Solving Initial-State Non-Determinism in Stochastic Linear Hybrid Automata with Underapproximative Backwards Refinement

Master Thesis Defense

Yvonne Heimowski
Supervision: József Kovács

LuFG Theory of Hybrid Systems

SS 2026

RWTH AACHEN UNIVERSITY

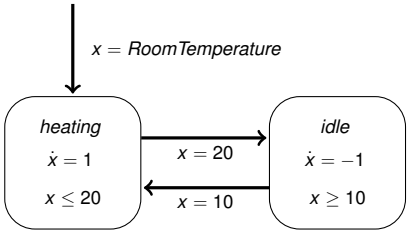
Contents

Hybrid Systems

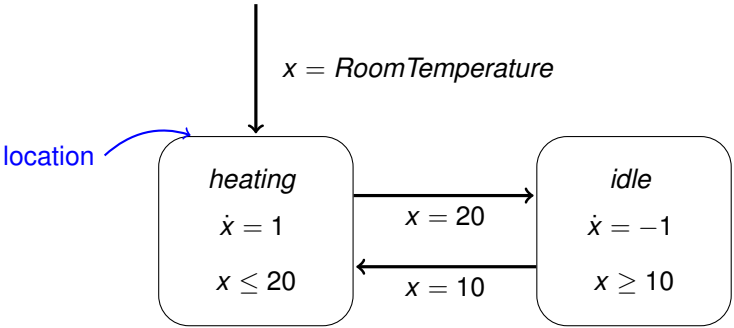
- Combine
  - Continuous Behavior
  - Discrete State Changes
- Example: Thermostat
  - Continuous Behavior: Temperature Rises and Falls
  - States: Heating (Temperature Rises), Idle (Temperature Falls)
  - State Changes: Switch between States when minimum resp. maximum temperature is reached

Hybrid Automata

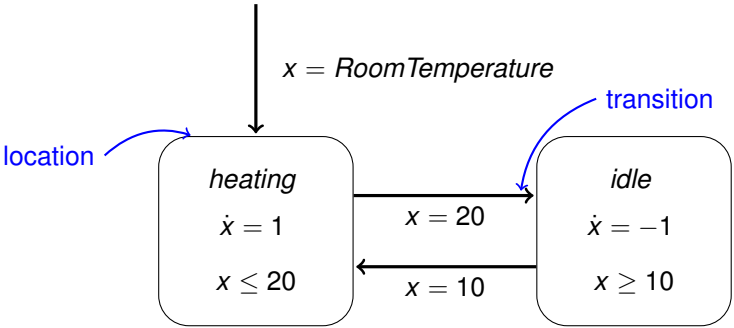
- Represent Hybrid Systems
- Model behavior and states



