Robust Weighted Timed Automata and Games

Ocan Sankur

Université Libre de Bruxelles

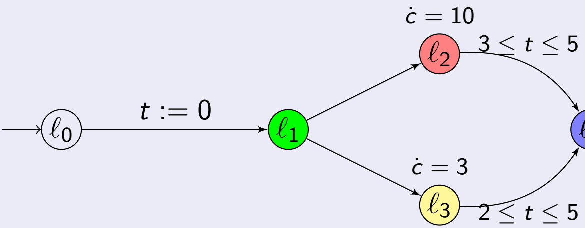
Joint with **Patricia Bouyer, Nicolas Markey**
ENS Cachan & CNRS

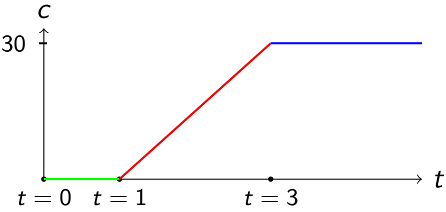
Weighted Timed Automata

[Alur, La Torre, Pappas 2001]

[Behrmann, Fehnker, Hune, Larsen, Petterson, Romijn, Vaandrager 2001]

A weighted timed automaton



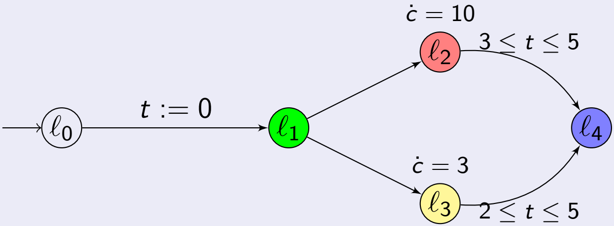


Weighted Timed Automata

[Alur, La Torre, Pappas 2001]

[Behrmann, Fehnker, Hune, Larsen, Petterson, Romijn, Vaandrager 2001]

A weighted timed automaton



Optimal-cost reachability is PSPACE-complete.

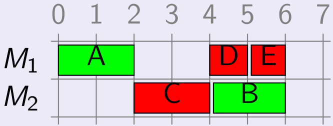
Motivation for Robustness: Scheduling

Scheduling analysis with timed automata [Abdeddaim, Asarin, Maler 2006]

Goal: analyse a *work-conserving* scheduling policy on given scenarios.

(*work-conserving:* no machine is idle if a task is waiting for execution)

Scenario



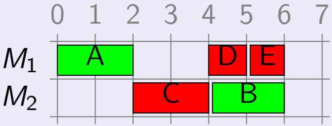
with the constraints: $A \to B, \quad C \to D, E.$

1. A, D, E must be scheduled on machine M_1,
2. B, C must be scheduled on machine M_2,
3. C starts no sooner than 2 time units,

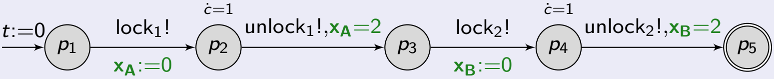
Motivation for Robustness: Scheduling

Scheduling analysis with timed automata [Abdeddaim, Asarin, Maler 2006]
Goal: analyse a *work-conserving* scheduling policy on given scenarios.
(*work-conserving:* no machine is idle if a task is waiting for execution)

Scenario



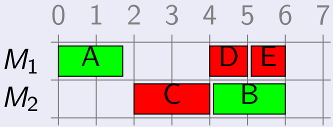
Timed automaton: model $A \rightarrow B$ as:



▶ Timing analysis: minimal cost reachability gives the fastest scheduling policy. Schedulable in 6 time units.

Motivation for Robustness: Scheduling

↯ Unexpectedly ↯: duration of A is now 1.999.

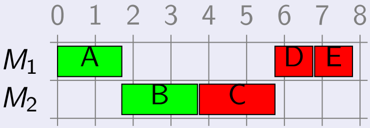


This cannot be an outcome of an algorithm (not work-conserving).

Best work-conserving scheduler is ...

Motivation for Robustness: Scheduling

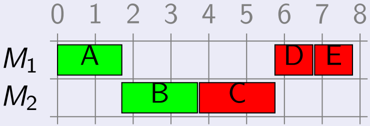
↯ Unexpectedly ↯: duration of A is now 1.999.



