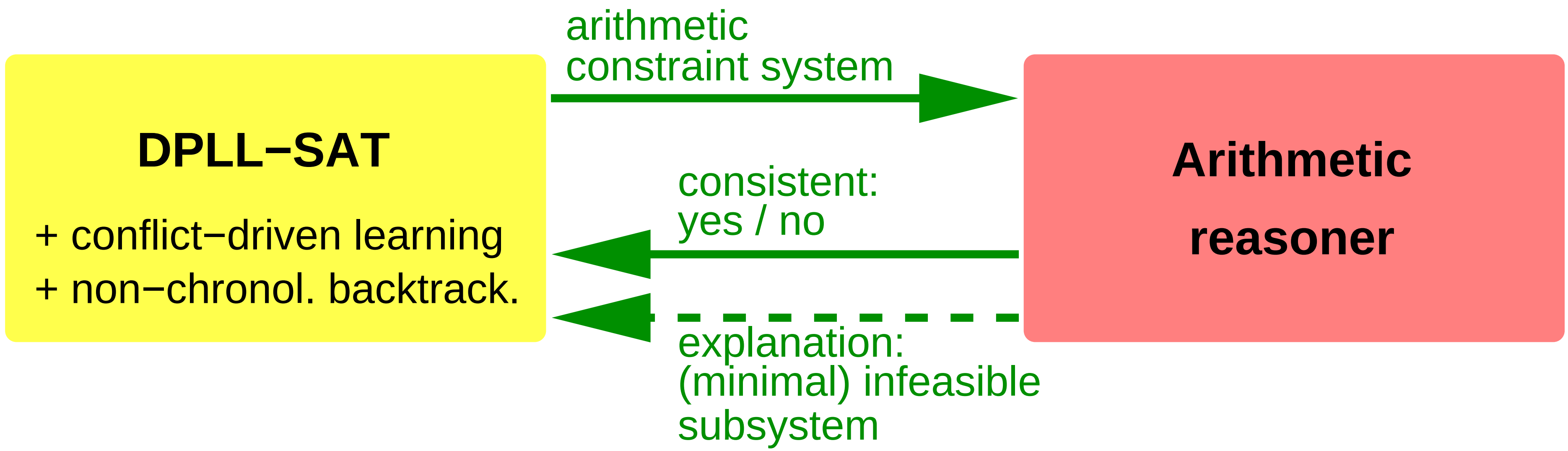
  - non-linear differential equations?

$$\frac{dx}{dt} = \sin y$$

Satisfiability solving in undecidable arithmetic domains

iSAT algorithm

Classical Lazy TP Layout



Problems with extending it to richer arithmetic domains:

- undecidability: answer of arithmetic reasoner no longer two-valued; don't know cases arise
- explanations: how to generate (nearly) minimal infeasible subsystems of undecidable constraint systems?

The Task

Find satisfying assignments (or prove absence thereof) for large (thousands of Boolean connectives) formulae of shape

$$
\begin{array}{ll}
 & (b_1 \implies x_1^2 - \cos y_1 < 2y_1 + \sin z_1 + e^{u_1}) \\
\wedge & (x_5 = \tan y_4 \vee \tan y_4 > z_4 \vee \ldots) \\
\wedge & \ldots \\
\wedge & (\frac{dx}{dt} = -\sin x \wedge x_3 > 5 \wedge x_3 < 7 \wedge x_4 > 12 \wedge \ldots) \\
\wedge & \ldots
\end{array}
$$

Conventional solvers

- do either address much smaller fragments of arithmetic
  - decidable theories: no transcendental fct.s, no ODEs
- or tackle only small formulae
  - some dozens of Boolean connectives.

Algorithmic basis:

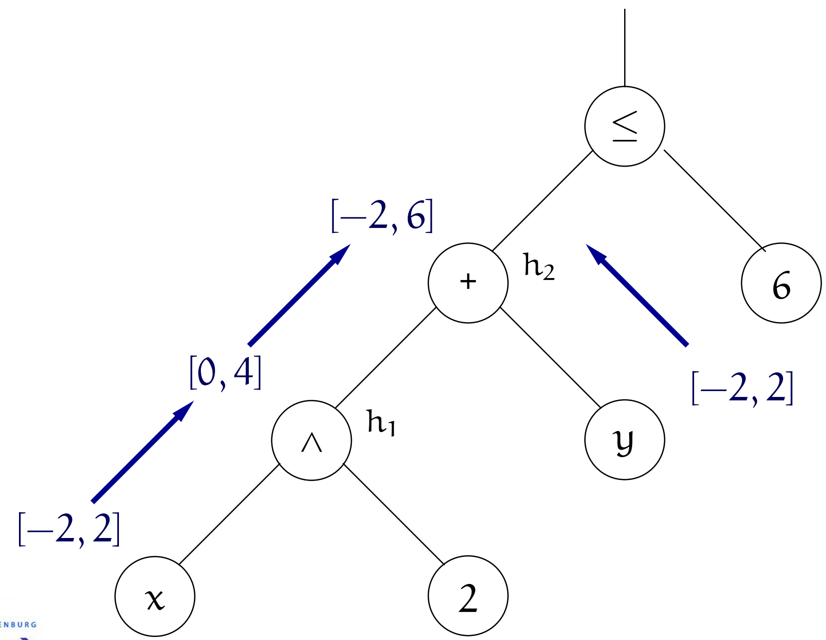
Interval constraint propagation (Hull consistency version)

Interval Constraint Solving (1)

- Complex constraints are rewritten to “triplets” (primitive constraints):

$$x^2 + y \leq 6 \;\rightsquigarrow\; \begin{array}{rll} c_1: & & h_1 \triangleq x \,\hat{}\, 2 \\ c_2: & \wedge & h_2 \triangleq h_1 + y \\ & \wedge & h_2 \leq 6 \end{array}$$

- “Forward” interval propagation yields justification for constraint satisfaction:



$$\begin{array}{rl} & x \in [-2, 2] \\ \wedge & y \in [-2, 2] \end{array}$$

$$\Downarrow$$

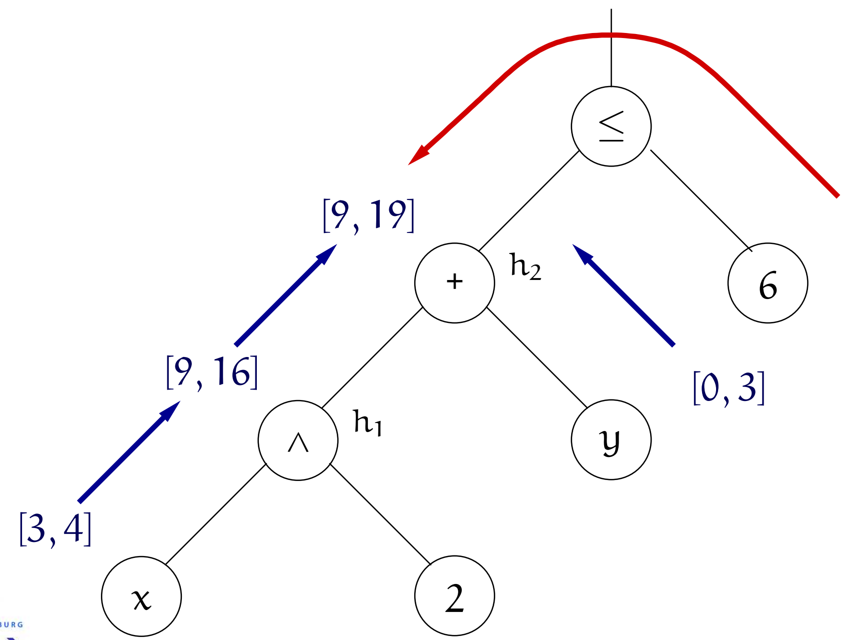
$h_2 \leq 6$ is satisfied in box

Interval Constraint Solving (1)

- Complex constraints are rewritten to “triplets” (primitive constraints):

$$x^2 + y \leq 6 \quad \rightsquigarrow \quad \begin{array}{rcl} c_1: & & h_1 \hat{=} x \,\hat{}\, 2 \\ c_2: & \wedge & h_2 \hat{=} h_1 + y \\ & \wedge & h_2 \leq 6 \end{array}$$

- Interval propagation (fwd & bwd) yields witness for unsatisfi ability:



$$\begin{array}{rl} & x \in [3, 4] \\ \wedge & y \in [0, 3] \end{array}$$

$$\Downarrow$$

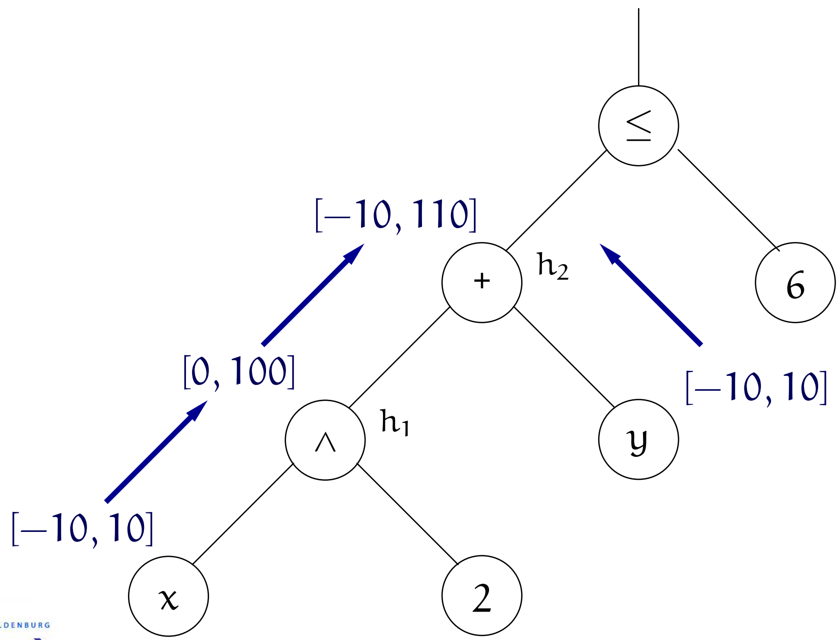
$h_2 \leq 6$ is unsat. in box

Interval Constraint Solving (1)

- Complex constraints are rewritten to "triplets" (primitive constraints):

$$x^2 + y \leq 6 \quad \rightsquigarrow \quad \begin{array}{rcl} c_1: & & h_1 \triangleq x \wedge 2 \\ c_2: & \wedge & h_2 \triangleq h_1 + y \\ & \wedge & h_2 \leq 6 \end{array}$$

- Interval prop. (fwd & bwd until fi xpoint is reached) yields contraction of box:



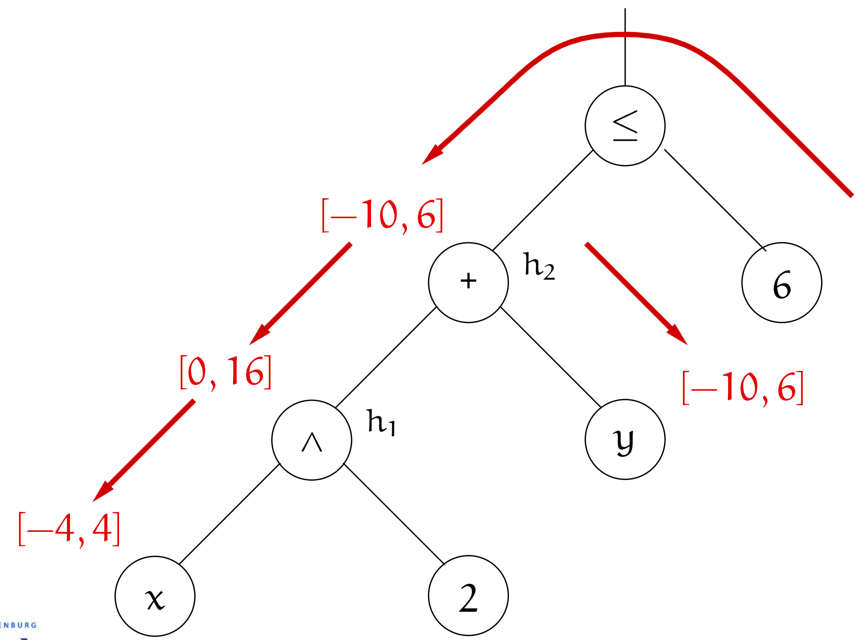
$$\begin{array}{r} x \in [-10, 10] \\ \wedge\ y \in [-10, 10] \end{array}$$

Interval Constraint Solving (1)

- Complex constraints are rewritten to “triplets” (primitive constraints):

$$x^2 + y \leq 6 \quad \rightsquigarrow \quad \begin{array}{rcl} c_1 : & & h_1 \triangleq x \wedge 2 \\ c_2 : & \wedge & h_2 \triangleq h_1 + y \\ & \wedge & h_2 \leq 6 \end{array}$$