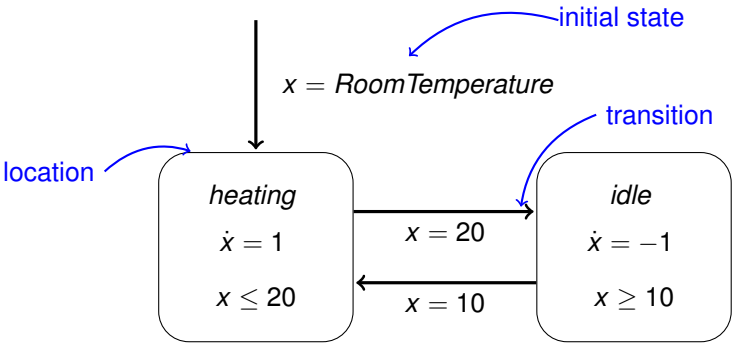
Hybrid Automata



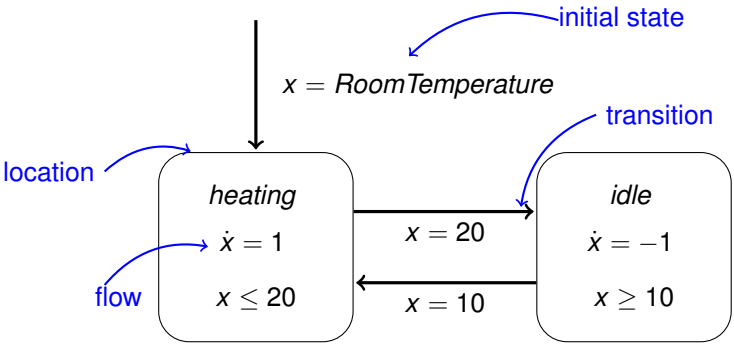
Hybrid Automata



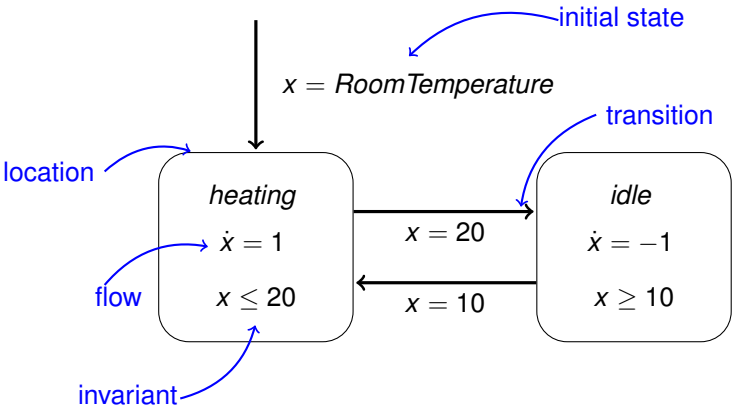
Hybrid Automata



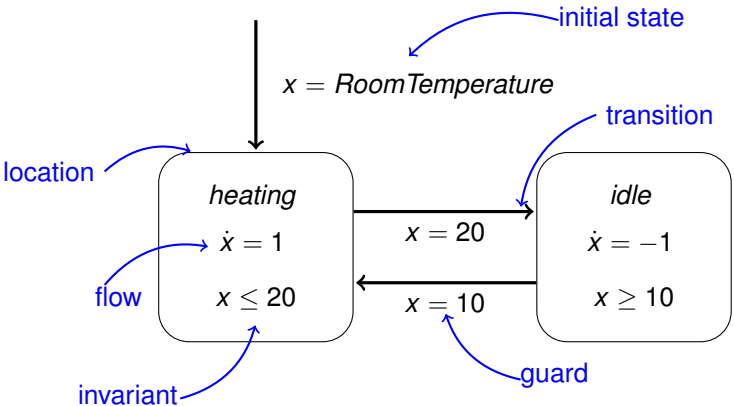
Hybrid Automata



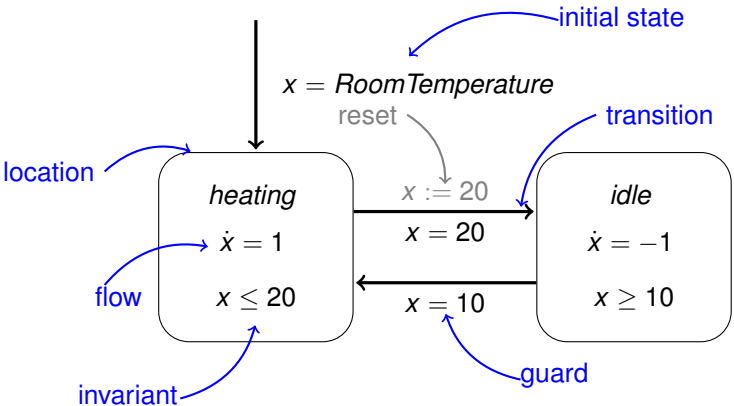
Hybrid Automata



Hybrid Automata



Hybrid Automata



Variants of Hybrid Automata

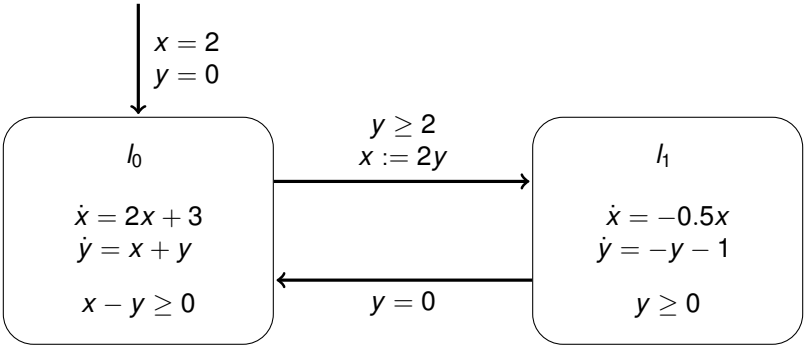
Rectangular Hybrid Automata:

- Uses Rectangular Sets for invariants, guards, flows and resets
- [a,b] represents the rectangular set defined as $\{x \mid a \leq x \leq b\}$

Linear Hybrid Automaton (LHA) (with non-linear behavior):

- Invariants, guards defined as systems of linear inequalities
- Flows defined as systems of linear Ordinary Differential Equation (ODE) : $\dot{x} = Ax + b$
- Resets are defined as affine mappings of the form $x' = Ax + b$

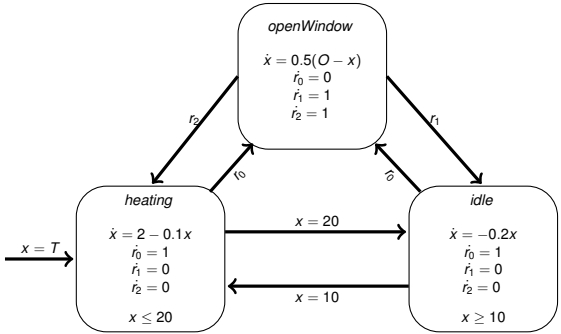
Linear Hybrid Automata



Stochastic Hybrid Automata

O = Outside Temperature
T = Room Temperature

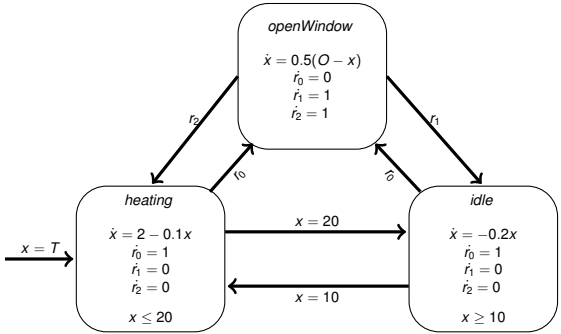
- Enhance Hybrid Systems with stochasticity



Stochastic Hybrid Automata

O = Outside Temperature
T = Room Temperature

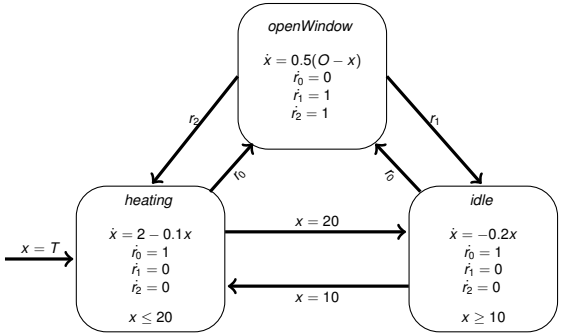
- Enhance Hybrid Systems with stochasticity
- Transition are composed of stochastic and non-stochastic ones



Stochastic Hybrid Automata

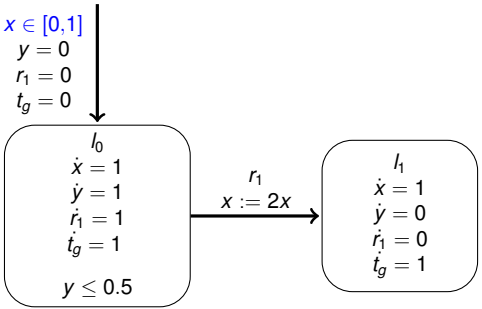
O = Outside Temperature
T = Room Temperature

- Enhance Hybrid Systems with stochasticity
- Transition are composed of stochastic and non-stochastic ones
- Stochastic transitions are accompanied by stochastic events and are syntactically modeled through random clocks



Initial-State Non-Determinism

- For LHA the initial set can be defined over a range of values
- The success of an automata run might depend on the point chosen from the initial set



Goal specification: $x \geq 2$
Time Bound:1

$x_{inital} = 0 \rightarrow x_{max} = 1.5$ ✗
$x_{inital} = 1 \rightarrow x_{max} = 3$ ✓

Solution

- Use existing forward analysis and extend with backwards refinement
- Computes the origin of successful state in the initial set
- Deduce probability that any point from the initial set can reach the specification

Forward Flowpipe Construction

- Use Forward Flowpipe construction to approximate forward reachability given a time bound

