Timed Push-Down Automata

Parosh Aziz Abdulla <parosh@it.uu.se>

Department of Information Technology
Uppsala University
Sweden

(Joint work with Mohamed Faouzi Atig and Jari Stenman)

1. Background
2. Push-Down Automata
3. Timed Push-Down Automata
4. Regions
5. Simulation
6. Zenoness
7. Priced Discrete-Timed Push-Down Automata
8. Conclusions and Future Work

Background

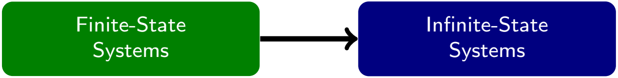
Finite-State Systems

Background

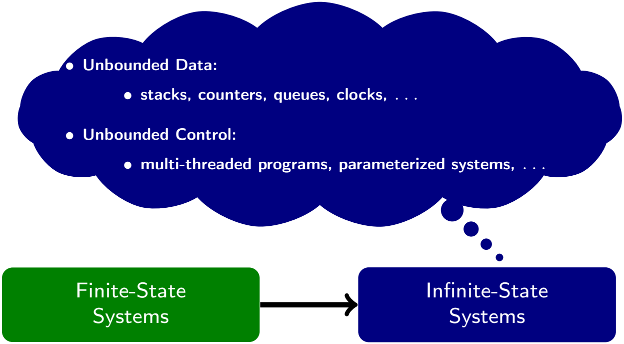
- **Classical Model Checking:**
  - **hardware verification**

Finite-State
Systems

Background

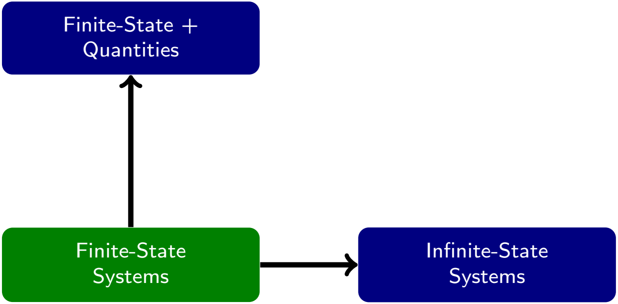


Background

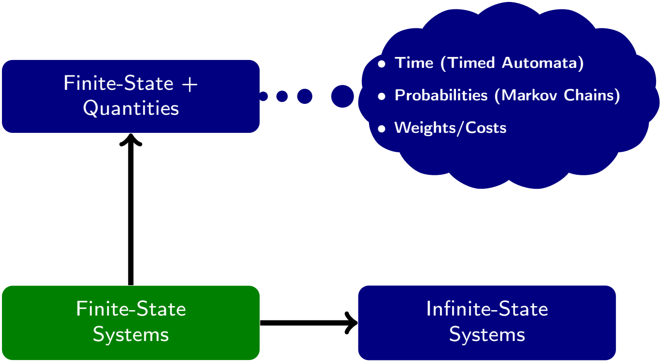


• Unbounded Data:
• stacks, counters, queues, clocks, . . .
• Unbounded Control:
• multi-threaded programs, parameterized systems, . . .
Finite-State Systems
Infinite-State Systems

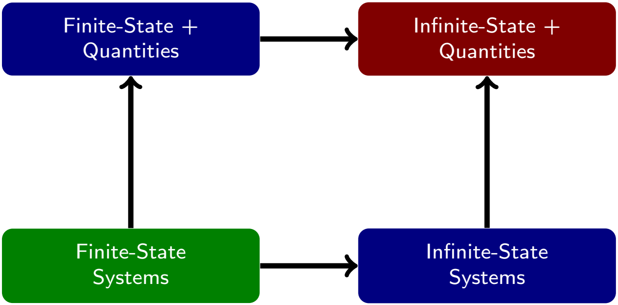
Background


Finite-State +
Quantities
Finite-State
Systems
Infinite-State
Systems

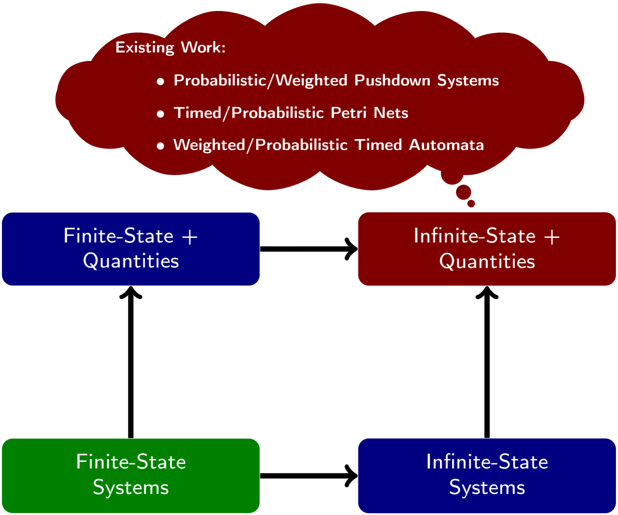
Background


Finite-State +
Quantities
Time (Timed Automata)
Probabilities (Markov Chains)
Weights/Costs
Finite-State
Systems
Infinite-State
Systems

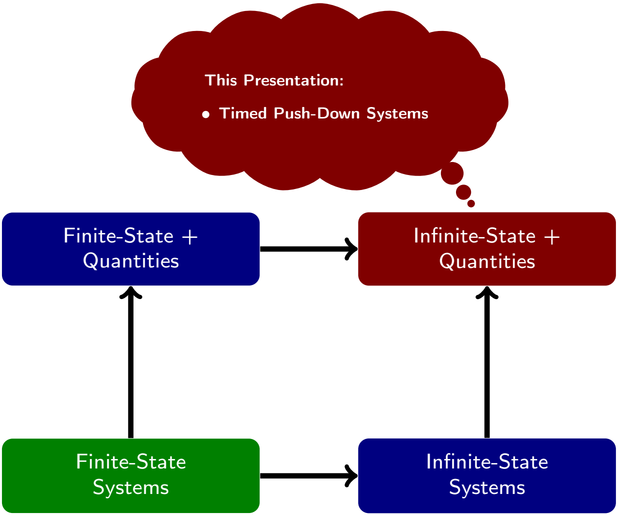
Background


Finite-State + Quantities
Infinite-State + Quantities
Finite-State Systems
Infinite-State Systems