- Interval prop. (fwd & bwd until fixpoint is reached) yields contraction of box:



$$\begin{array}{rl} & x \in [-10, 10] \\ \wedge & y \in [-10, 10] \end{array}$$

$$\Downarrow$$

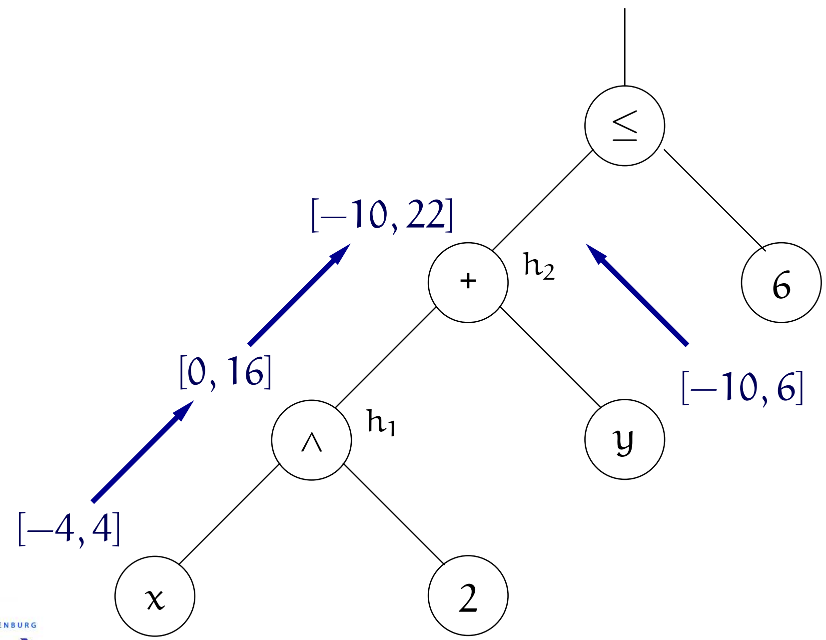
$$\begin{array}{rl} & x \in [-4, 4] \\ \wedge & y \in [-10, 6] \end{array}$$

Interval Constraint Solving (1)

- Complex constraints are rewritten to “triplets” (primitive constraints):

$$x^2 + y \leq 6 \quad \rightsquigarrow \quad \begin{array}{lrl} c_1: & & h_1 \triangleq x \wedge 2 \\ c_2: & \wedge & h_2 \triangleq h_1 + y \\ & \wedge & h_2 \leq 6 \end{array}$$

- Interval prop. (fwd & bwd until fi xpoint is reached) yields contraction of box:



Constraint is not satisfi ed

by the contracted box!

$$\begin{array}{rl} & x \in [-4, 4] \\ \wedge & y \in [-10, 6] \end{array}$$

Interval contraction

Backward propagation yields rectangular overapproximation of non-rectangular pre-images.

Thus, interval contraction provides a highly incomplete deduction system:

$$\begin{array}{rl} & x \in [0,\infty) \\ \wedge & h \hat{=} x \cdot y \\ \wedge & h > 5 \end{array} \implies \begin{array}{rl} & x \in (0,\infty) \\ \wedge & y \in (0,\infty) \end{array} \implies h \in (0,\infty) \not\Longrightarrow h > 5$$

$\leadsto$ enhance through branch-and-prune approach.

Schema of Interval-CP based CS Alg. / DPLL

Given: Constraint / clause set $C = \{c_1, \ldots, c_n\}$,
initial box (= cartesian product of intervals) B in $\mathbb{R}^{|\mathrm{free}(C)|}$ / $\mathbb{B}^{|\mathrm{free}(C)|}$

Goal: Find box $B' \subseteq B$ containing satisfying valuations throughout
or show non-existence of such B'.

Alg.:

1. $L := \{B\}$
2. If $L \neq \emptyset$ then take some box $b :\in L$, (LIFO)
   otherwise report "unsatisfiable" and stop.
3. Use contraction to determine a sub-box $b' \subseteq b$. (Unit Prop.)
4. If $b' = \emptyset$ then set $L := L \setminus \{b\}$, goto 2.
5. Use forward interval propagation to determine whether all constraints are satisfied throughout b'; if so then report b' as satisfying and stop.
6. If $b' \subset b$ then set $L := L \setminus \{b\} \cup \{b'\}$, goto 2.
7. Split b into subboxes b_1 and b_2, set $L := L \setminus \{b\} \cup \{b_1, b_2\}$, goto 2.

Observation

DPLL-SAT and interval-CP based CS are inherently similar:

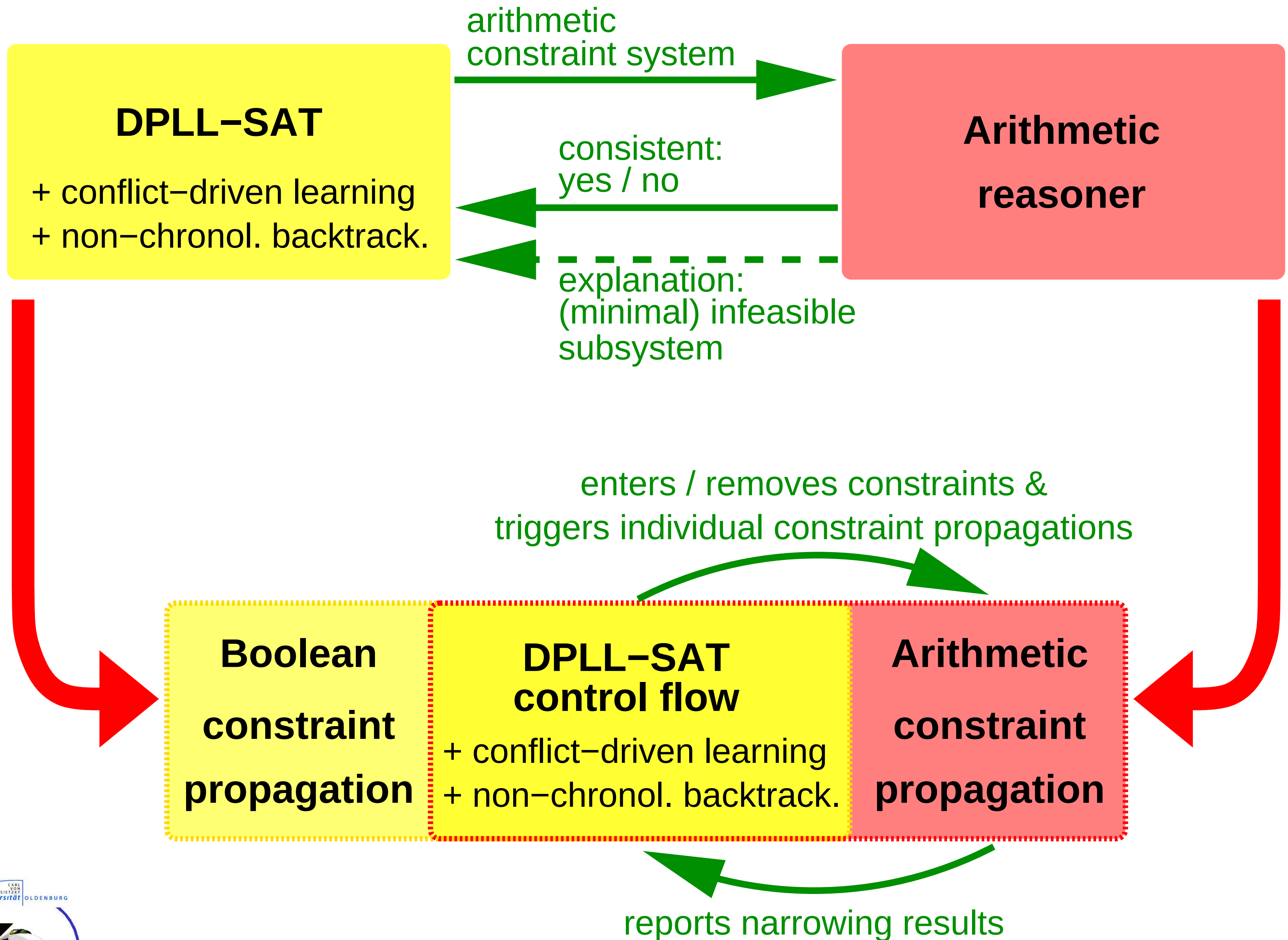
	DPLL-SAT	**Interval-based CS**

Propagation:	contraction in lattice
	`{}`
	`{false}` `{true}`
	`{false,true}`
	of Boolean intervals
		contraction in lattice of intervals over $\mathbb{R}$

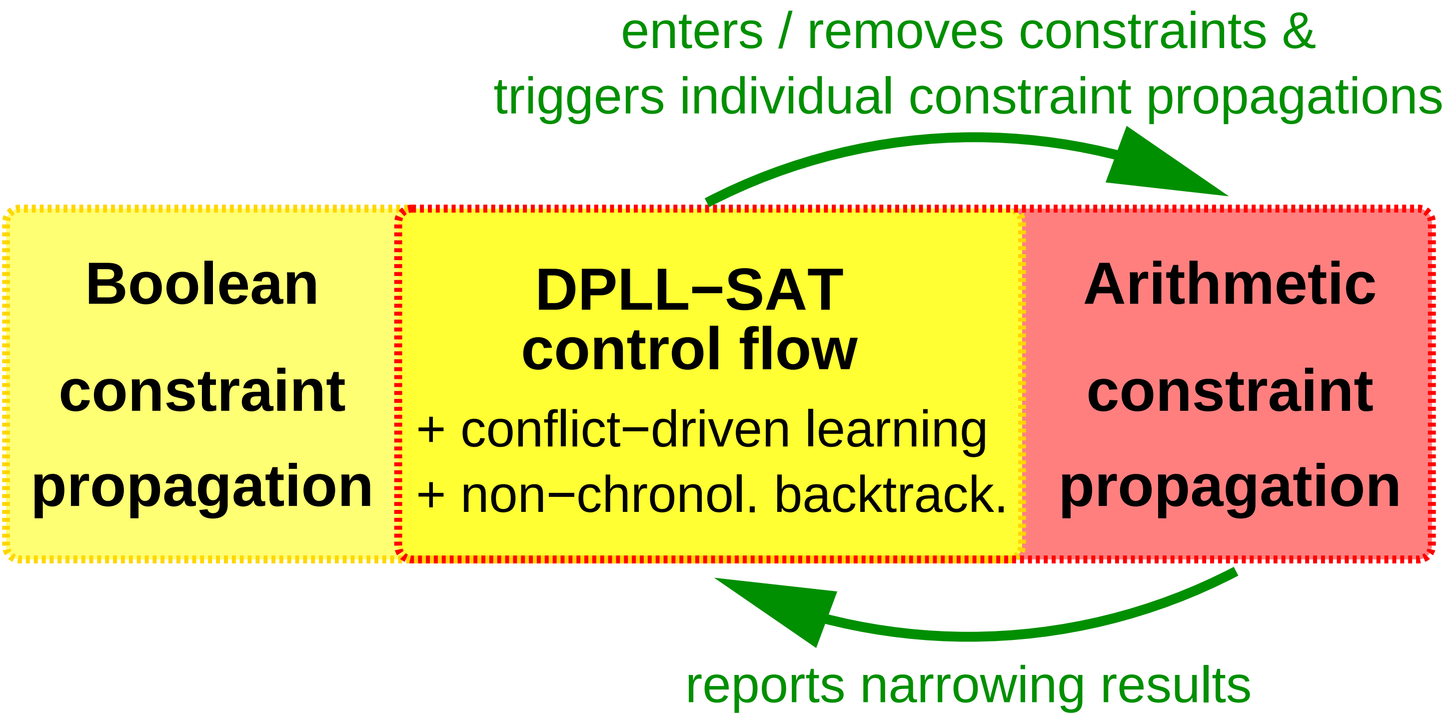
Split:	split of Boolean interval $[\texttt{false}, \texttt{true}]$	split of interval over $\mathbb{R}$

This suggests a tighter integration than lazy TP:
common algorithms should be shared, others should be lifted to both domains.

Lazy TP: Tightening the Interaction



Properties of Modified Layout



- SAT engine has introspection into CP
- thus can keep track of inferences *and their reasons*

can use recent SAT mechanisms for generalizing reasons of conflicts and learning them, thus pruning the search tree

Optimizations inherited from modern prop. SAT:

- conflct-driven learning
- non-chronological backtracking
- watched literal scheme
- restarts

$\longrightarrow$ have been instrumental to thousand-fold increase in tractable formula size for prop. SAT.