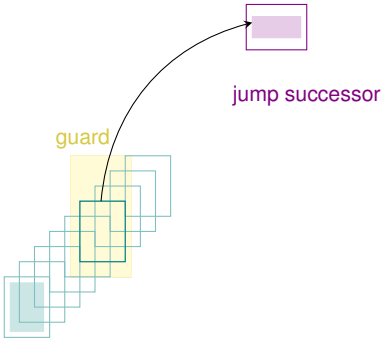
first Segment

Forward Flowpipe Construction

- Use Forward Flowpipe construction to approximate forward reachability given a time bound

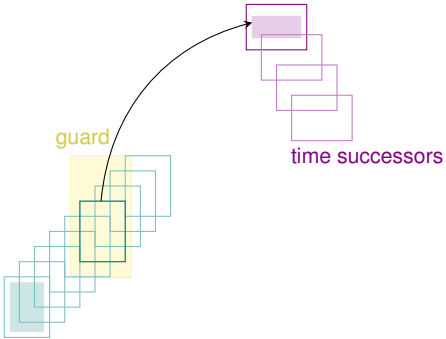
Forward Flowpipe Construction

- Use Forward Flowpipe construction to approximate forward reachability given a time bound



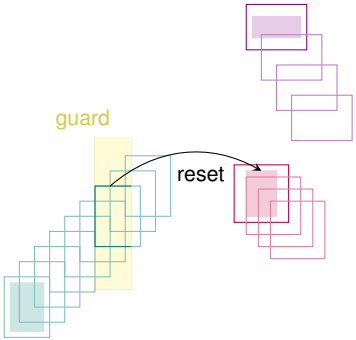
Forward Flowpipe Construction

- Use Forward Flowpipe construction to approximate forward reachability given a time bound



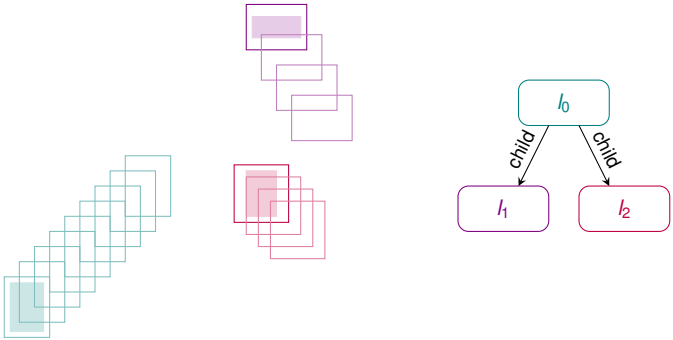
Forward Flowpipe Construction

- Use Forward Flowpipe construction to approximate forward reachability given a time bound



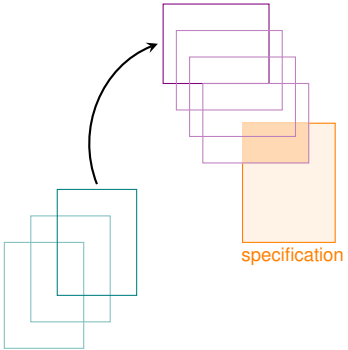
Forward Flowpipe Construction

- Use Forward Flowpipe construction to approximate forward reachability given a time bound



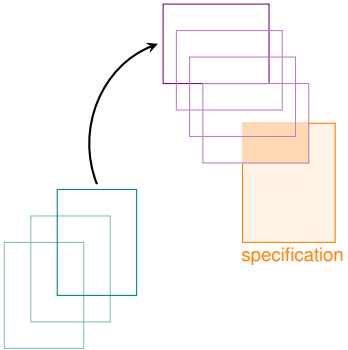
Forward Reachability

Take forward flowpipes and find intersection with specification



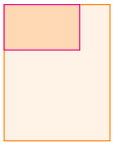
Backward Refinement - Concept

Input: Intersections of flowpipes and specification



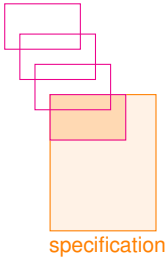
Backward Refinement - Concept

Input: Intersections of flowpipes and specification



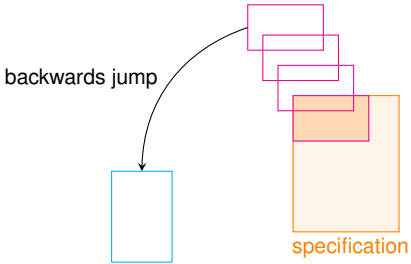
Backward Refinement - Concept

Input: Intersections of flowpipes and specification
Process: Compute time and jumps backwards for each segment



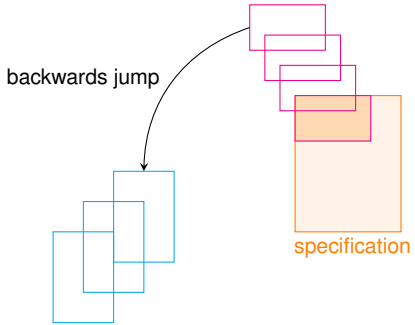
Backward Refinement - Concept

Input: Intersections of flowpipes and specification
Process: Compute time and jumps backwards for each segment
Goal: Subset of the initial set that lead to specification



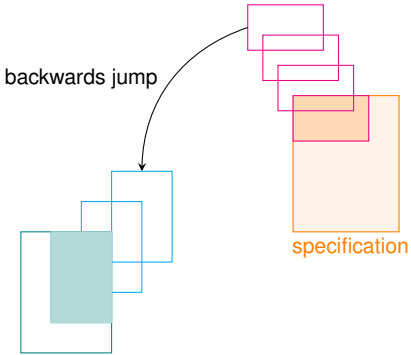
Backward Refinement - Concept

Input: Intersections of flowpipes and specification
Process: Compute time and jumps backwards for each segment
Goal: Subset of the initial set that lead to specification



Backward Refinement - Concept

Input: Intersections of flowpipes and specification
Process: Compute time and jumps backwards for each segment
Goal: Subset of the initial set that lead to specification



Backwards Refinement - Backwards Time Step

Time step is defined by the flow of the location
Flow is defined as ODE
$\dot{x} = Ax + b$
Can be reconstructed as
$\dot{y} = My$ with $y = \begin{bmatrix} x \\ 1 \end{bmatrix}, M = \begin{bmatrix} A & b \\ 0 & 0 \end{bmatrix}$

To compute the time predecessor, flow must be inverted.
Solve a time-continuous ODE involves forming the matrix exponential.

$$y(t) = e^{Mt} y(0)$$

for time step t.
The compute the time **pre**decessor, computation must be inverted

$$y(0) = e^{Mt^{-1}} y(t)$$

Backwards Jump

To compute the pre-jump segement, we exploit the HPolytope representation.