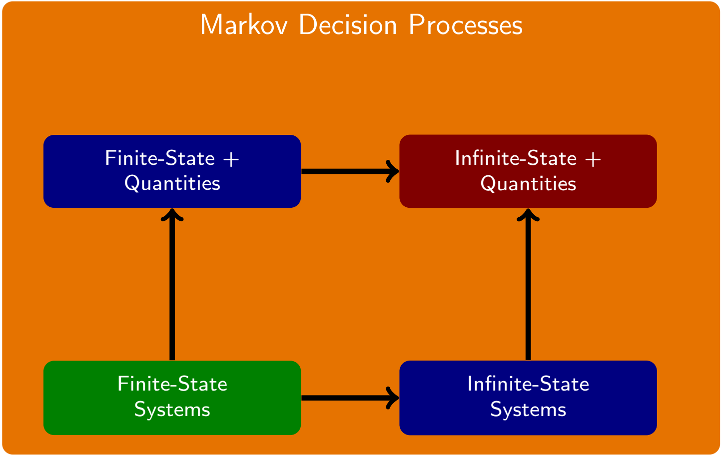
Background


Existing Work:
• Probabilistic/Weighted Pushdown Systems
• Timed/Probabilistic Petri Nets
• Weighted/Probabilistic Timed Automata
Finite-State + Quantities
Infinite-State + Quantities
Finite-State Systems
Infinite-State Systems

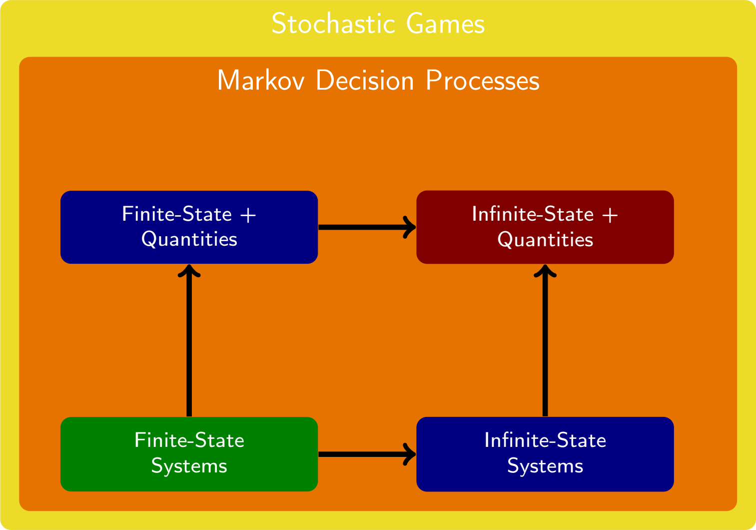
Background


This Presentation:
• Timed Push-Down Systems
Finite-State + Quantities
Infinite-State + Quantities
Finite-State Systems
Infinite-State Systems

Background

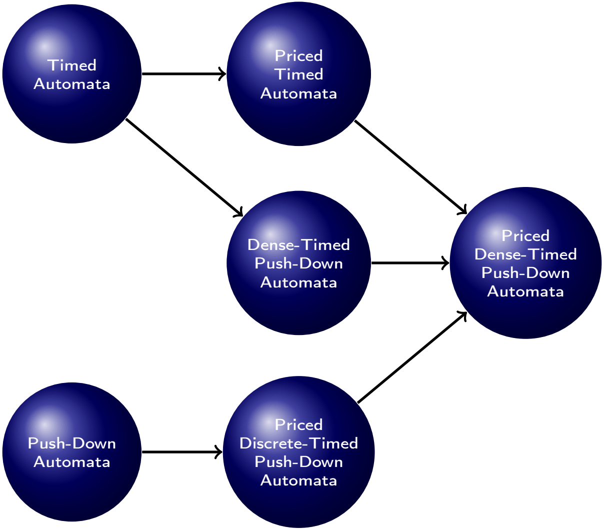

Markov Decision Processes
Finite-State + Quantities
Infinite-State + Quantities
Finite-State Systems
Infinite-State Systems

Background

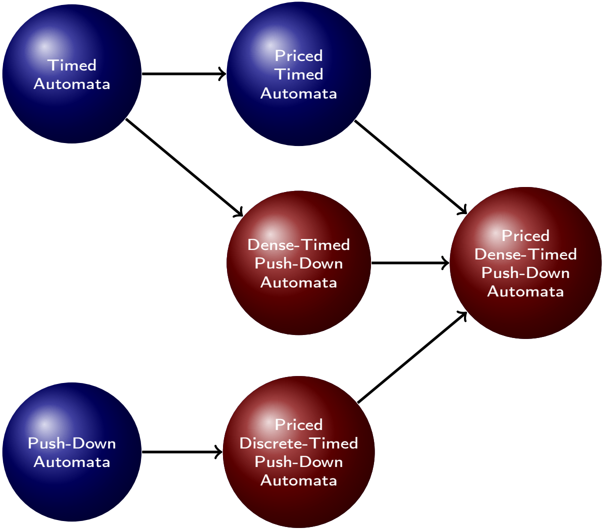


Stochastic Games
Markov Decision Processes
Finite-State + Quantities
Infinite-State + Quantities
Finite-State Systems
Infinite-State Systems

Related Models



Related Models



Pushdown Automata

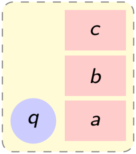
Definition

A *pushdown automaton* is a tuple (Q, Γ, Δ), where

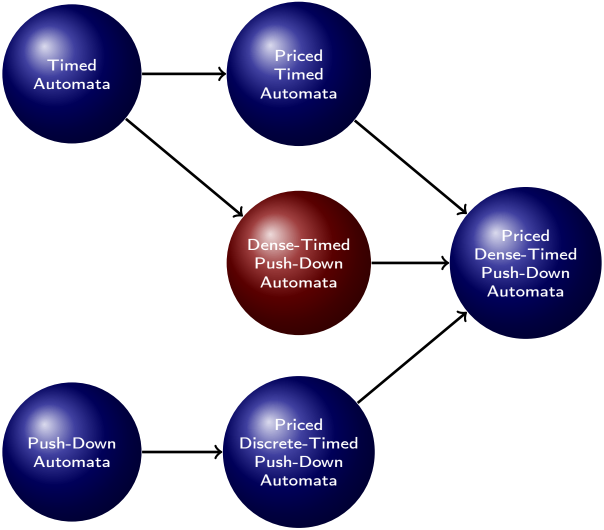
- Q finite set of states
- Γ finite stack alphabet
- Δ finite set of transition rules