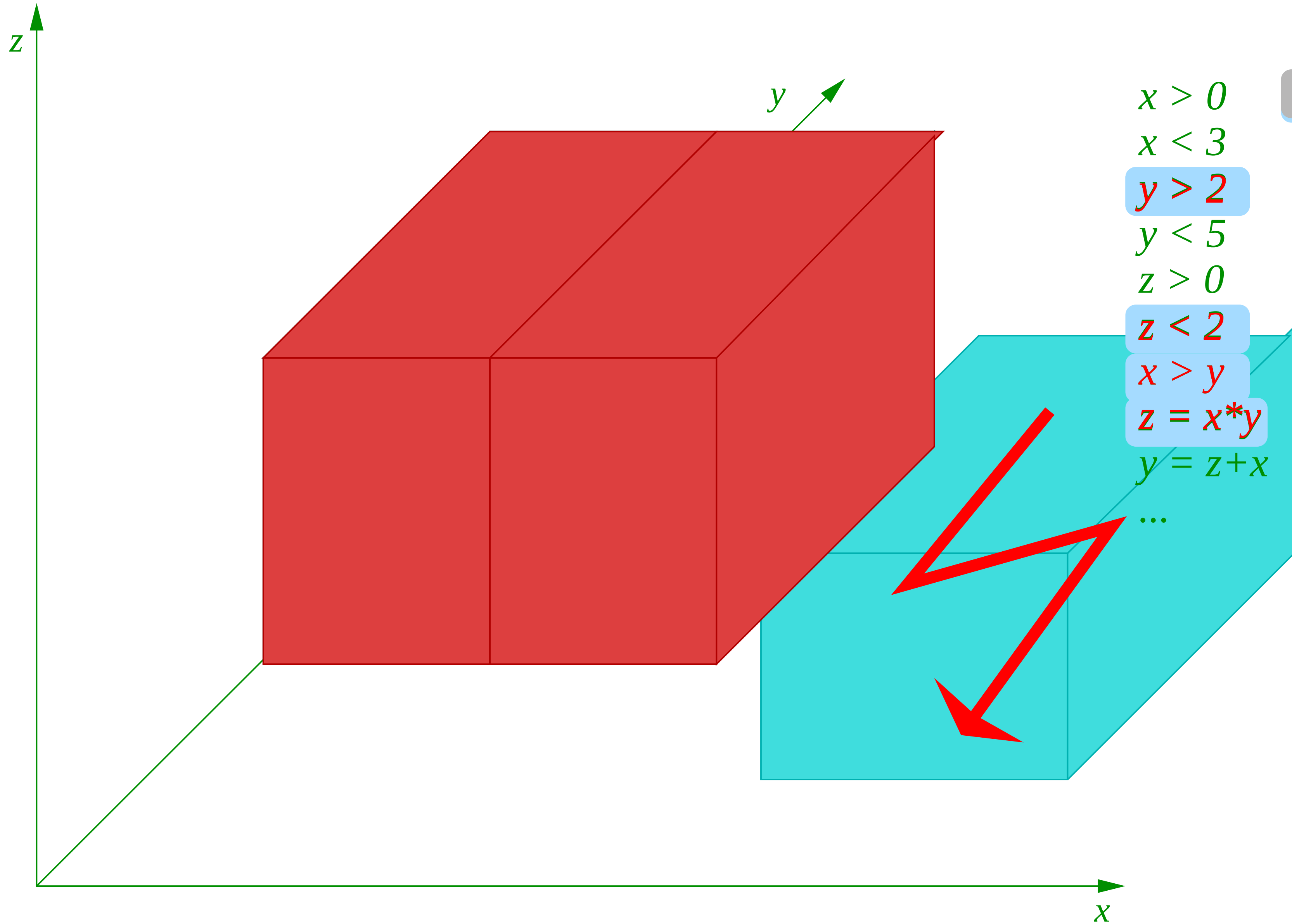
Conflict-driven learning in multi-valued case

Works like a charme w/o fundamental modifi cations:

- Decision variables coincide to interval splits;
  the assigned values to asserted bounds $x \geq c$, $x > c$, $x < c$, $x \leq c$;
- Implications correspond to contractions;
- Reasons to sets of asserted atoms giving rise to a contraction.

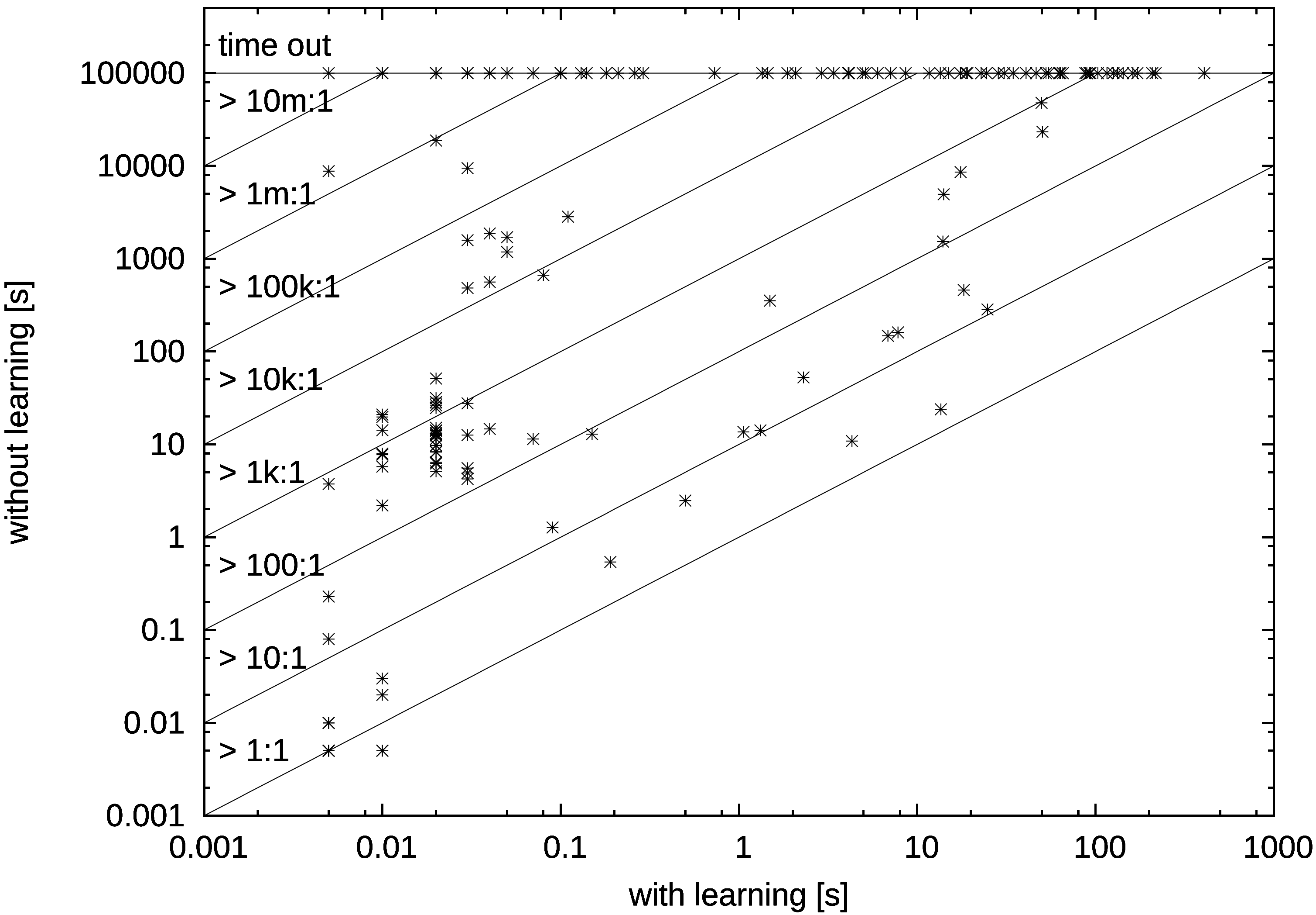
> Through embedding into SAT, we get conflict-driven learning and non-chronological backtracking for free!

Deduction and Learning



Refutes other candidate boxes and constraint combinations immediately.

The impact of learning: runtime



Examples:
BMC of

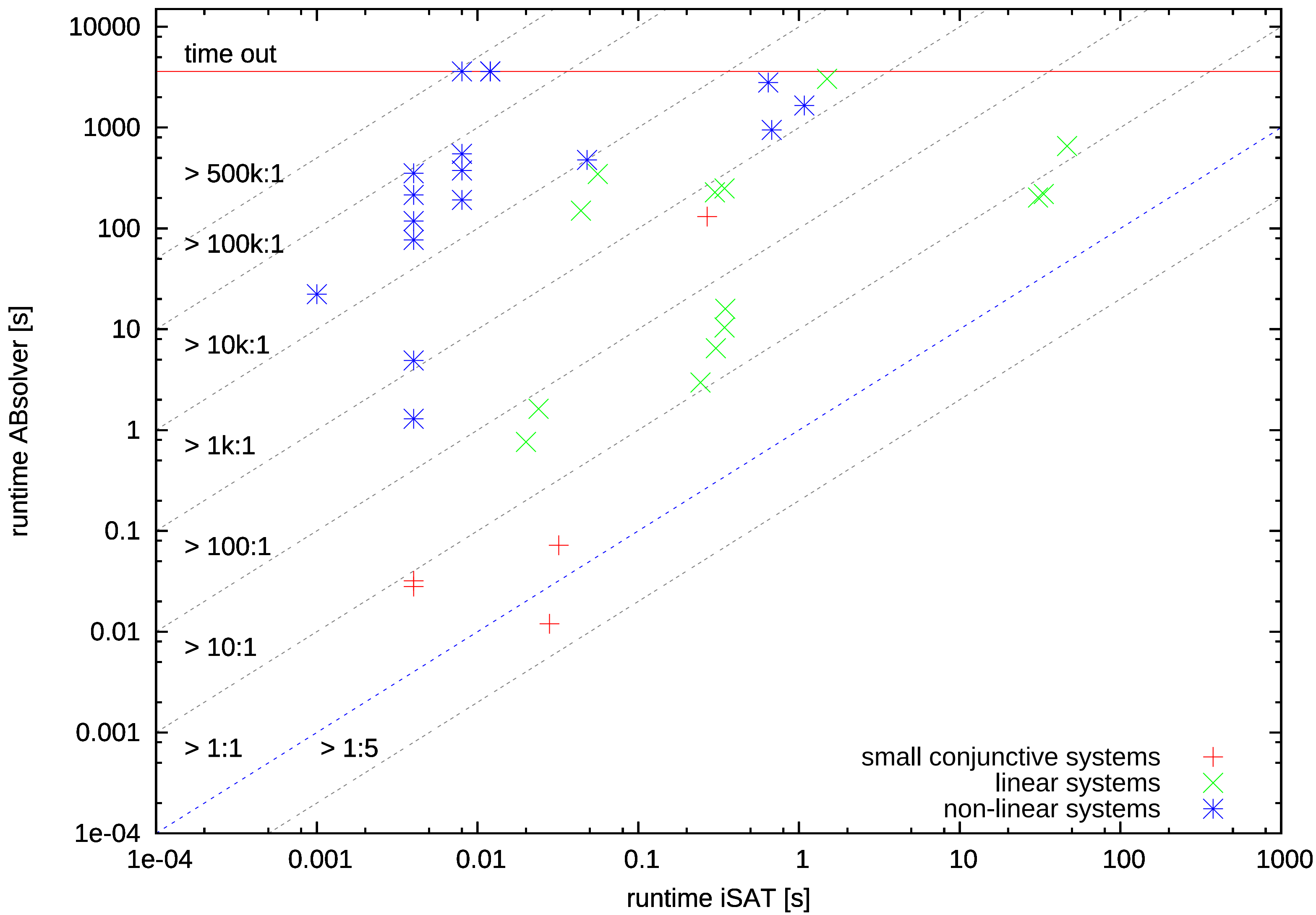
- platoon ctrl.
- bounc. ball
- gingerbread map
- oscillatory logistic map

Intersect. of geometric bodies

Size:
Up to 2400 var.s, $\gg$ 10^3 Boolean connectives.

[2.5 GHz AMD Opteron, 4 GByte physical memory, Linux]

The competition: ABsolver



ABsolver: Bauer, Pister, Tautschnig, "Tool support for the analysis of hybrid systems and models", DATE '07

Discussion

Approach: Unification of ICP-based constraint solving and DPLL-based propositional SAT solving in order to

- maintain the excellent reasoning power of ICP for robust constraints over $\mathbb{R}$,
- boost the performance on complex Boolean compositions of constraints

[Fränzle, Herde, Ratschan, Schubert, Teige 2006/07]

First experimental results:

- conflict-driven learning and other SAT optimizations of ICP yield enormous pruning of proof tree

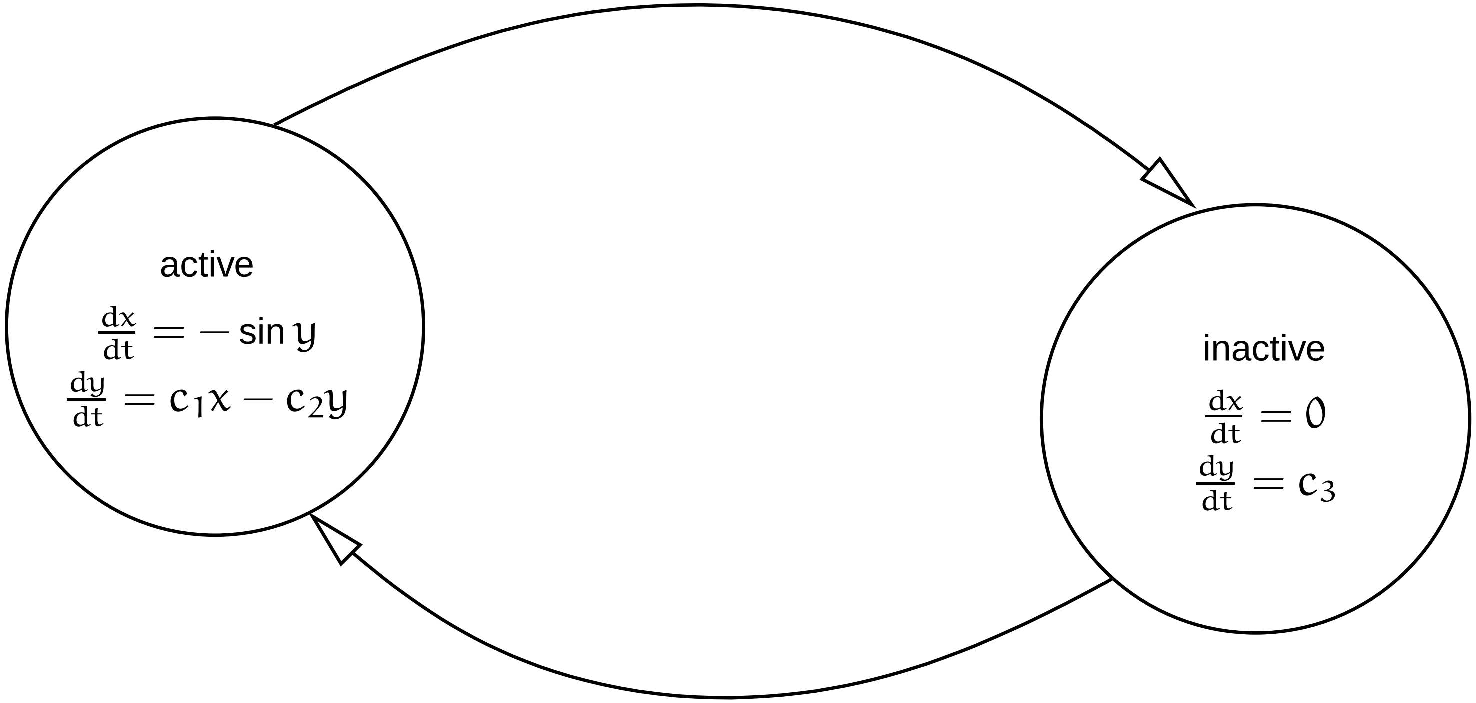
$\Rightarrow$ corresponding growth in size of tractable formulae

Consequences:

- can solve large boolean combinations of non-linear arithmetic constraints:
  - non-linear time-discrete hybrid systems
    (no differen*tial* equations, only differen*ce* equations)
  - appropriate hybridisations of ODEs
  - direct support for ODEs missing.