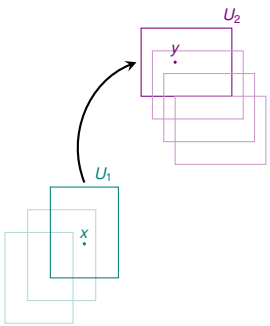
$$H = \{x \in \mathbb{R}^n | Ax \leq b\}, A \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^n$$

The forward pre- and post-jump segment can be defined as:

$$U_1 = \{x \in \mathbb{R}^n | K_1 x \leq c_1\}, K_1 \in \mathbb{R}^{n \times n}, c_1 \in \mathbb{R}^n$$
$$U_2 = \{y \in \mathbb{R}^n | K_2 y \leq c_2\}, K_2 \in \mathbb{R}^{n \times n}, c_2 \in \mathbb{R}^n$$

The jump reset is defined by $Ax + b$, therefore

$$\forall y \in U_2 \; \exists x \in U_1 : y = Ax + b$$



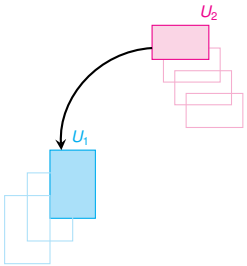
Backwards Jump

Using this information for the backwards refinement:

$$U_1 = \{x \in \mathbb{R}^n | K_1 x \leq c_1\}, K_1 \in \mathbb{R}^{n \times n}, c_1 \in \mathbb{R}^n$$
$$U_2 = \{y \in \mathbb{R}^n | K_2 y \leq c_2\}, K_2 \in \mathbb{R}^{n \times n}, c_2 \in \mathbb{R}^n$$

In the HPolytope definition of U_2, we can replace y with the reset function $Ax + b$:

$$K_2(Ax + b) \leq c_2$$
$$K_2 Ax \leq c_2 - K_2 b : \forall x \in U_1$$

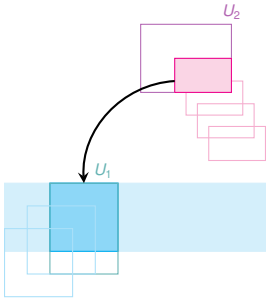


which provides alternative definition of U_1

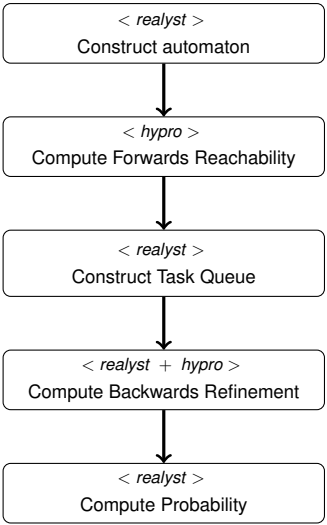
$$U_1 = \{x | K_2 Ax \leq c_2 - K_2 b\}$$

Backwards Jump - Infinite Bounds

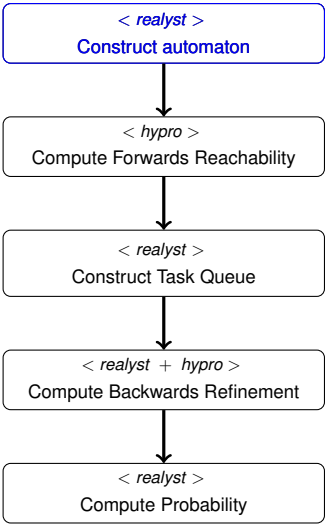
Problem: Some variables are not referenced in the reset
Example: $x := 3 \rightarrow$ No knowledge of pre-jump value of x
Solution: Intersect with pre-jump flowpipe



Implementation



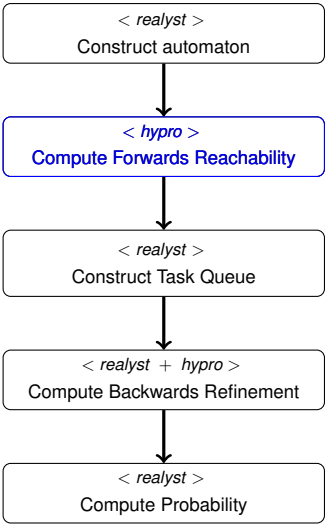
Implementation



User must define

- The continuous and stochastic variables
- The locations
- The flow in each location
- The invariants in each location
- The jumps between the locations
- The guard of each jump
- The reset of each jump
- The probability distribution over the stochastic variables
- **the probability distribution over the initial set**
- The possible initial values for every variable
- The goal specification

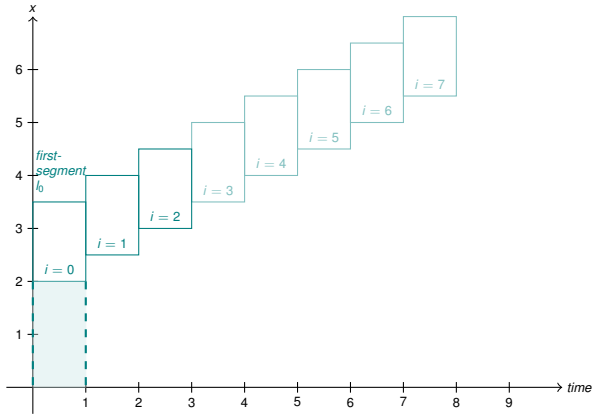
Implementation



- Uses Forward Flowpipe Construction
- Can be computed over- or underapproximative
- Returns a Reach Tree

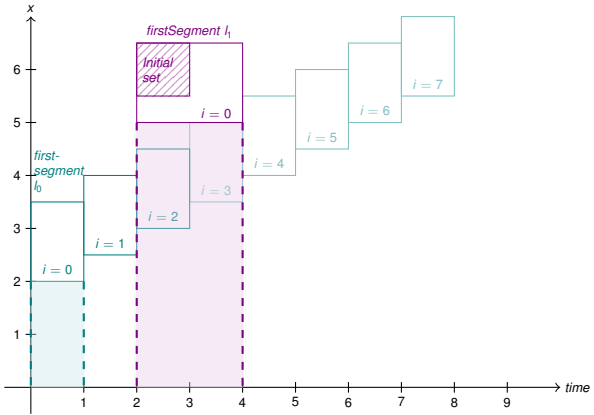
Implementation - Utilization of Timings

Reach Tree Nodes specify a `Timings` attribute
`Timings` define the global time covered by the initial set of the location