Configurations consist of:

- A state $q \in Q$
- A word (stack content) w over Γ



Related Models



Pushdown Automata

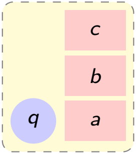
Definition

A *pushdown automaton* is a tuple (Q, Γ, Δ), where

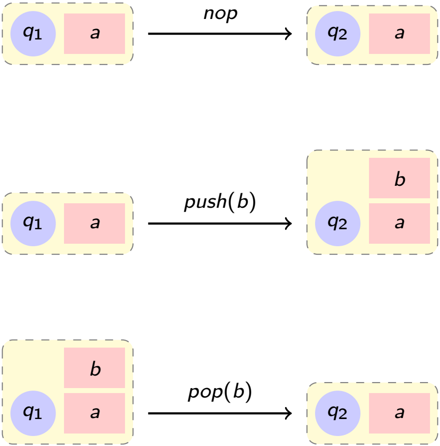
- Q finite set of states
- Γ finite stack alphabet
- Δ finite set of transition rules

Configurations consist of:

- A state $q \in Q$
- A word (stack content) w over Γ



Pushdown Automata



Extending Pushdown Automata

Model

- extend PDA by adding:
  - clocks
  - ages to the stack symbols
- make decisions based on:
  - clock values
  - ages of stack symbols

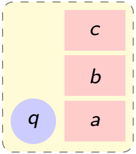
Challenge

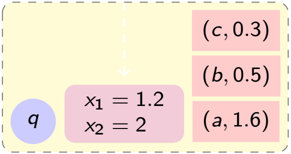
- One clock per stack symbol
- Unbounded number of clocks

TPDA Configurations

TPDA configuration =

- PDA configuration
- clock valuation $v : X \to \mathbb{R}^{\geq 0}$
- ages of stack symbols



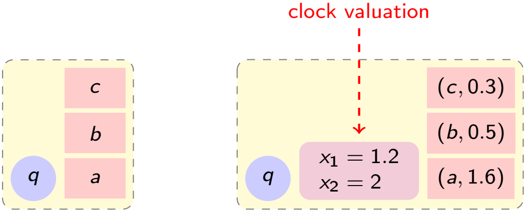


PDA Configuration **TPDA Configuration**

TPDA Configurations

TPDA configuration =

- PDA configuration
- clock valuation $v : X \to \mathbb{R}^{\geq 0}$
- ages of stack symbols

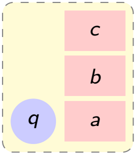


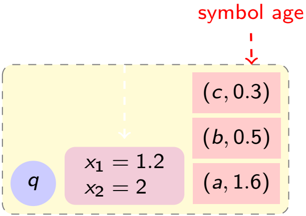
PDA Configuration **TPDA Configuration**

TPDA Configurations

TPDA configuration =

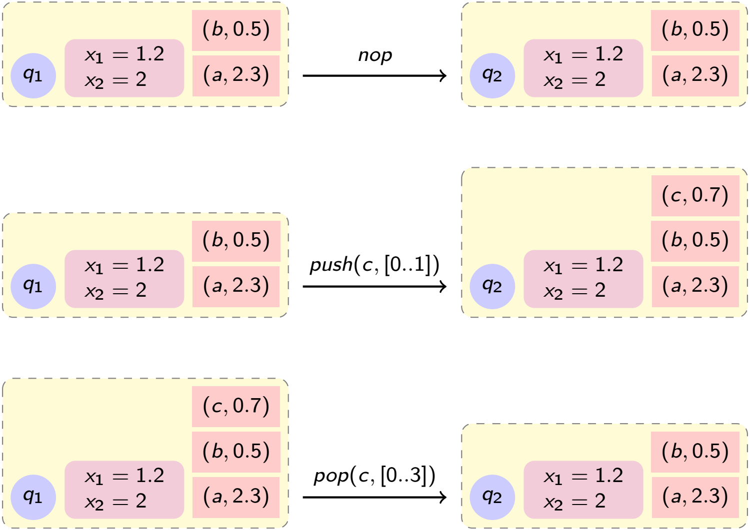
- PDA configuration
- clock valuation $v : X \to \mathbb{R}^{\geq 0}$
- ages of stack symbols





PDA Configuration **TPDA Configuration**

TPDA Transitions



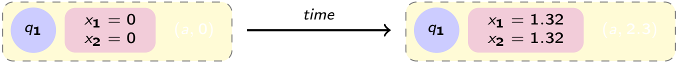
TPDA Transitions

$$q_1 \quad \begin{array}{l} x_1 = 1.2 \\ x_2 = 2 \end{array} \quad \begin{array}{c} (b, 0.5) \\ (a, 2.3) \end{array} \xrightarrow{x_1 \in [1..2]?} q_2 \quad \begin{array}{l} x_1 = 1.2 \\ x_2 = 2 \end{array} \quad \begin{array}{c} (b, 0.5) \\ (a, 2.3) \end{array}$$

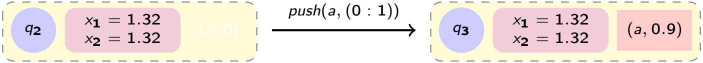
$$q_1 \quad \begin{array}{l} x_1 = 1.2 \\ x_2 = 2 \end{array} \quad \begin{array}{c} (b, 0.5) \\ (a, 2.3) \end{array} \xrightarrow{x_1 \leftarrow [0..0]} q_2 \quad \begin{array}{l} x_1 = 0 \\ x_2 = 2 \end{array} \quad \begin{array}{c} (b, 0.5) \\ (a, 2.3) \end{array}$$

$$q_1 \quad \begin{array}{l} x_1 = 0 \\ x_2 = 2 \end{array} \quad \begin{array}{c} (b, 0.5) \\ (a, 2.3) \end{array} \xrightarrow{\mathit{time}} q_1 \quad \begin{array}{l} x_1 = 0.4 \\ x_2 = 2.4 \end{array} \quad \begin{array}{c} (b, 0.9) \\ (a, 2.7) \end{array}$$