Direct reasoning over images and pre-images of ODEs

Motivation



- Linear and non-linear ordinary Differential Equations (ODEs) describing continous behaviour in the discrete modes of a hybrid system
- Want to do BMC on these models w/o prior hybridisation

The Problem

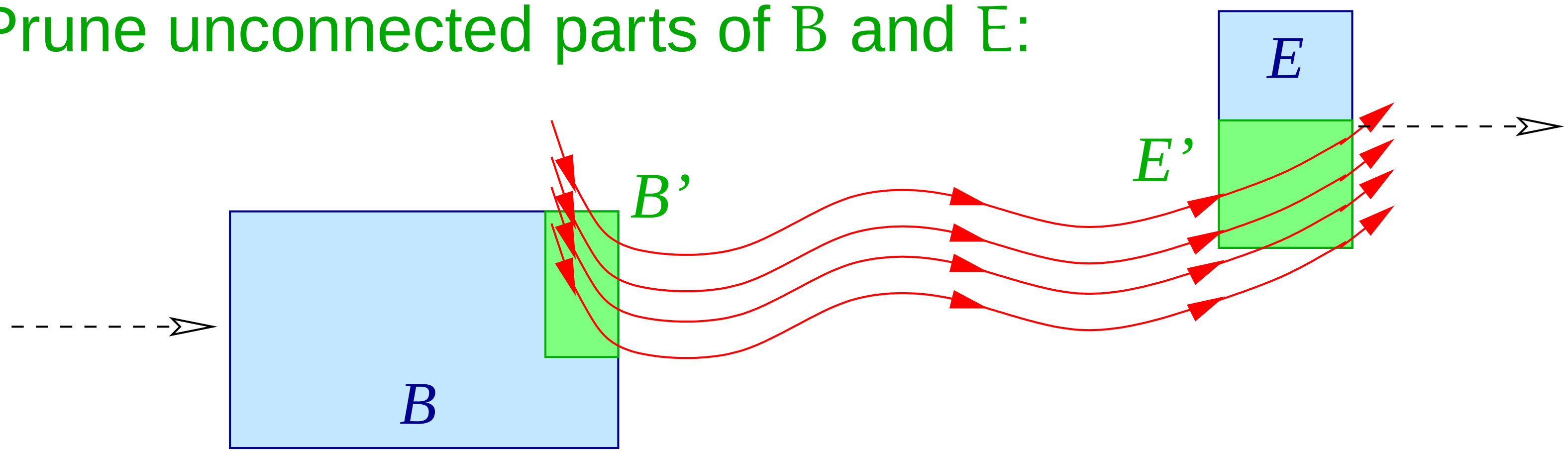
Given: a system of time-invariant ODEs

$$\frac{dx_1}{dt} = f_1(x_1, \ldots, x_n)$$
$$\vdots$$
$$\frac{dx_n}{dt} = f_n(x_1, \ldots, x_n)$$

plus three boxes $B, I, E \subset \mathbb{R}^n$.

Problem: determine whether E is reachable from B along a trajectory satisfying the ODE and not leaving I.

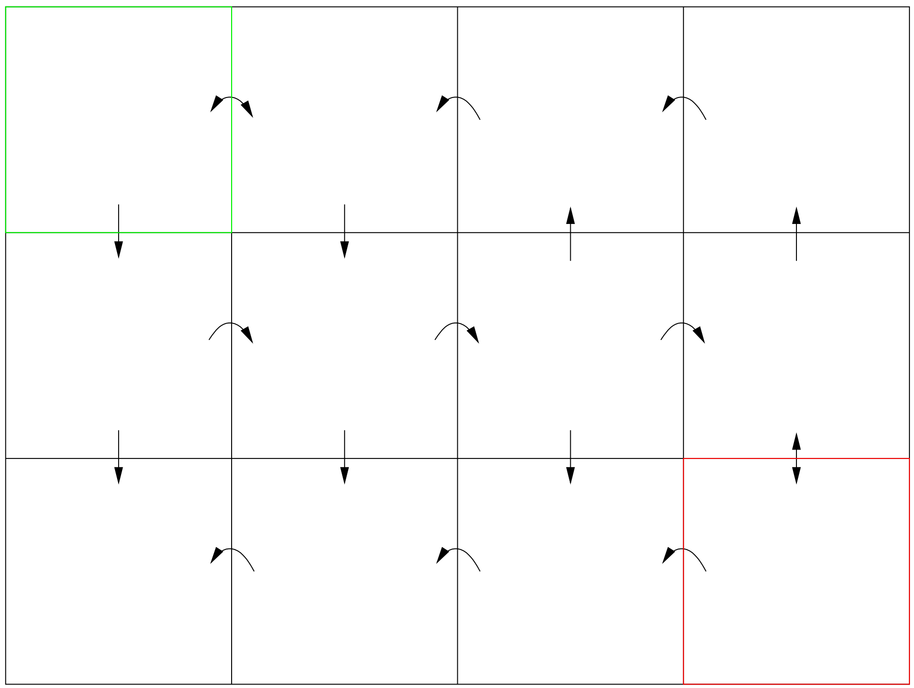
Added value: Prune unconnected parts of B and E:



Special case: adjacent boxes

Stursberg,Kowalewski et. al. [1997]:

Check sign of relevant derivative at box border:



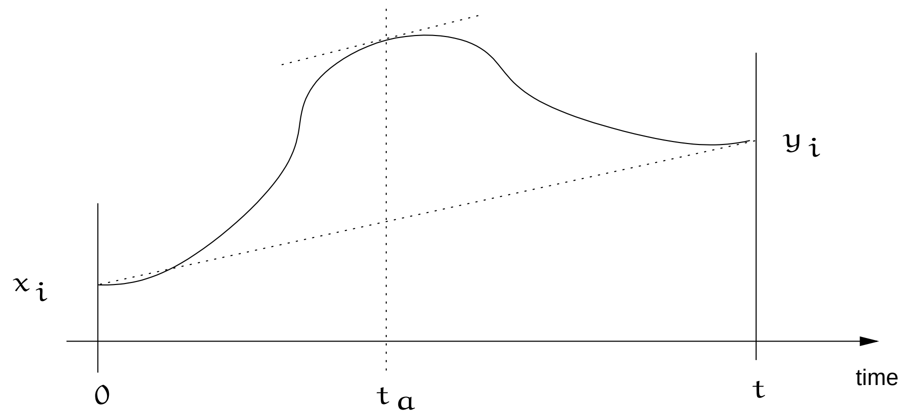
$\dot{x} \in [-5, 1]$

use interval arithmetic for evaluating the ODE over the box border.

Towards Pre-Post-Constraints

Lemma (n-dimensional mean value theorem): If $(y_1, \ldots, y_n) \in E \cap I$ is reachable from $(x_1, \ldots, x_n) \in B \cap I$ via a fbw in I satisfying $\frac{d\mathbf{x}}{dt} = \mathbf{f}$ then

$$\exists t \in \mathbb{R}_{\geq 0} : \bigwedge_{1 \leq i \leq n} \exists \mathbf{a} \in I : y_i = x_i + f_i(\mathbf{a}) \cdot t$$



HSolver [Ratschan, 2004–]

Problem: Safely determine whether E is unreachable from B along a trajectory satisfying the ODE and not leaving I.

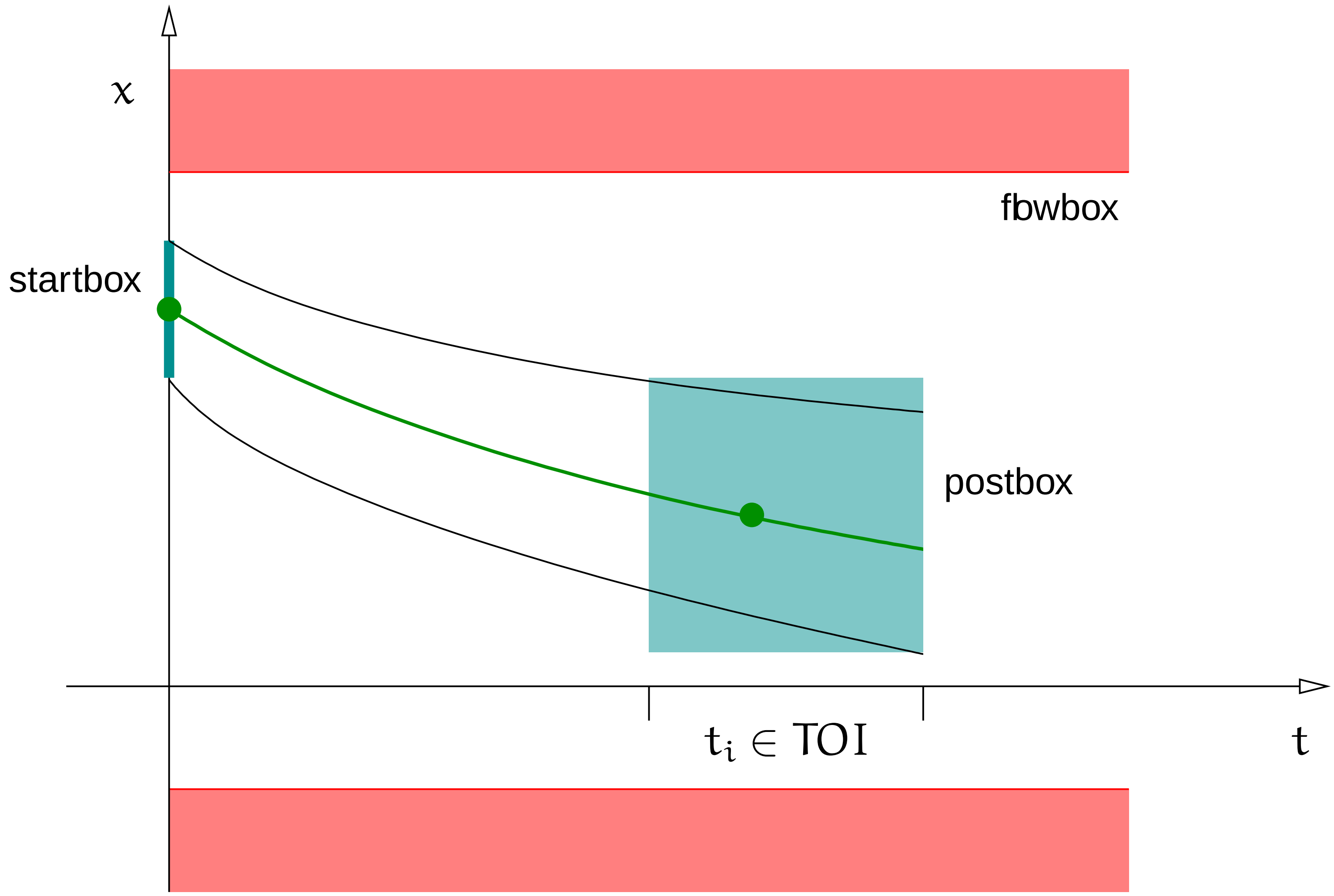
Some approaches:

1. Interval-based safe numeric approximation of ODEs [Moore 1965, Lohner 1987, Stauning 1997]

   (used in Hypertech [Henzinger, Horowitz, Majumdar, Wong-Toi 2000])