Implementation - Utilization of Timings

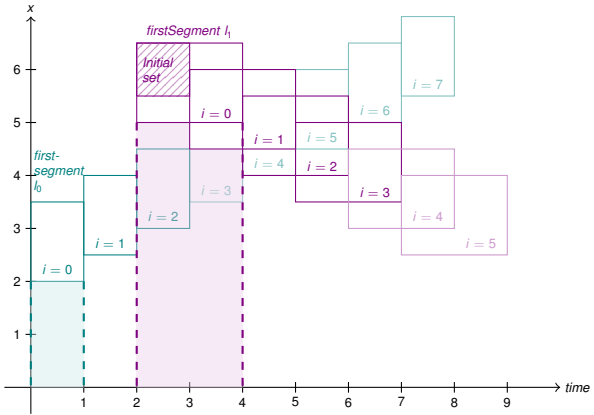
Reach Tree Nodes specify a `Timings` attribute
`Timings` define the global time covered by the initial set of the location



Implementation - Utilization of Timings

Reach Tree Nodes specify a `Timings` attribute

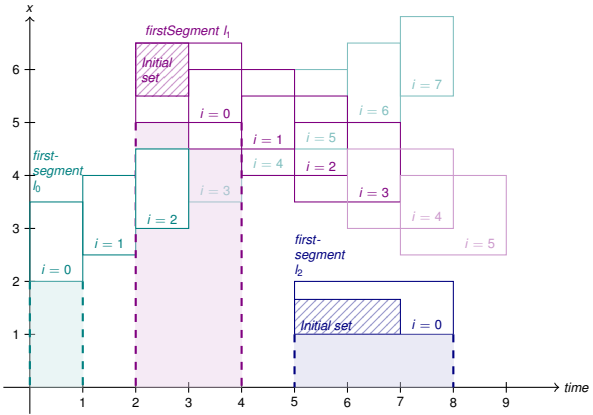
`Timings` define the global time covered by the initial set of the location



Implementation - Utilization of Timings

Reach Tree Nodes specify a `Timings` attribute

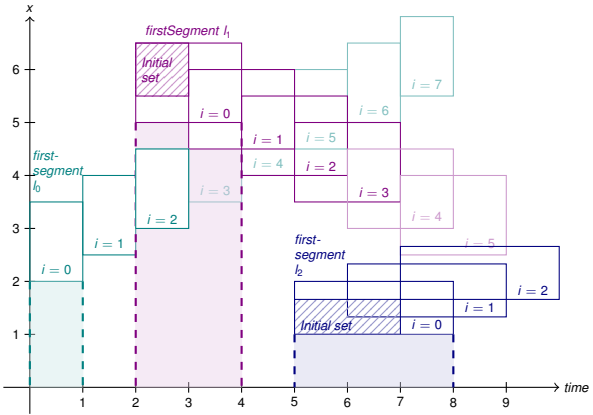
`Timings` define the global time covered by the initial set of the location



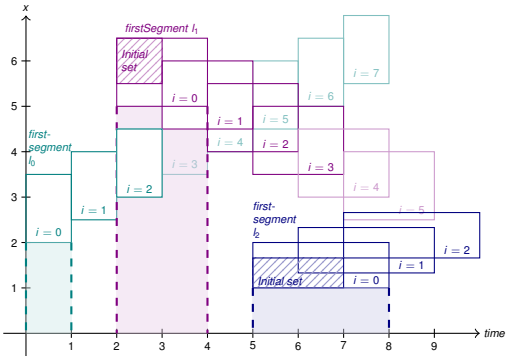
Implementation - Utilization of Timings

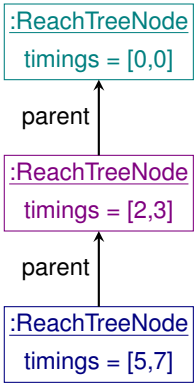
Reach Tree Nodes specify a `Timings` attribute

`Timings` define the global time covered by the initial set of the location

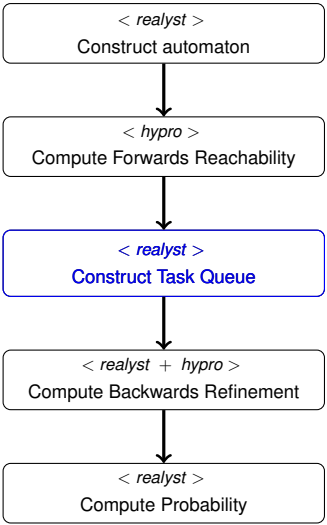


Implementation - Utilization of Timings



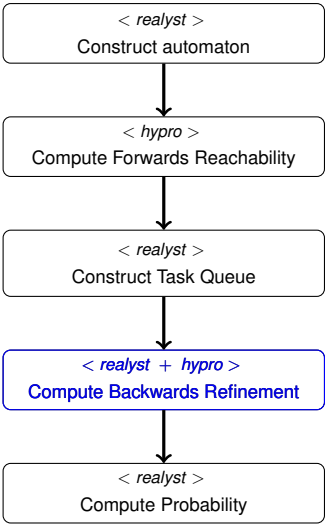


Implementation - Construct Task Queue



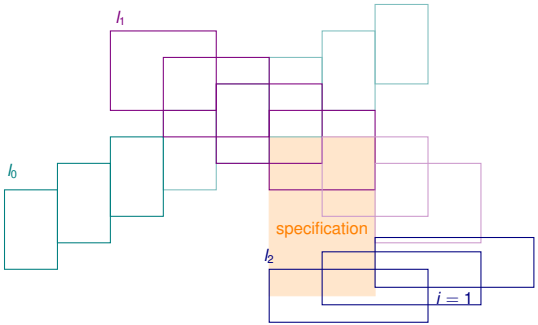
1. Check all flowpipes for intersection with specification
2. Create `TraceElement` for each intersection
3. Store all `TraceElements` of one ReachTreeNode's flowpipes in a `Trace`
4. Create a Task for each Trace

Implementation - Backwards Refinement



Iterate Task Queue to process each TraceElement in the Trace.