TPDA Computation



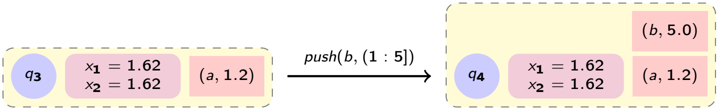
TPDA Computation



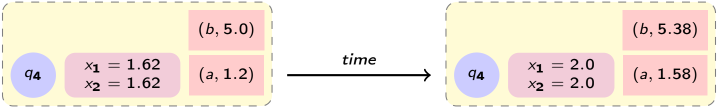
TPDA Computation

$$\left[\; q_3 \;\; \begin{array}{l} x_1 = 1.32 \\ x_2 = 1.32 \end{array} \;\; (a, 0.9) \;\right] \xrightarrow{\textit{time}} \left[\; q_3 \;\; \begin{array}{l} x_1 = 1.62 \\ x_2 = 1.62 \end{array} \;\; (a, 1.2) \;\right]$$

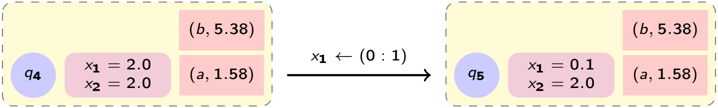
TPDA Computation



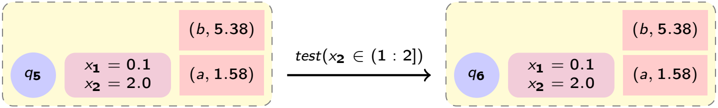
TPDA Computation



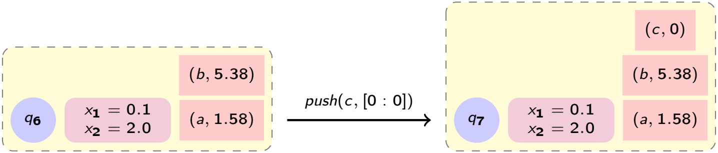
TPDA Computation



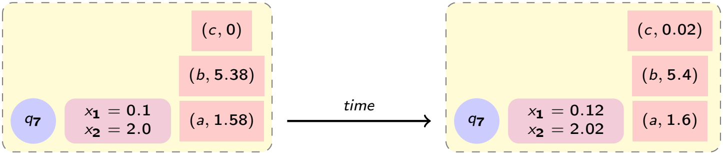
TPDA Computation



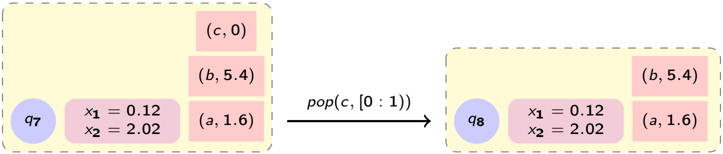
TPDA Computation



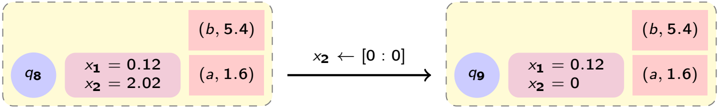
TPDA Computation



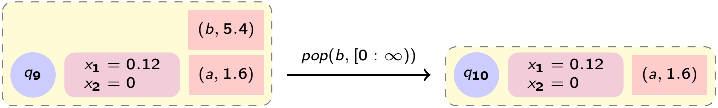
TPDA Computation



TPDA Computation



TPDA Computation



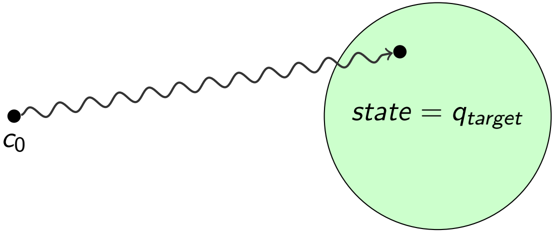
Reachability

Definition (The Reachability Problem)

Given:

- TPDA P
- Initial configuration c_0
- Target state q_{target}

Is there a computation from c_0 to some configuration in which $state = q_{target}$



Main Result

Reachability Problem for TPDA is decidable

- Reduction from reachability problem for TPDA to reachability problem for PDA
- Simulate TPDA with PDA

Challenge:

- All components need to be finite, despite
  - Continuous time
  - Unboundedly many clocks

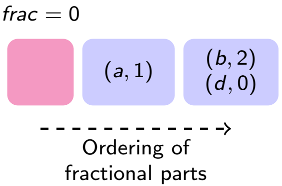
Method:

- Symbolic representation: Regions
- Simulation of TPDA by PDA

Regions

Definition

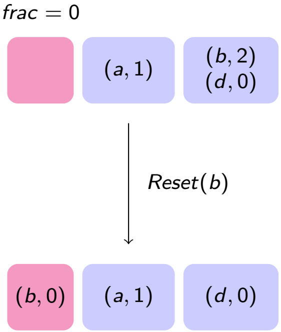
A region is a word over $2^{\Gamma \times [0..max]}$.



- $[\![\text{Region}]\!]$ = clock valuations satisfying it
- Finitely many regions

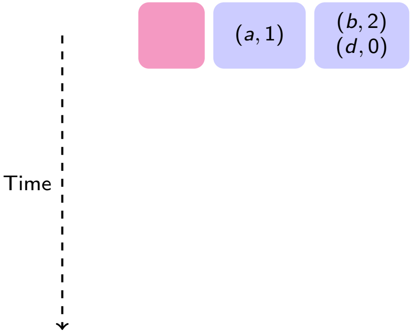
Regions

Resetting Clock Values:



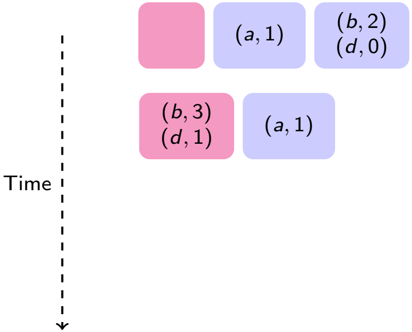
Regions

Passage of time: Rotation



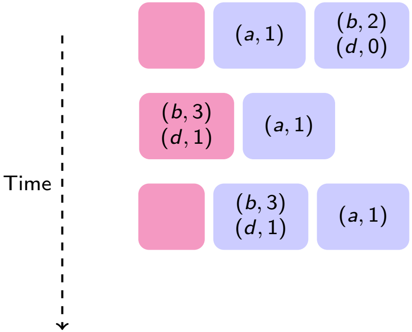
Regions

Passage of time: Rotation



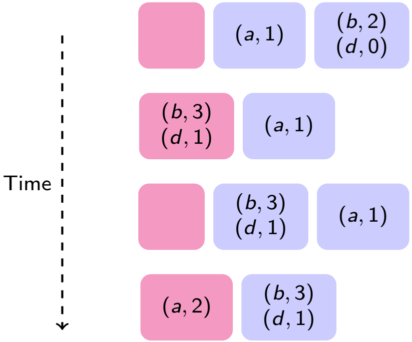
Regions

Passage of time: Rotation



Regions

Passage of time: Rotation



Simulating a TPDA

Main Ideas:

- extend regions to TPDA
- store regions in the stack
- relate each stack symbol to global clocks

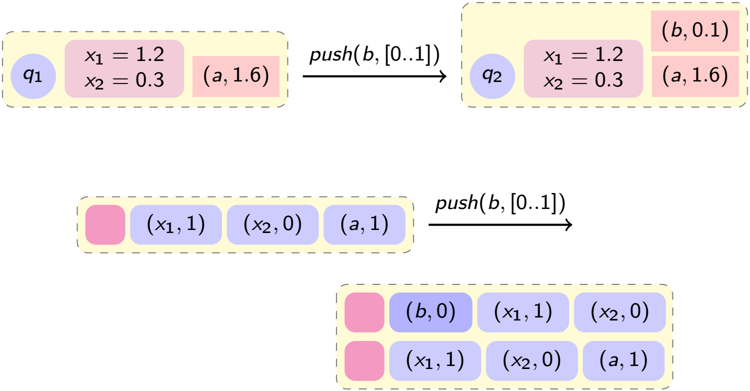
Simulating a TPDA

q | $x_1 = 1.2$, $x_2 = 0.3$ | $(a, 1.6)$

q | | $(x_1, 1)$ | $(x_2, 0)$ | $(a, 1)$

TPDA Configuration **PDA Configuration**

Simulating Push

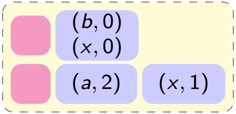


Simulating Pop

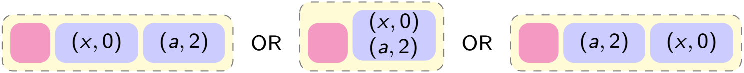
Five operations are simple:

- Nop and Test do not modify anything
- Push creates new topmost region
- Reset and Time modify topmost region

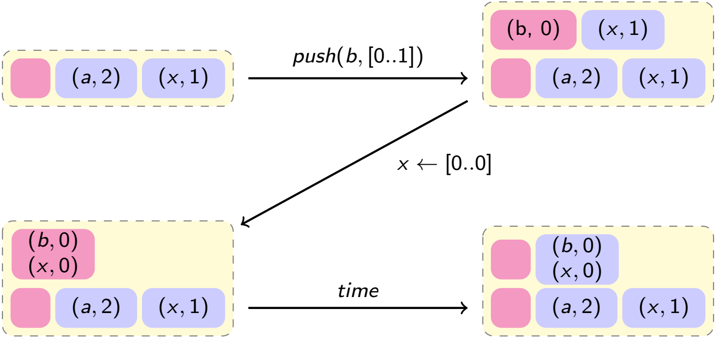
The difficult operation is pop:



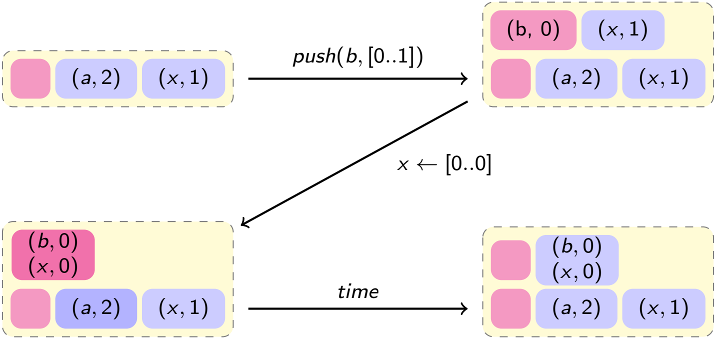
Which new topmost region when popping b?



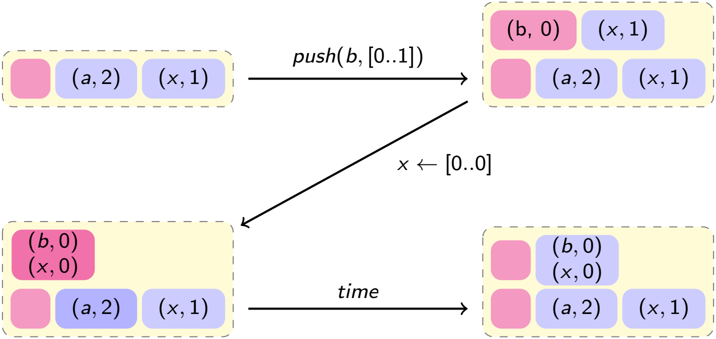
Simulating Pop



Simulating Pop

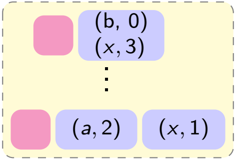


Simulating Pop



Lost information: $\mathit{frac}(x) \neq \mathit{frac}(a)$

Simulating a TPDA



- In general: relate clocks in topmost region with symbols that lie arbitrarily deep in the stack.
- Can we do this in a finite way?

Shadow Items

A little bit of information about previous region.