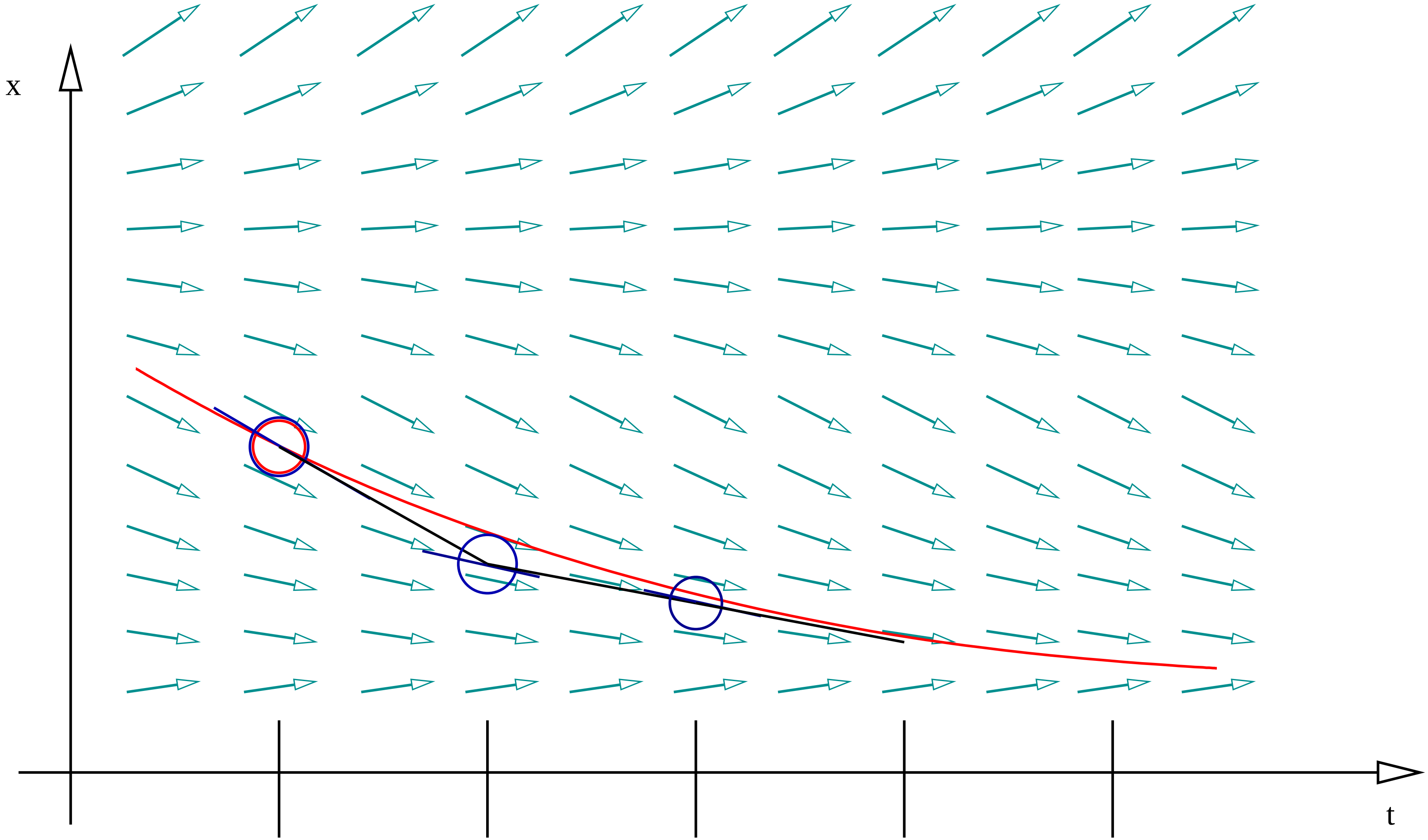
2. CLP(F): a symbolic, constraint-based technology for reasoning about ODEs grounded in (in-)equational constraints obtained from Taylor expansions [Hickey, Wittenberg 2004]

Safe Approximation



Should also be tight! And effi cient to compute!

Euler's Method



Taylor Series

Exact solution $x(t)$ has slope determined by f in each point:
$\frac{dx}{dt} = f(x(t))$
Taylor expansion of exact solution:

$$
\begin{aligned}
x(t_0 + h) =& x(t_0) + \frac{h^1}{1!}\frac{dx}{dt}(t_0) \\
&+ \frac{h^2}{2!}\frac{d^2x}{dt^2}(t_0) + \dots \\
&+ \frac{h^n}{n!}\frac{d^nx}{dt^n}(t_0) \\
&+ \frac{\mathbf{h}^{\mathbf{n}+\mathbf{1}}}{(\mathbf{n}+\mathbf{1})!}\frac{\mathbf{d}^{\mathbf{n}+\mathbf{1}}\mathbf{x}}{\mathbf{dt}^{\mathbf{n}+\mathbf{1}}}(\mathbf{t_0} + \theta\mathbf{h}), \text{ with } \mathbf{0} < \theta < \mathbf{1}
\end{aligned}
$$

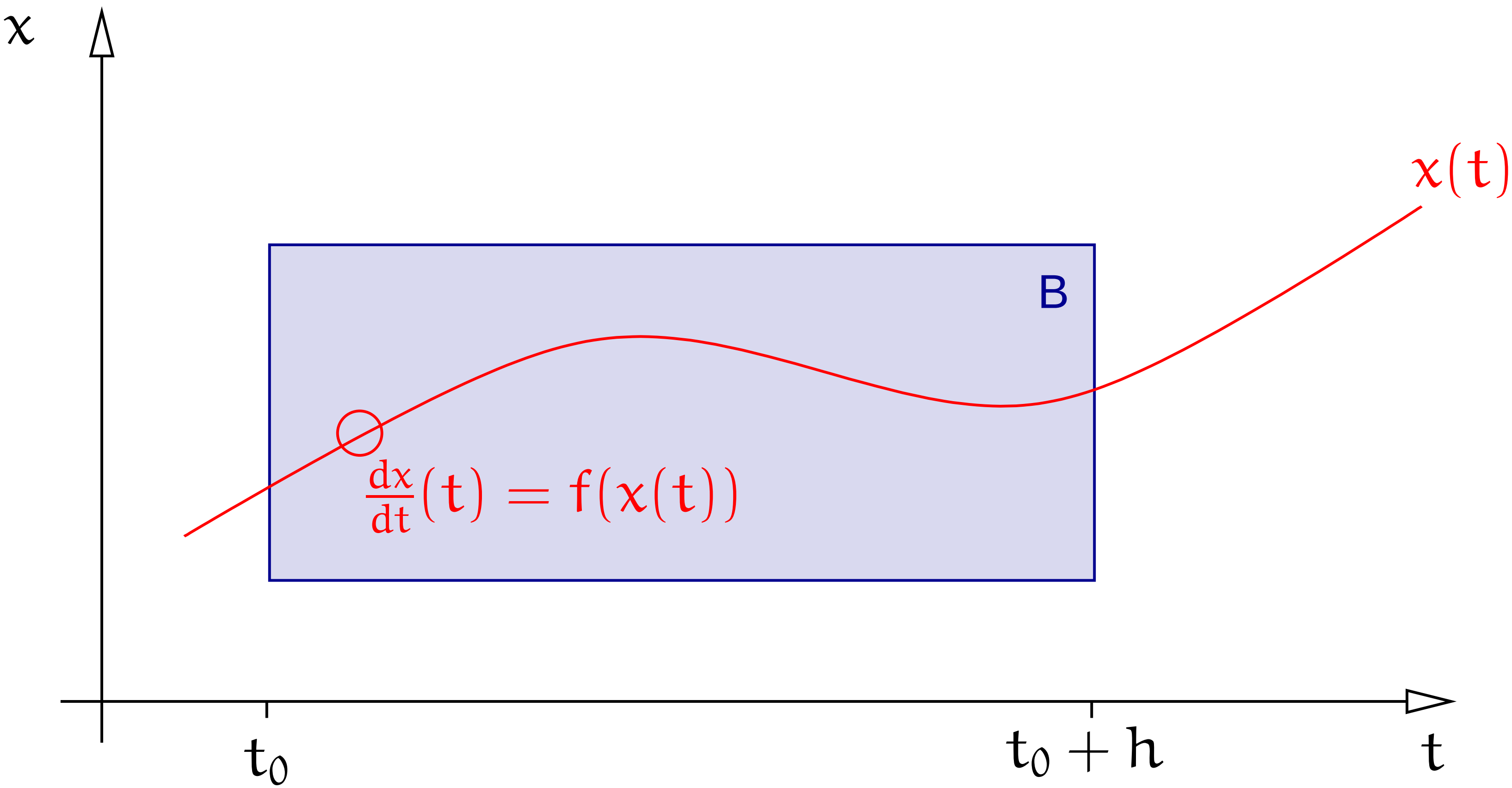
(LAGRANGE REMAINDER)

Taylor Series

$$
\begin{aligned}
x(t_0 + h) = & x(t_0) + \frac{h^1}{1!} \underbrace{\frac{dx}{dt}(t_0)}_{f(x(t_0))} \\
& + \frac{h^2}{2!} \underbrace{\frac{d^2x}{dt^2}(t_0)}_{\frac{df}{dt}(x(t_0)) \cdot f(x(t_0))} + \ldots \\
& + \frac{h^n}{n!} \frac{d^nx}{dt^n}(t_0) \\
& + \frac{h^{n+1}}{(n+1)!} \underbrace{\frac{d^{n+1}x}{dt^{n+1}}(t_0 + \theta h)}_{\text{unknown}}, \text{ with } 0 < \theta < 1
\end{aligned}
$$

Can use interval arithm. to evaluate $f(x(t_0))$, etc., if $x(t_0)$ is set-valued!

Bounding Box



$$\begin{aligned} \frac{dx}{dt}(t) &\leq \max(f(B)) \\ \frac{dx}{dt}(t) &\geq \min(f(B)) \end{aligned} \quad \text{for all } t \in [t_0, t_0 + h]$$

If we only knew B...

Bounding Box [Lohner]

Given: Initial value problem:

$$\frac{dx}{dt} = f(x), \quad x(t_0) = x_0 \text{ may also be a box}$$

Theorem (Lohner): If

$$[B^1] := u_0 + [0, h] \cdot f([B^0])$$

and

$$[B^1] \subseteq [B^0]$$

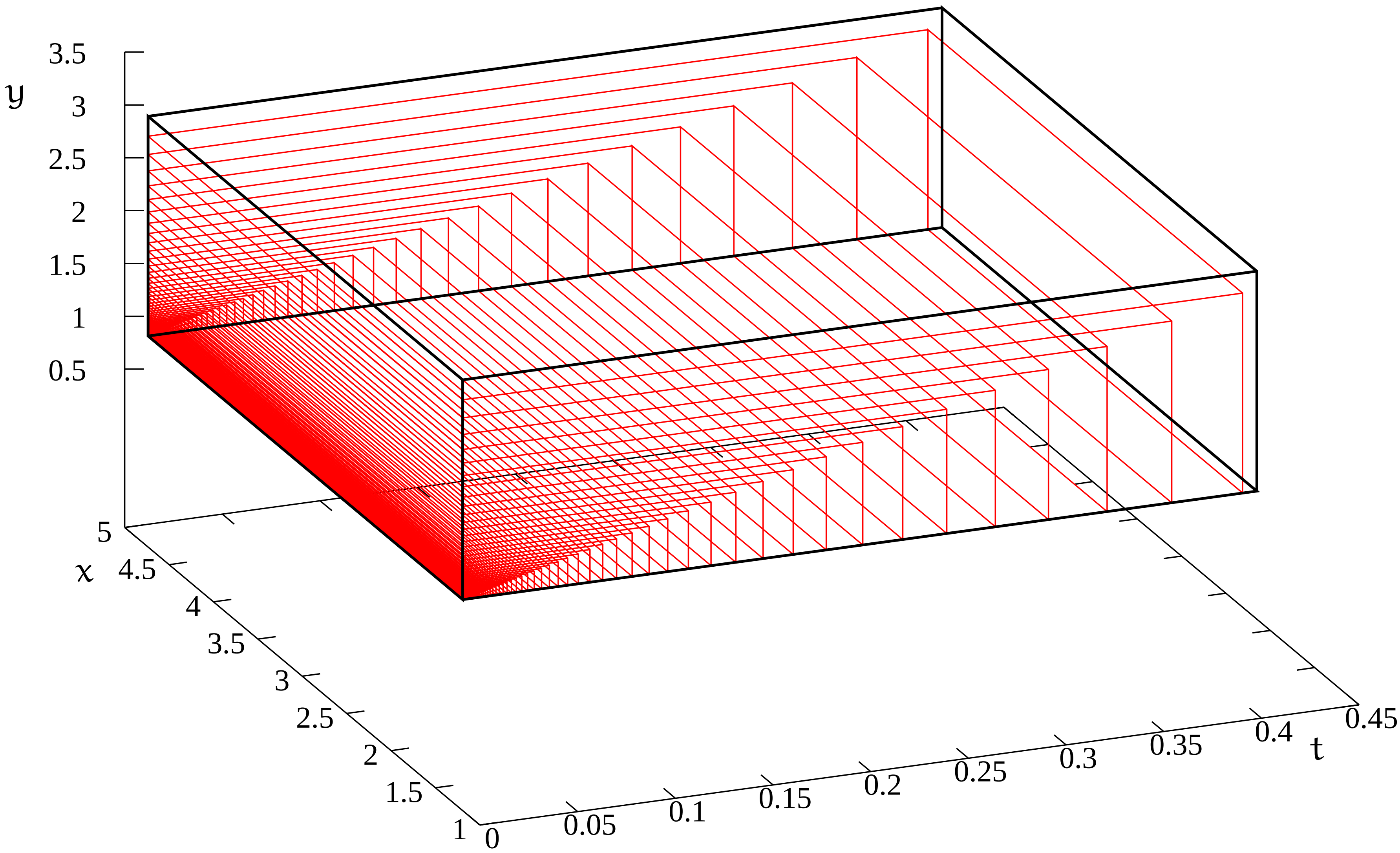
then the initial value problem above has exactly one solution over $[t_0, t_0 + h]$ which lies entirely within $[B^1] \rightarrow$ Bounding Box.