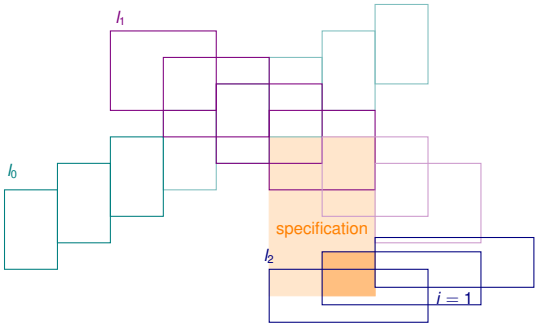
Implementation - Backwards Refinement



Process: Segment in l_3 with $i = 1$

1. Compute 1 Backwards Time Step

Implementation - Backwards Refinement



Process: Segment in l_3 with $i = 1$

1. Compute 1 Backwards Time Step

Implementation - Backwards Refinement

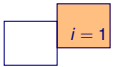
Process: Segment in l_3 with $i = 1$

1. Compute 1 Backwards Time Step

Implementation - Backwards Refinement

Process: Segment in l_3 with $i = 1$

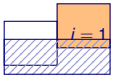
1. Compute 1 Backwards Time Step



Implementation - Backwards Refinement

Process: Segment in l_3 with $i = 1$

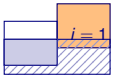
1. Compute 1 Backwards Time Step
2. Intersect with initial set in l_3



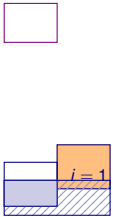
Implementation - Backwards Refinement

Process: Segment in l_3 with $i = 1$

1. Compute 1 Backwards Time Step
2. Intersect with initial set in l_3



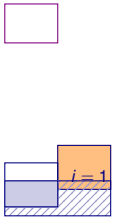
Implementation - Backwards Refinement

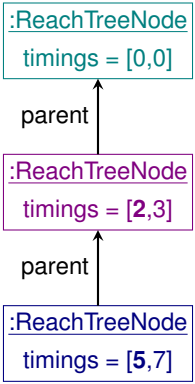


Process: Segment in l_3 with $i = 1$

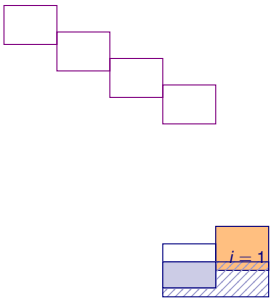
1. Compute 1 Backwards Time Step
2. Intersect with initial set in l_3
3. Perform backwards jump $\rightarrow$ Compute position in flowpipes vector from timings (3)

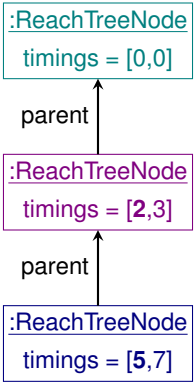
Implementation - Backwards Refinement



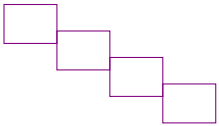


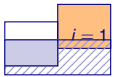
Implementation - Backwards Refinement





Implementation - Backwards Refinement

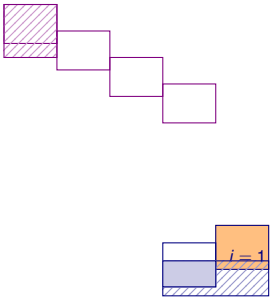




Process: Segment in l_3 with $i = 1$

1. Compute 1 Backwards Time Step
2. Intersect with initial set in l_3
3. Perform backwards jump $\rightarrow$ Compute position in flowpipes vector from timings (3)
4. Compute 3 Backwards Time Step

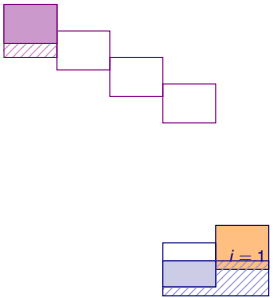
Implementation - Backwards Refinement



Process: Segment in l_3 with $i = 1$

1. Compute 1 Backwards Time Step
2. Intersect with initial set in l_3
3. Perform backwards jump $\rightarrow$ Compute position in flowpipes vector from timings (3)
4. Compute 3 Backwards Time Step
5. Intersect with initial set in l_2

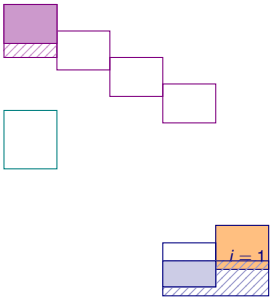
Implementation - Backwards Refinement



Process: Segment in l_3 with $i = 1$

1. Compute 1 Backwards Time Step
2. Intersect with initial set in l_3
3. Perform backwards jump $\rightarrow$ Compute position in flowpipes vector from timings (3)
4. Compute 3 Backwards Time Step
5. Intersect with initial set in l_2

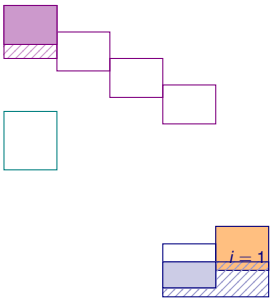
Implementation - Backwards Refinement

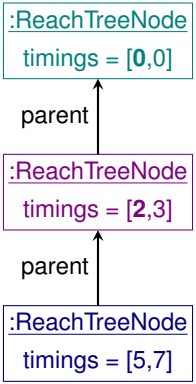


Process: Segment in l_3 with $i = 1$

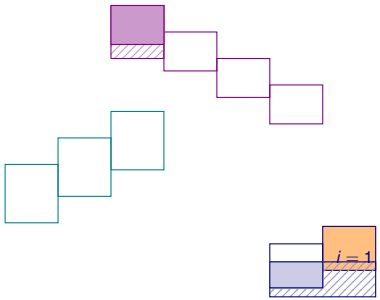
1. Compute 1 Backwards Time Step
2. Intersect with initial set in l_3
3. Perform backwards jump $\rightarrow$ Compute position in flowpipes vector from timings (3)
4. Compute 3 Backwards Time Step
5. Intersect with initial set in l_2
6. Perform backwards jump $\rightarrow$ Compute position in flowpipes vector from timings (2)