Regions contain

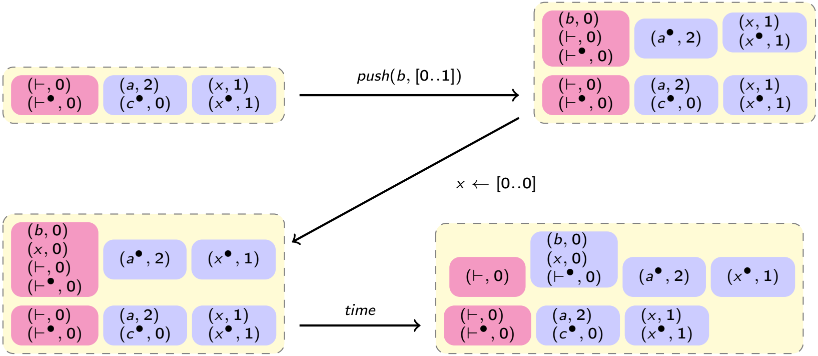
- Plain items: $X \cup \Gamma \cup \{\vdash\}$
- Shadow items: $X^\bullet \cup \Gamma^\bullet \cup \{\vdash^\bullet\}$

$(\vdash, 0)$ $(\vdash^\bullet, 0)$	$(b, 1)$	$(a^\bullet, 2)$	$(x, 1)$ $(x^\bullet, 1)$
$(\vdash, 0)$ $(\vdash^\bullet, 0)$	$(a, 2)$ $(c^\bullet, 0)$	$(x, 1)$ $(x^\bullet, 1)$	

$\vdots$

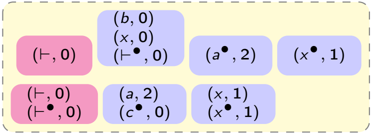
$\vdash$ is a reference clock which is (almost) always 0

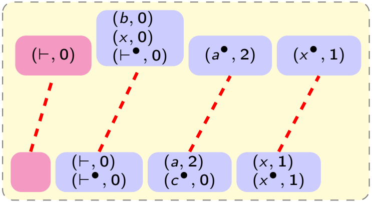
Simulating Pop with Shadow Items



Simulating Pop with Shadow Items

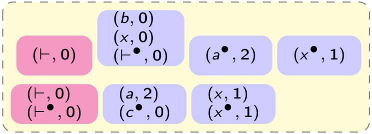
- Rotate lower region until matching
-
-
-

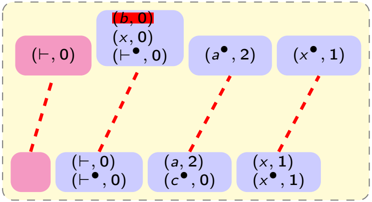




Simulating Pop with Shadow Items

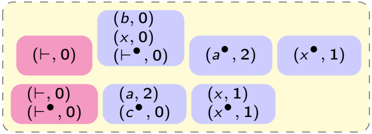
- Rotate lower region until matching
- Plain stack symbol taken from lower region

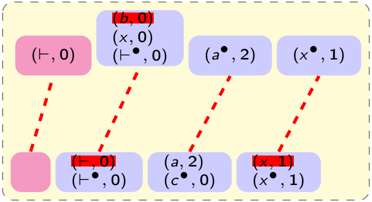




Simulating Pop with Shadow Items

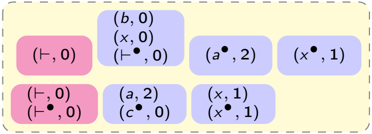
- Rotate lower region until matching
- Plain stack symbol taken from lower region
- Plain clock symbols taken from upper region
-

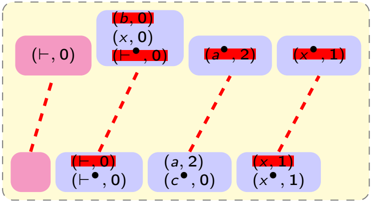




Simulating Pop with Shadow Items

- Rotate lower region until matching
- Plain stack symbol taken from lower region
- Plain clock symbols taken from upper region
- Shadow items taken from lower region

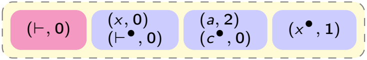


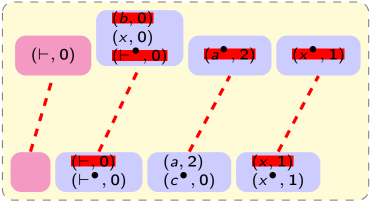


Simulating Pop with Shadow Items

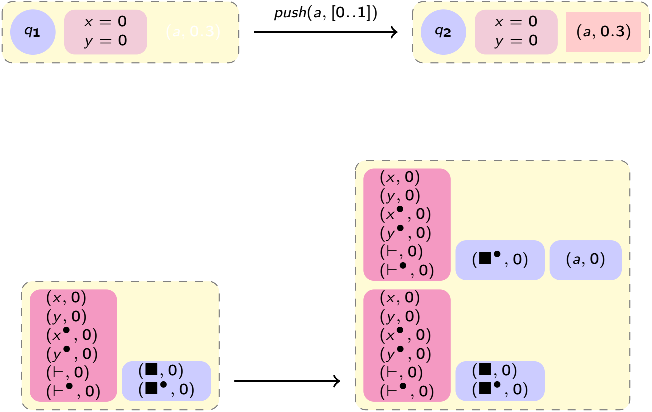
- Rotate lower region until matching
- Plain stack symbol taken from lower region
- Plain clock symbols taken from upper region
- Shadow items taken from lower region

Merge:

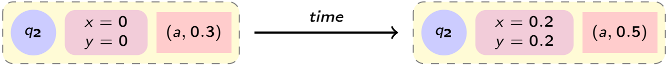


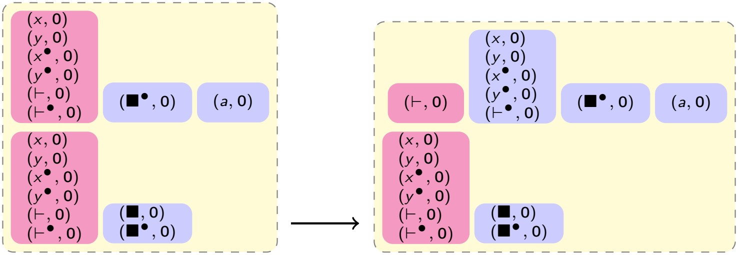


Example Simulation

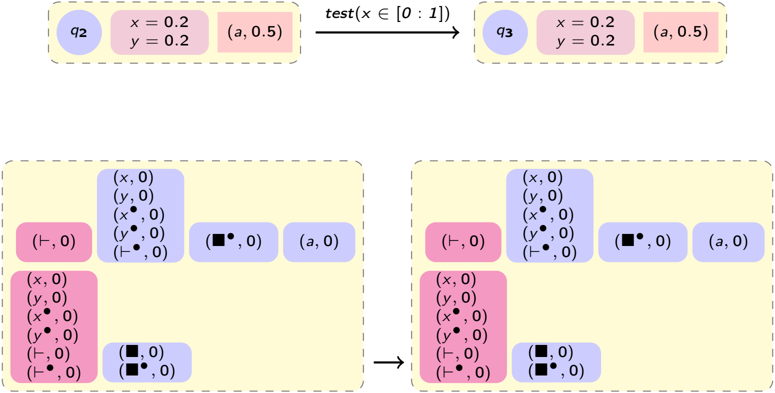


Example Simulation

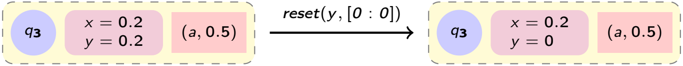


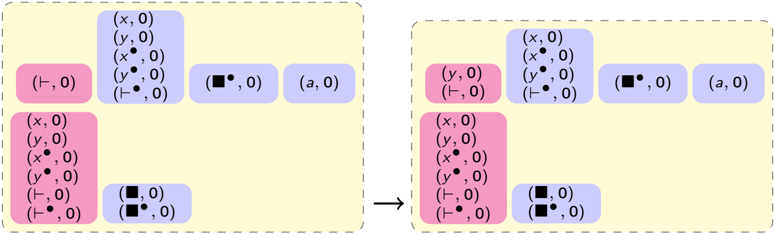


Example Simulation

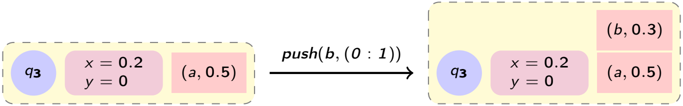


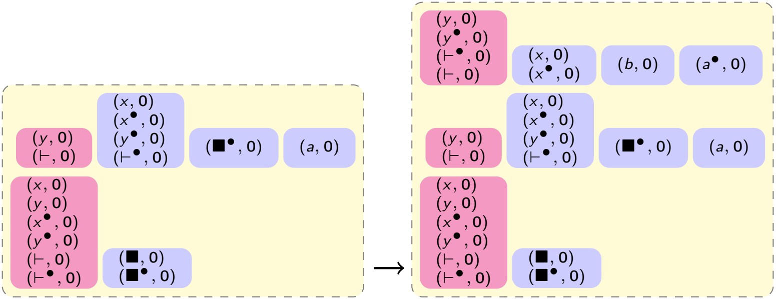
Example Simulation



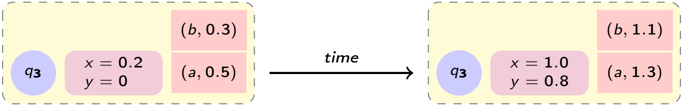


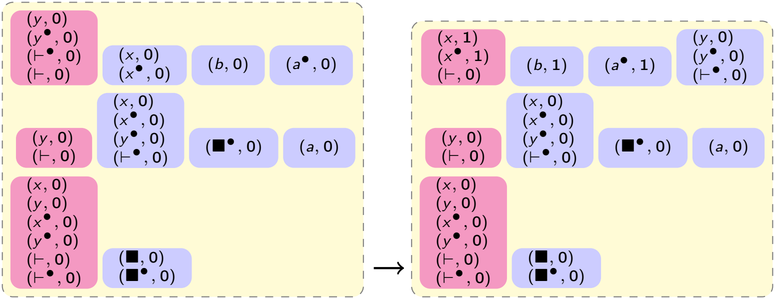
Example Simulation



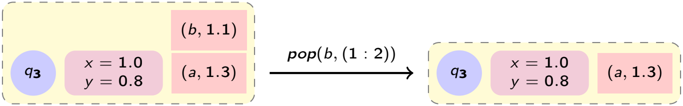


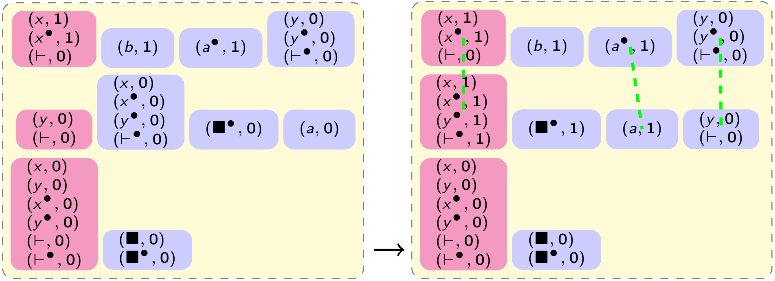
Example Simulation



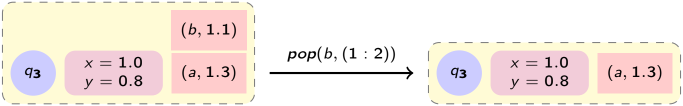


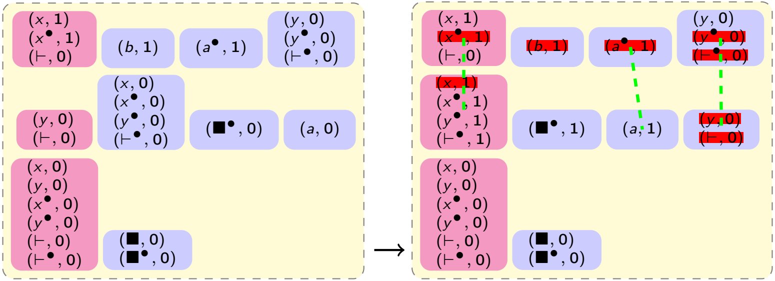
Example Simulation



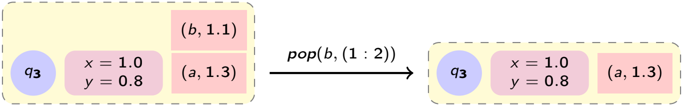


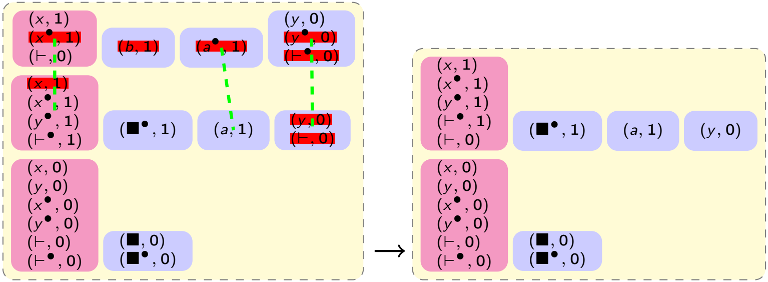
Example Simulation



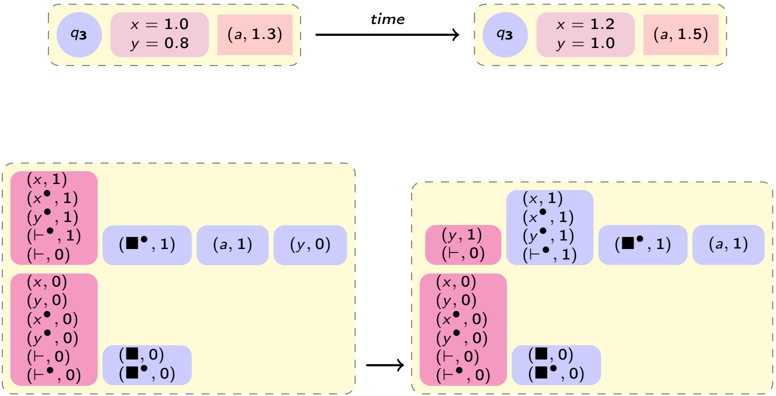


Example Simulation

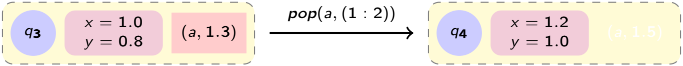


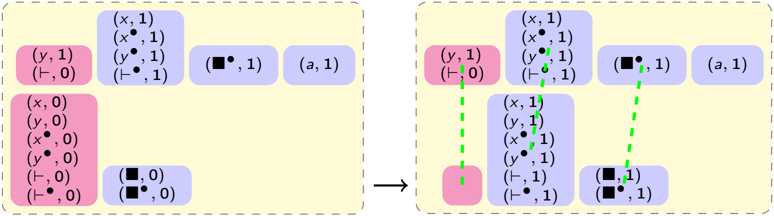