Algorithm

To get an enclosure . . .

- Determine bounding box and stepsize
- Evaluate Taylor series up to desired order over startbox
- Evaluate remainder term over bounding box

Bounding Box

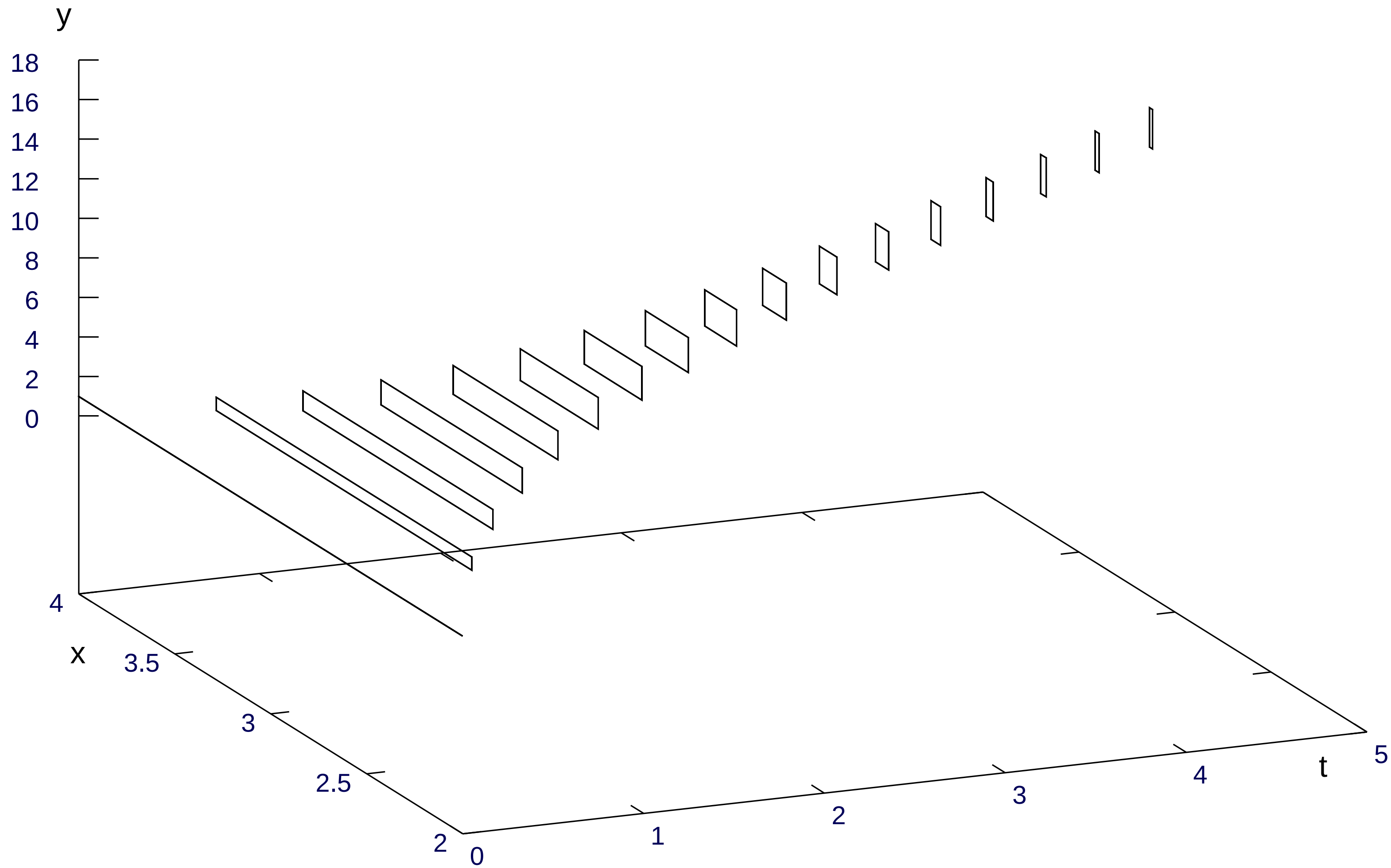


Algorithm

- Find **bounding box** with greedy algorithm
- Generate **derivatives** symbolically
- Simplify expressions to reduce alias effects on variables
- **Evaluate expressions** with interval arithmetic
  - Taylor series
  - Lagrange remainder

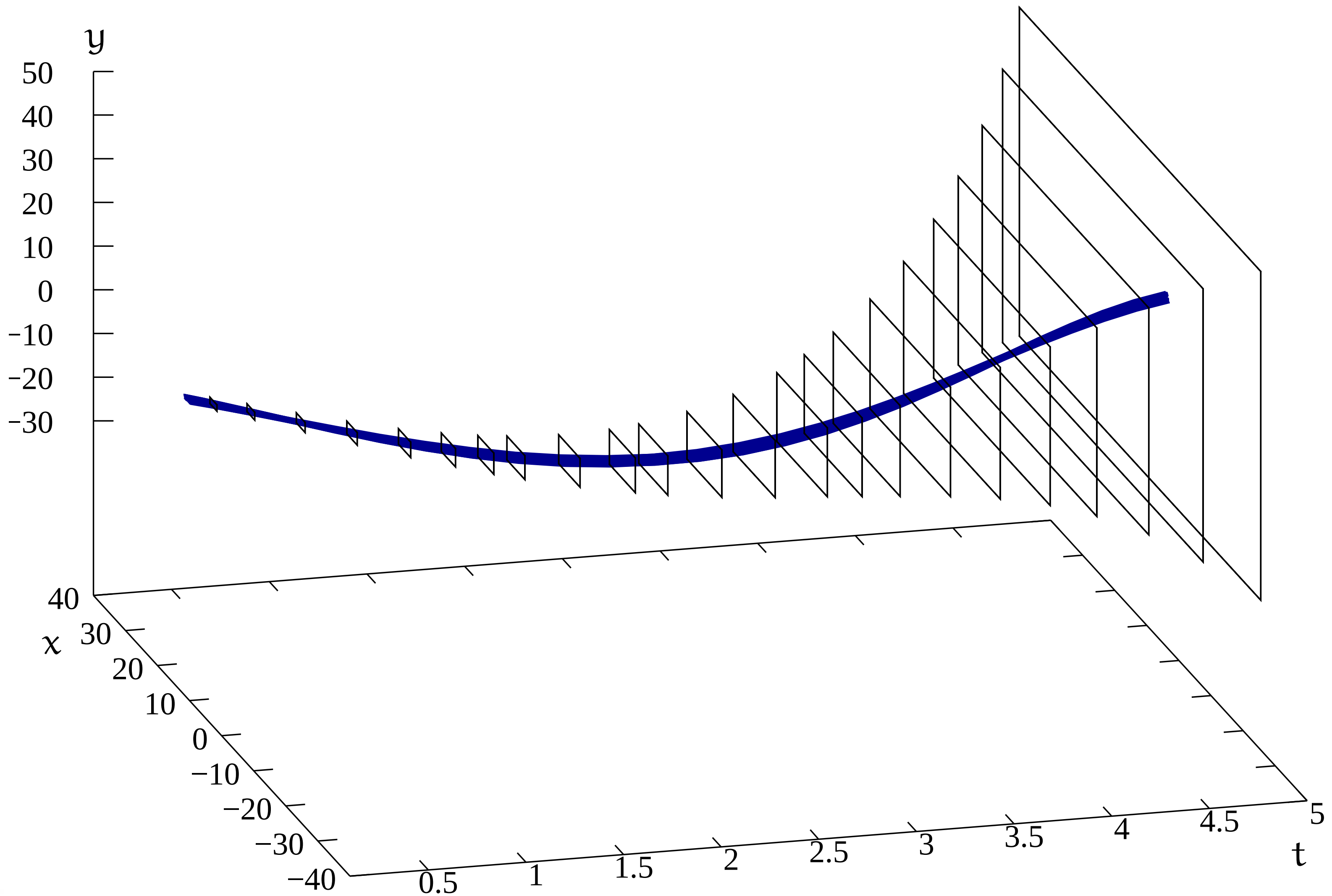
Example

$$\frac{dx}{dt} = -x + 3, \; \frac{dy}{dt} = x, \; x_0 = [2, 4], \; y_0 = [1, 1]$$