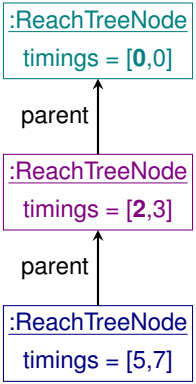
Implementation - Backwards Refinement



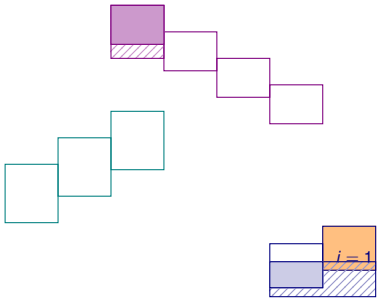


Implementation - Backwards Refinement





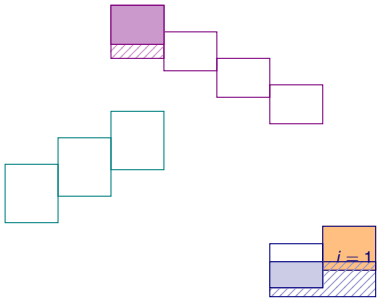
Implementation - Backwards Refinement



Process: Segment in l_3 with $i = 1$

1. Compute 1 Backwards Time Step
2. Intersect with initial set in l_3
3. Perform backwards jump $\rightarrow$ Compute position in flowpipes vector from timings (3)
4. Compute 3 Backwards Time Step
5. Intersect with initial set in l_2
6. Perform backwards jump $\rightarrow$ Compute position in flowpipes vector from timings (2)

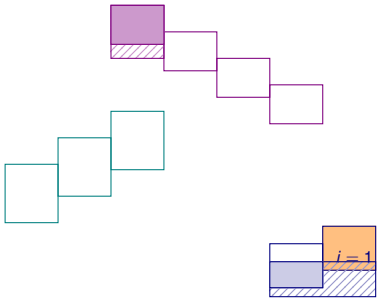
Implementation - Backwards Refinement



Process: Segment in l_3 with $i = 1$

1. Compute 1 Backwards Time Step
2. Intersect with initial set in l_3
3. Perform backwards jump $\rightarrow$ Compute position in flowpipes vector from timings (3)
4. Compute 3 Backwards Time Step
5. Intersect with initial set in l_2
6. Perform backwards jump $\rightarrow$ Compute position in flowpipes vector from timings (2)
7. Intersect with initial set in l_1

Implementation - Backwards Refinement



Process: Segment in l_3 with $i = 1$

1. Compute 1 Backwards Time Step
2. Intersect with initial set in l_3
3. Perform backwards jump $\rightarrow$ Compute position in flowpipes vector from timings (3)
4. Compute 3 Backwards Time Step
5. Intersect with initial set in l_2
6. Perform backwards jump $\rightarrow$ Compute position in flowpipes vector from timings (2)
7. Intersect with initial set in l_1
8. Cannot perform backwards jump $\rightarrow$ Done

Implementation

< *realyst* >
Construct automaton

↓

< *hypro* >
Compute Forwards Reachability

↓

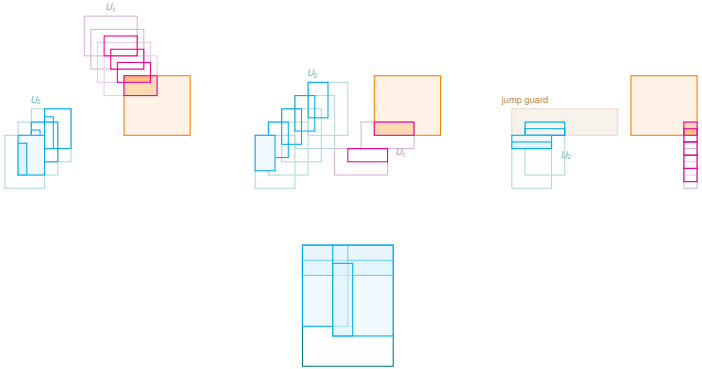
< *realyst* >
Construct Task Queue

↓

< *realyst* + *hypro* >
Compute Backwards Refinement

↓

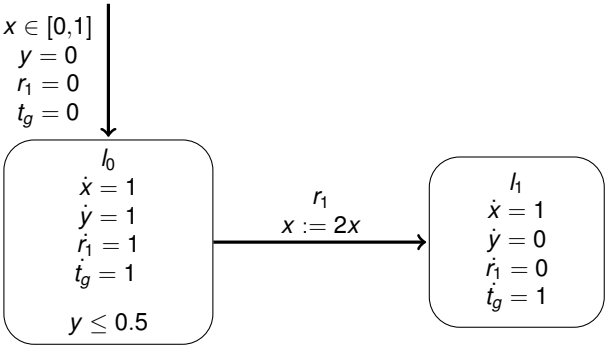
< *realyst* >
Compute Probability



⇒ Integrate using Monte Carlo Integration with defined Probability Distributions

⇒ Result: Initial-state Probability

Benchmark A



Goal Specification:

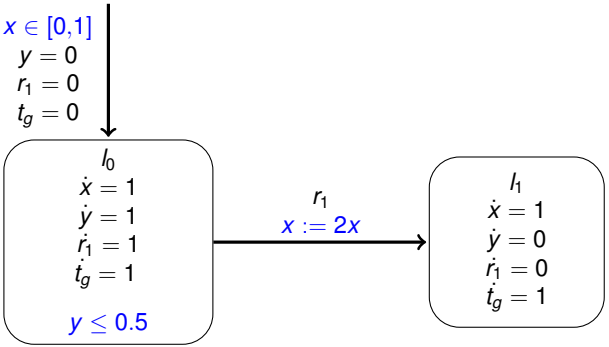
$x \geq 2$

$t_g \leq 1$

Time bound: 1

Uniform initial-state probability distributions for x over [0,1]

Benchmark A



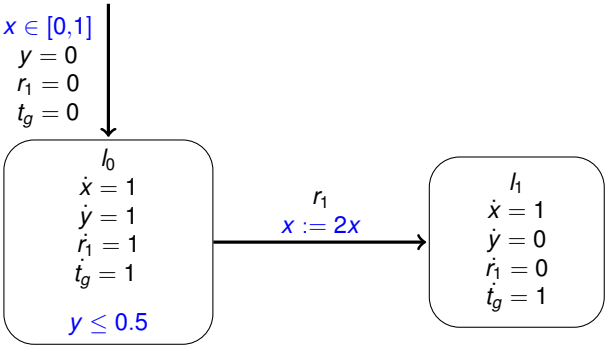
Goal Specification:

$$x \geq 2$$
$$t_g \leq 1$$

Time bound: 1

Uniform initial-state probability distributions for x over [0,1]

Benchmark A



Estimate:

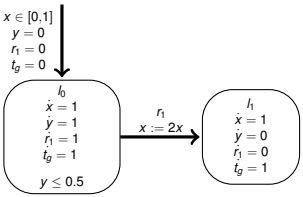
$$x_{end} = 2(x_{init} + t_o) + (T - t_0)$$