Example Simulation

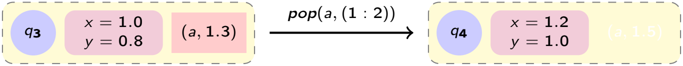


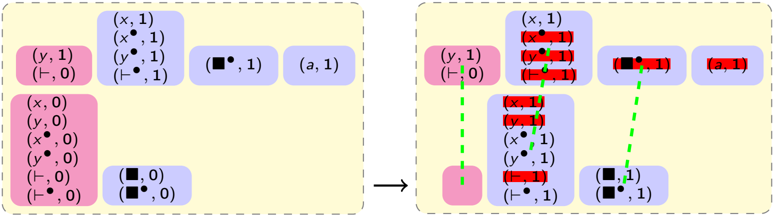
Example Simulation



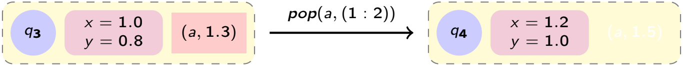


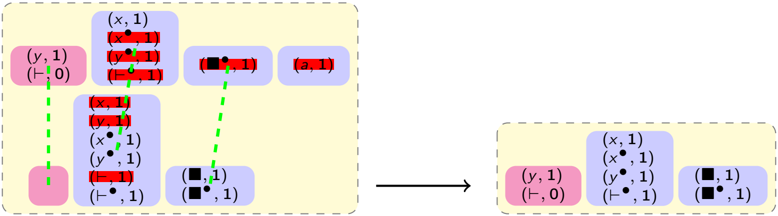
Example Simulation





Example Simulation





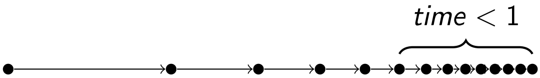
Zenoness

Definition

A computation is *zeno* if it has infinitely many discrete transitions in finite time.

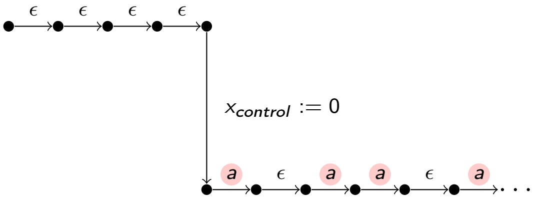
Detecting Zenoness in TPDA

TPDA



Mode 1: No restrictions

PDA

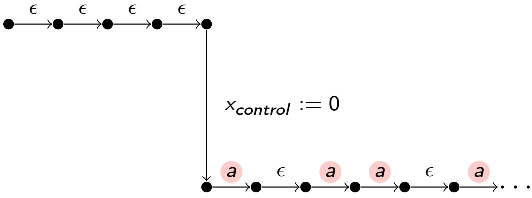


Mode 2: $x_{control} < 1$

Detecting Zenoness in TPDA

Mode 1: No restrictions

PDA

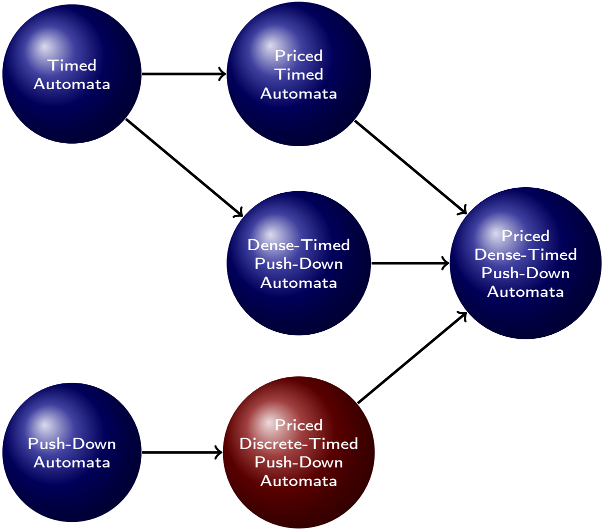


Mode 2: $x_{control} < 1$

Theorem

$Zenoness \iff a^{\omega} \in Traces(PDA)$

Related Models



Conclusions and Future Work

Conclusions

- Timed and cost extensions of push-down automata
- Reachability problem for TPDA is EXPTIME-complete

Future Work

- Priced Dense-Timed PDA