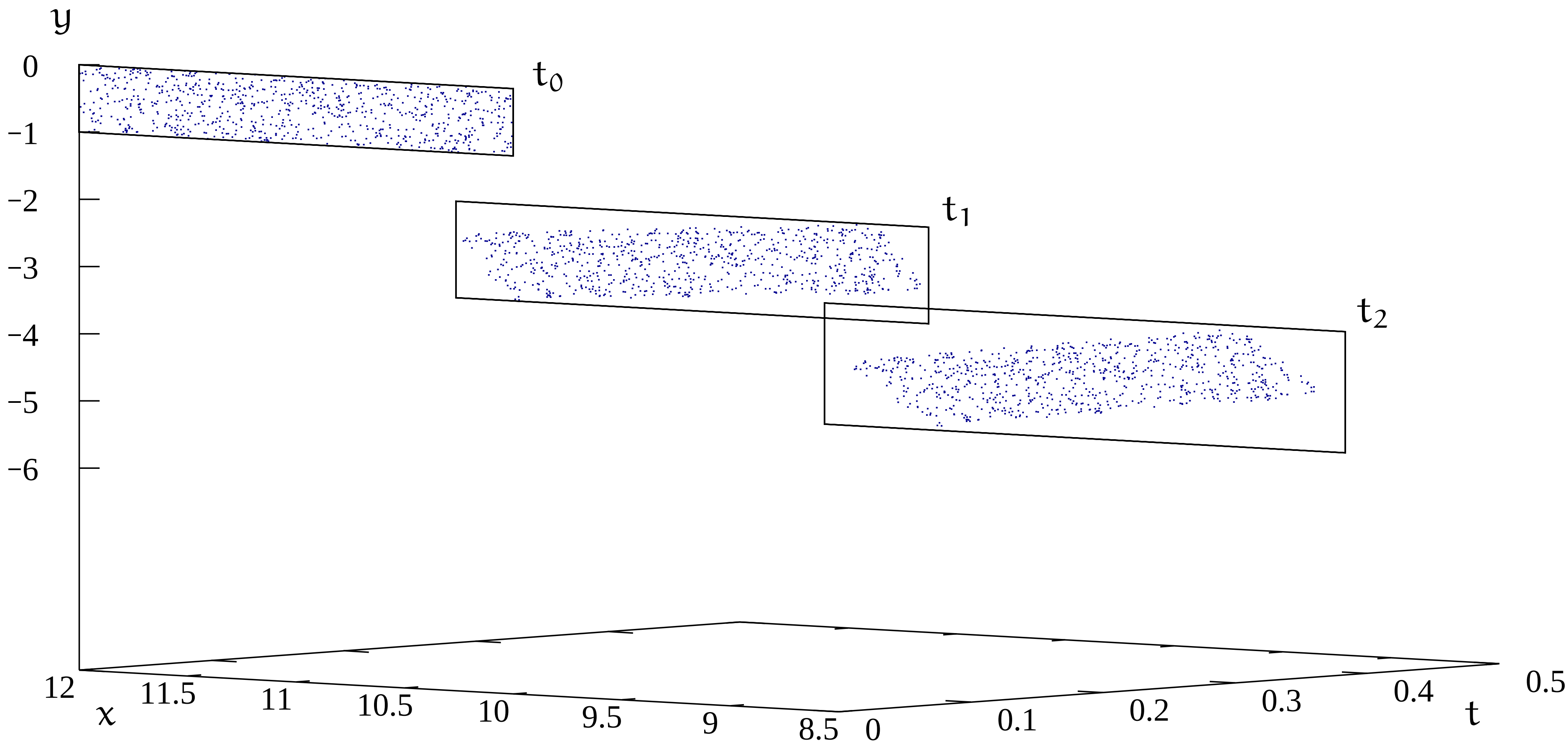
Example II: Stable Oscillator

$$\frac{dx}{dt} = y,\ \frac{dy}{dt} = -x,\ x_0 = [10, 12],\ y_0 = [-1, 0]$$



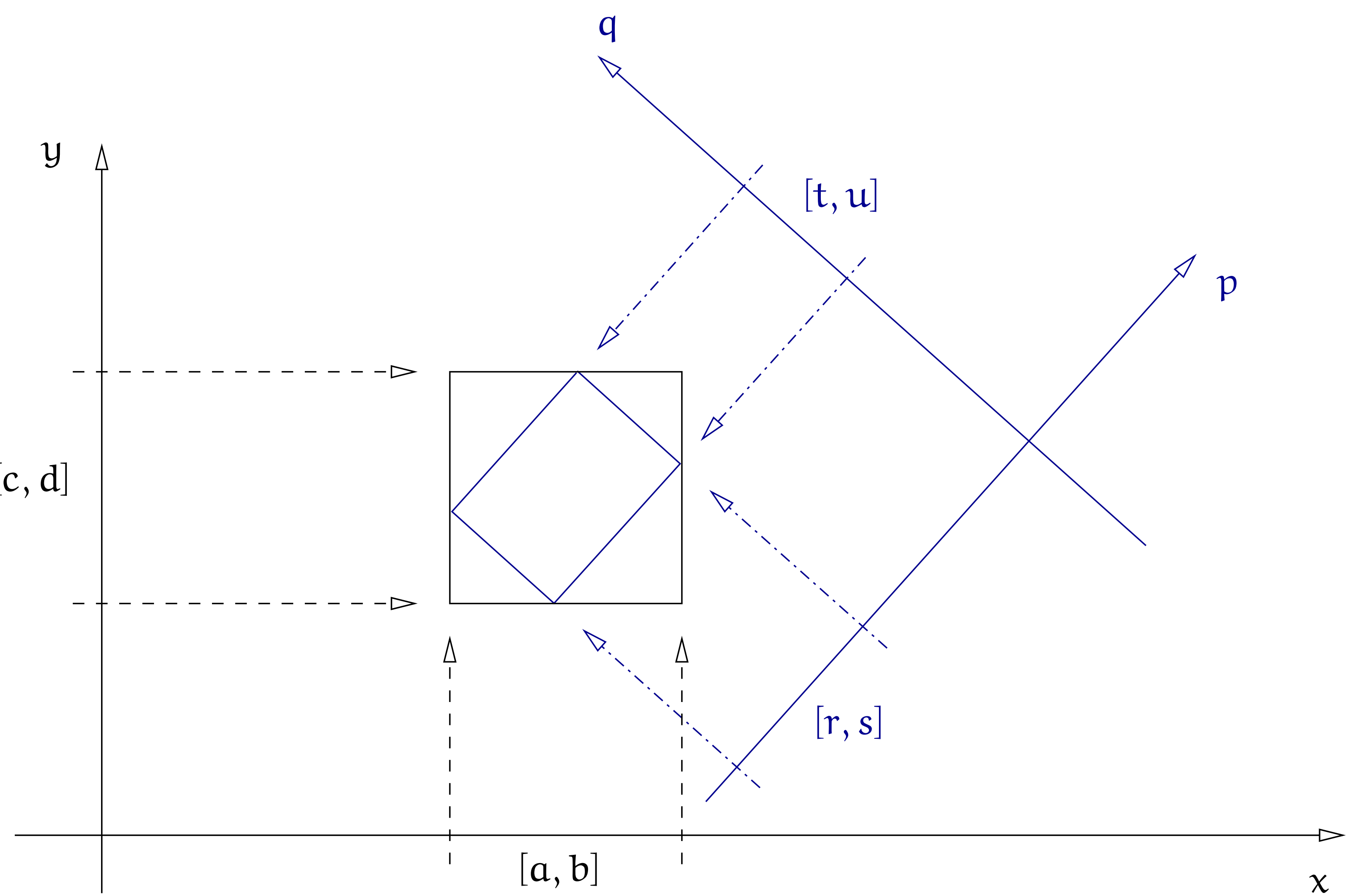
Wrapping Effect

$$\frac{dx}{dt} = y, \; \frac{dy}{dt} = -x, \; x_0 = [10, 12], \; y_0 = [-1, 0]$$



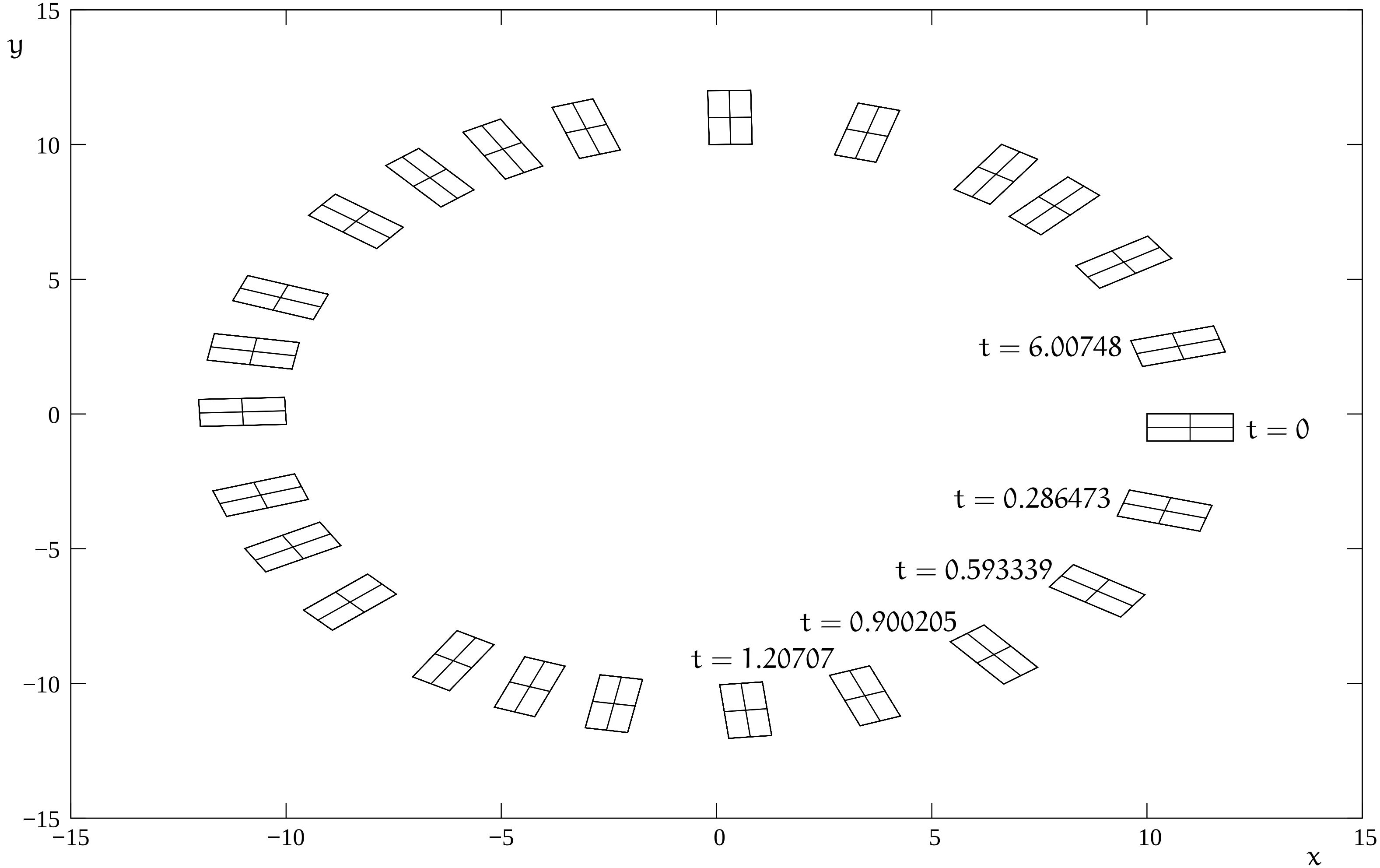
Fight Wrapping Effect

Lohner, Stauning, . . . : use **coordinate transformation**



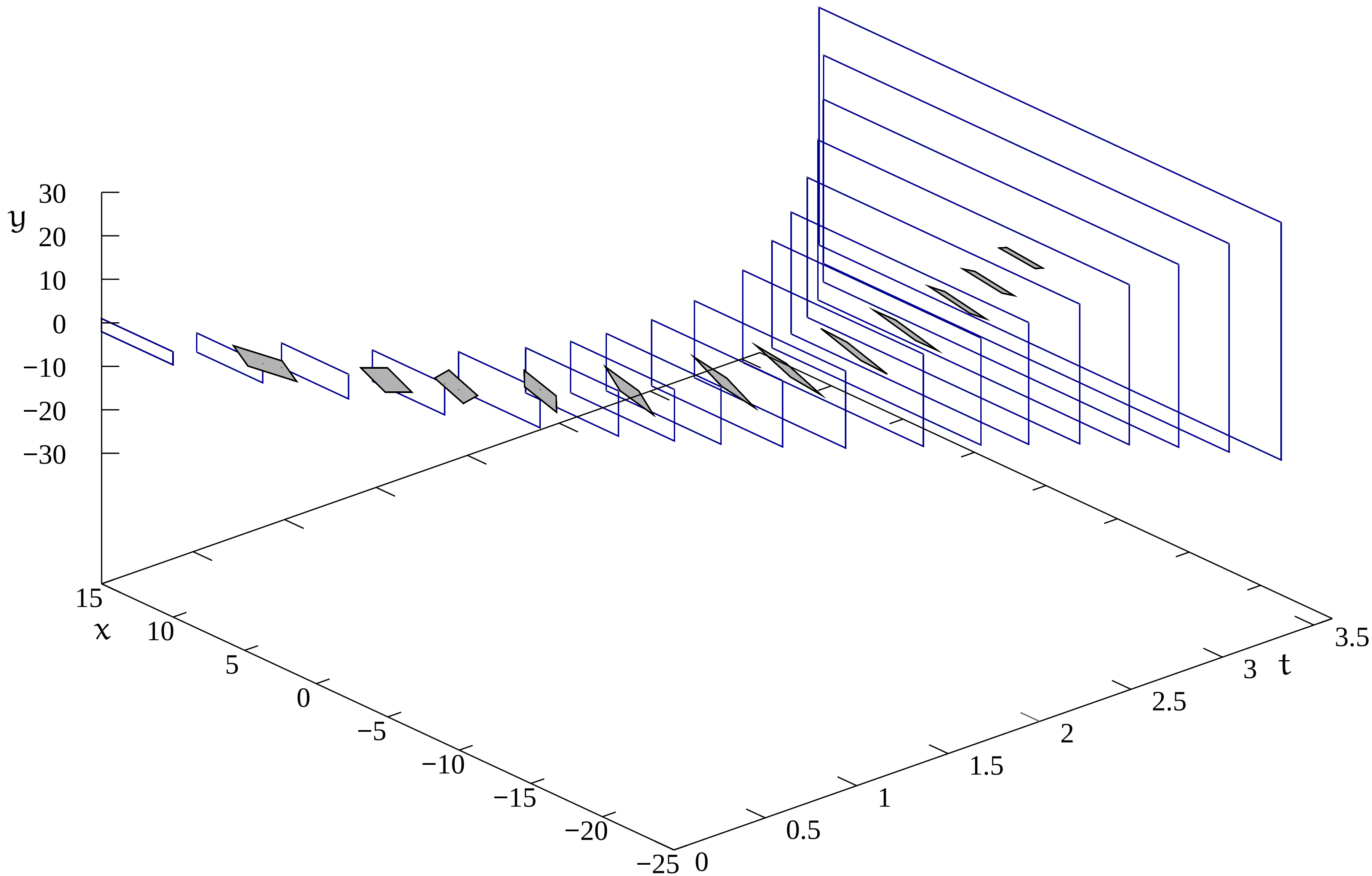
Stable Oscillator

$$\frac{dx}{dt} = y,\ \frac{dy}{dt} = -x,\ x_0 = [10, 12],\ y_0 = [-1, 0]$$

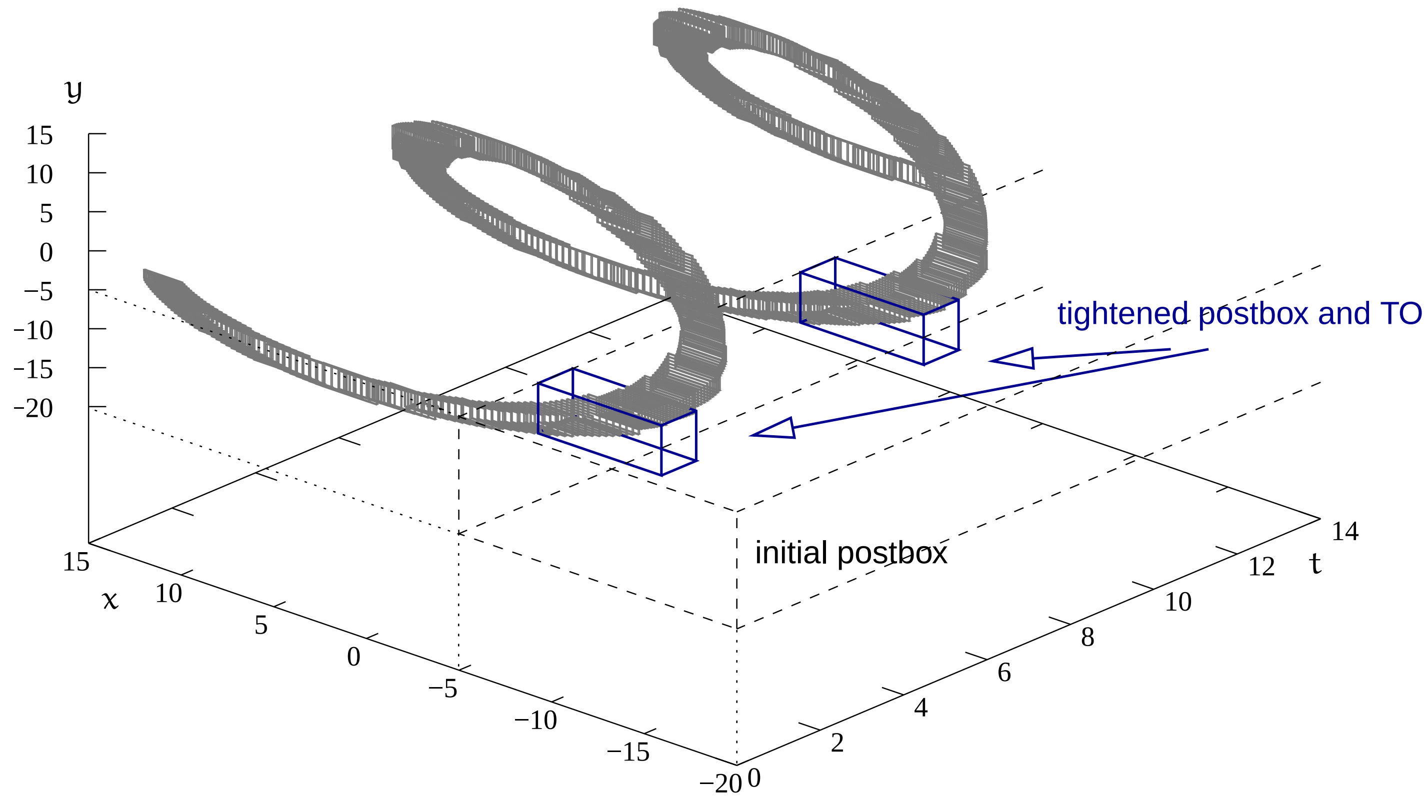


Damped Oscillator

$$\frac{dx}{dt} = y - 0.8 \cdot x,\ \frac{dy}{dt} = -x + 0.3 \cdot y,\ x_0 = [10, 15],\ y_0 = [-2, 1]$$