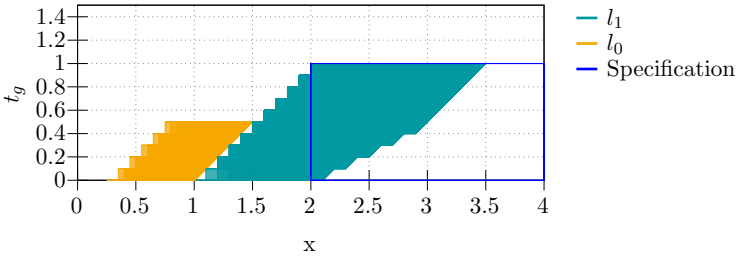
With $t_0 \leq 0.5$ due to invariant:

$$2 \leq 2(x_{min} + 0.5) + (1 - 0.5)$$
$$x_{min} \geq 0.25$$

$\Rightarrow$ Initial-state probability of 75%

Benchmark A





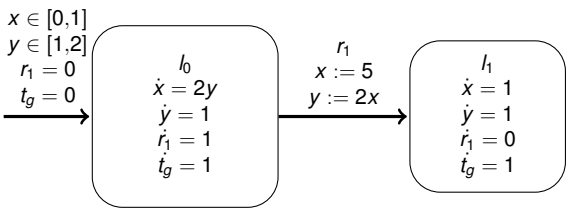
Result: 75%

Duration:

Forward reachability: 0.57s

Backwards refinement: 1.77s

Benchmark B



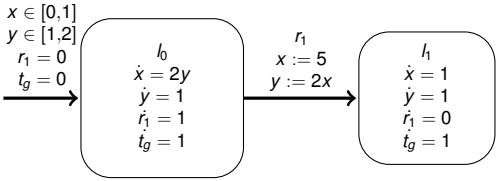
Goal Specification:

$$x \geq 5$$
$$y \geq 20$$
$$t_g \leq 2$$

Time bound: 2

Uniform initial-state probability distributions

Benchmark B



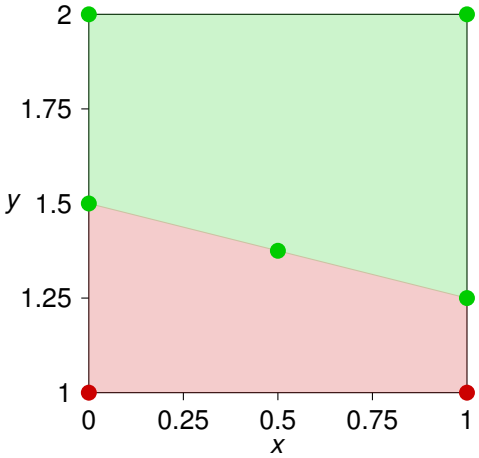
Estimate:

$$y_{end} = 2 \cdot (\int_0^{t_0} 2(y_{init} + t)dt + x_{init}) + (T - t_0)$$

Inserting values from specification:

$$2 \cdot {t_0}^2 + 4 \cdot y_{init} \cdot t_0 - t_0 + 2 \cdot x_{init} + 2 \geq 20$$

Benchmark B



Estimate:

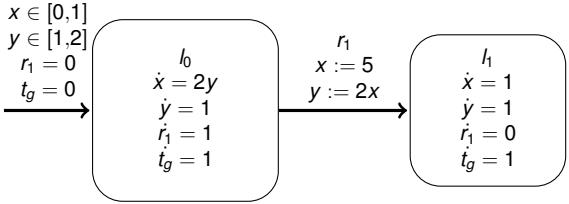
$$y_{end} = 2 \cdot (\int_0^{t_0} 2(y_{init} + t)dt + x_{init}) + (T - t_0)$$

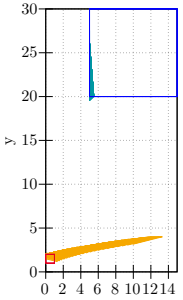
Inserting values from specification:

$$2 \cdot {t_0}^2 + 4 \cdot y_{init} \cdot t_0 - t_0 + 2 \cdot x_{init} + 2 \geq 20$$

$\Rightarrow$ Initial-state probability of 62.5%

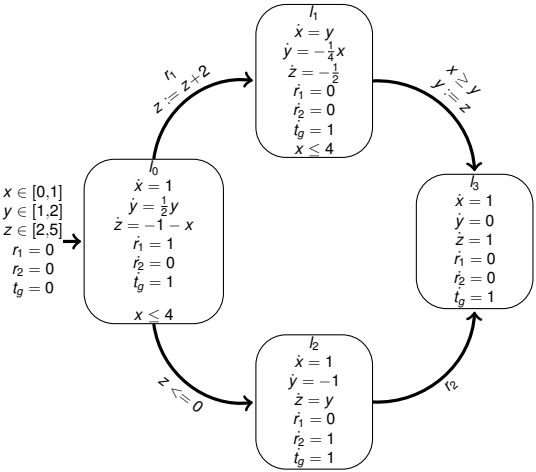
Benchmark B





Result: 62.47% (62.29%)
Duration:
Forward analysis: 3.49s
Backwards Refinement: 1.89s

Benchmark C



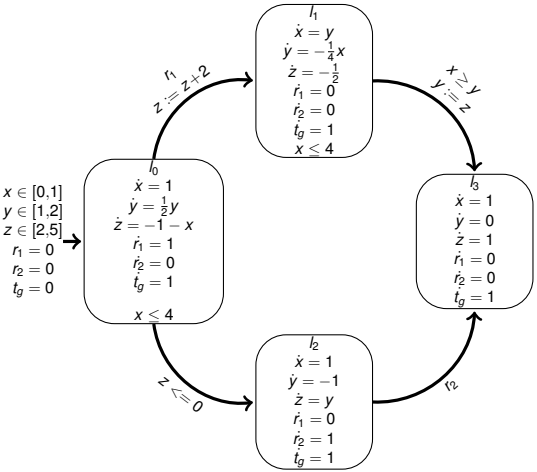
Goal Specification:

$$x \geq 4$$
$$y \leq 3$$
$$t_g \leq 2$$

Time bound: 2

Uniform initial-state probability distributions

Benchmark C



No estimate possible

Result: 17.04% (17.01%)
Duration:
  Forward Reachability: 5744.88s
  Backwards Refinement: 930.14s

Conclusion

- Builds backwards refinement on top of forwards reachability
- Successfully approximates initial-state probability to solve initial-state non-determinism probabilistically
- Point of Extension for Maximum Reachability Probability
- Runtime increases with automata complexity $\rightarrow$ Needs improvement (Multi-threading, aggregation,...)

Thank you!

Thank you for your attention!

Additional - Plots Benchmark C