Use in ICP: Tighten Target Box



- Given target box (including phase space and time)
- Intersect target box with enclosure
- Remove elements with empty intersection (narrows also time-window of interest)

Backward Propagation

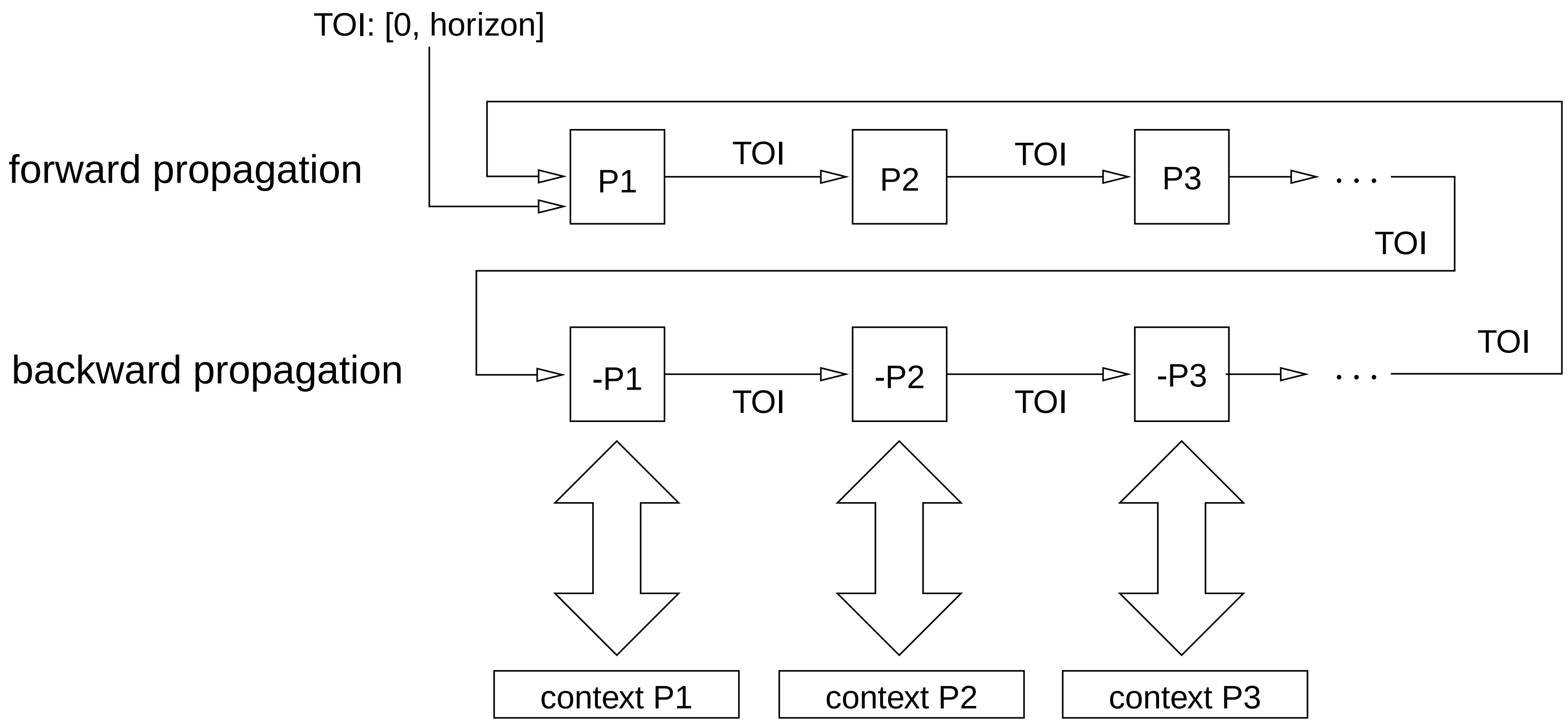
- Use temporally reversed ODEs
- Use start box as target box and do normal forward propagation
- Intersect resulting target box with original start box

Fwd. and bwd. propagation do

- narrow the start box $\mathtt{B}$ and target box $\mathtt{E}$ — also iteratively!
- narrow the time window for both $\mathtt{B}$ and $\mathtt{E}$,
- thus give fresh meat to constraint propagation along adjacent parts of the transition sequence!

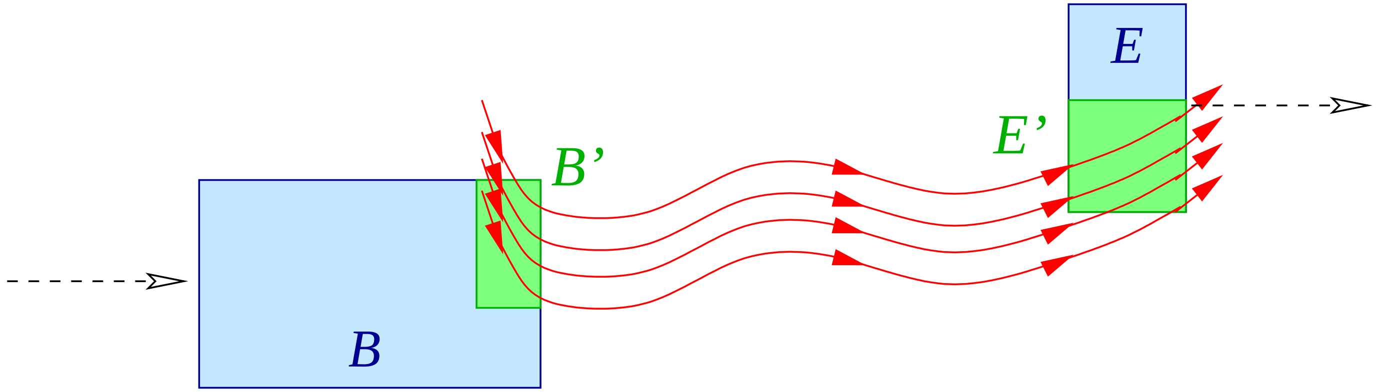
Controlling Complexity: Partitioning

- Partition ODEs: Group together ODEs with common variables
- Deduction process alternates between different partitions and between forward and backward pruning:



Summary

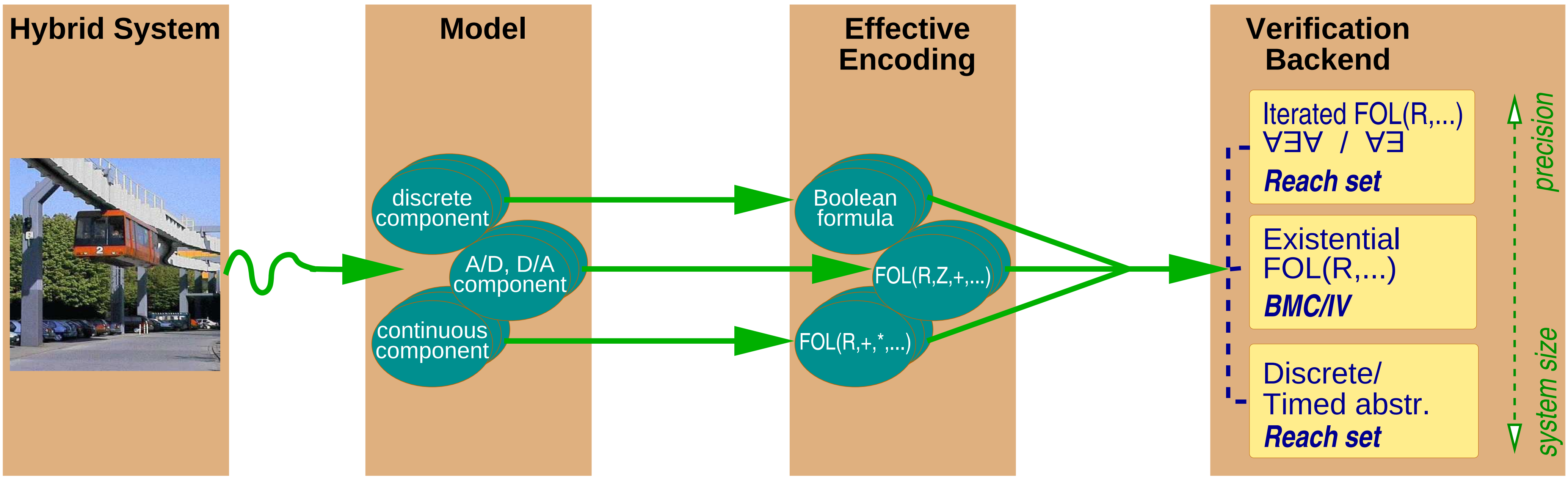
- Taylor-based numerical method with error enclosure
- Tightly integrated with non-linear arithmetic constraint solving:
  - provides an interval contractor, just like ICP



  - temporally symmetric (fwd. and bwd. contraction), unlike traditional image computation
  - refutes trajectory bundles based on partial knowledge
- experimental: fi rst proof-of-concept implemented.

Summary

Verification Flow



Strictly symbolic approach,
exemplified on an SMT-based tool set.

Summary

- These were just some appetizers shedding light on principles.
- Haven't touched major topics in hybrid systems, e.g.
  - Data structures (and related image computation procedures) for more precise representation of images:
    - polytopes (e.g., [Henzinger, Ho, Wong-Toi 1995, Chutinan, Krogh 1998, Frehse 2005]), zonotopes [Girard 2005, Girard, le Guernic, Maler 2006, ...], ellipsoids [Kurzhanski, Varaiya 2000], level sets of functions [Tomlin], ...
    - AIG(LP) [Damm et al. 2006], hybrid restriction diagrams [Wang 2004], ...
  - Stability theory
    - Lyapunov and Lyapunov-like functions
    - discharging the related proof obligations; synthesizing these witness functions

to name just a few.

Perspectives for researchers

- Approximation theories and decidability issues
  - Safe approximation is essential; under which circumstances do they provide decision procedures; what are the appropriate notions of approximate decision?
  - Robust systems and “almost decidability” [Fränzle 1999, Asarin, Bouajjani 2001, Collins 2006, Platzer, Clarke 2006, Girard, Pappas 2006, Girard 2007]
- Scalability and performance issues
  - All current algorithms are quite confi ned
  - Massively branching behavior of non-deterministic hybrid systems together w. intricate continuous dynamics
  - Better algorithms and data structures; maybe tailored to specifi c analysis goals and system types
- Modeling issues
  - Adequate modeling languages for the variety of hybrid phenomena
  - Currently, most modeling is simulation-oriented
  - Languages should concisely model system dynamics (including non-determinism, probabilism, etc., were adequate) and the input domain of open systems (shapes of inputs, controllability attributes, ...)

Thanks

- to the collaborators within AVACS project area hybrid systems:

  A. Eggers[a], A. Mikschl[a], A. Platzer[a], A. Podelski[b], A. Rybalchenko[b], B. Badban[a], B. Becker[b], B. Nebel[b], B. Westphal[a], B. Wirtz[a], C. Herde[a], C. Scholl[b], E. Abraham[b], E.-R. Olderog[a], F. Eisenbrand[d], F. Klaedtke[f], F. Pigorsch[b], H. Burchhardt[a], H. Dierks[a], H. Hermanns[c], H. Hungar[a], I. Polian[b], J. Eisinger[b], J. Oehlerking[a], J.-G. Smaus[b], Jun Pang[a], M. Behle[d], M. Herbstritt[b], M. Lewis[b], M. Swaminathan[a], N. Kalinnik[b], O. Theel[a], P. Maier[d], R. Wimmer[b], S. Disch[b], S. Jacobs[d], S. Kupferschmied[b], S. Ratschan[e], S. Wagner[b], T. Schubert[b], T. Teige[a], U. Waldmann[d], V. Sofronie-Stokkermanns[d], W. Damm[a], Zhikun She[d]

- and to the contributing institutions:

  [a] Carl von Ossietzky Universität Oldenburg, Germany

  [b] Albert-Ludwigs-Universität Freiburg, Germany

  [c] Universität des Saarlandes & [d] MPII, Saarbrücken, Germany

  [e] Academy of Sciences of the Czech Republic, Prague, Czech Rep.

  [f] Eidgenössische Technische Hochschule Zurich, Switzerland

- and to Deutsche Forschungsgemeinschaft for funding AVACS.