Best work-conserving scheduler is ... which completes in 7.999 time units.
Previous analysis did not capture this **timing anomaly.**

Motivation for Robustness: Scheduling

↯ Unexpectedly ↯: duration of A is now 1.999.

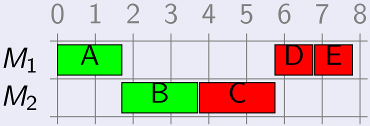


Best work-conserving scheduler is ... which completes in 7.999 time units. Previous analysis did not capture this **timing anomaly.**

Conclusion: Need for robustness of models against *longer* and *shorter* execution times than expected.

Motivation for Robustness: Scheduling

↯ Unexpectedly ↯: duration of A is now 1.999.



Best work-conserving scheduler is ... which completes in 7.999 time units.
Previous analysis did not capture this **timing anomaly.**

Conclusion: Need for robustness of models against *longer* and *shorter* execution times than expected.

Goal 1

Study robust optimal-cost reachability in weighted timed automata.

Motivation for Robustness: Undecidability in Games

Weighted timed **games**

Two-player zero-sum games played on a weighted timed automaton.
First player's objective: reachability with optimal cost

Theorem [Brihaye, Bruyère, Raskin 2005] [Bouyer, Brihaye, Markey 2006]

Optimal-cost reachability in weighted timed games is undecidable.

Note that WTG $\leftrightarrow$ control problems with costs

Motivation for Robustness: Undecidability in Games

Weighted timed games

Two-player zero-sum games played on a weighted timed automaton.
First player's objective: reachability with optimal cost

Theorem [Brihaye, Bruyère, Raskin 2005] [Bouyer, Brihaye, Markey 2006]

Optimal-cost reachability in weighted timed games is undecidable.

Note that WTG $\leftrightarrow$ control problems with costs

Goal 2

Introduce fuzziness in the semantics to recover decidablity for weighted timed games.

Next: Two variants of a robust semantics + results

1. **Excess** Robust Game Semantics

Let $\mathcal{A}$ be a **weighted timed automaton** and $\delta > 0$.

Semantics $\mathcal{G}_\delta(\mathcal{A})$

At any state (ℓ, ν, c),

1. **Excess** Robust Game Semantics

Let $\mathcal{A}$ be a **weighted timed automaton** and $\delta > 0$.

Semantics $\mathcal{G}_\delta(\mathcal{A})$

At any state (ℓ, ν, c),

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d \models g$,

1. **Excess** Robust Game Semantics

Let $\mathcal{A}$ be a **weighted timed automaton** and $\delta > 0$.

Semantics $\mathcal{G}_\delta(\mathcal{A})$

At any state (ℓ, ν, c),

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d \models g$,
2. Environment chooses $d' \in [d - \delta, d + \delta]$,

1. **Excess** Robust Game Semantics

Let $\mathcal{A}$ be a **weighted timed automaton** and $\delta > 0$.

Semantics $\mathcal{G}_\delta(\mathcal{A})$

At any state (ℓ, ν, c),

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d \models g$,
2. Environment chooses $d' \in [d - \delta, d + \delta]$,
3. New state is $(\ell', (\nu + d')[R \leftarrow 0], c + \text{Slope}(\ell) \cdot d')$

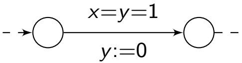
1. **Excess** Robust Game Semantics

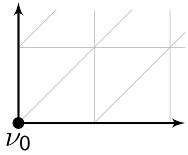
Let $\mathcal{A}$ be a **weighted timed automaton** and $\delta > 0$.

Semantics $\mathcal{G}_\delta(\mathcal{A})$

At any state (ℓ, ν, c),

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d \models g$,
2. Environment chooses $d' \in [d - \delta, d + \delta]$,
3. New state is $(\ell', (\nu + d')[R \leftarrow 0], c + \mathrm{Slope}(\ell) \cdot d')$





Guards may not be satisfied **after** perturbation.

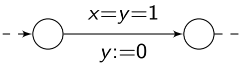
1. **Excess** Robust Game Semantics

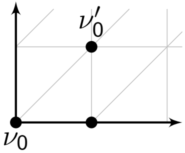
Let $\mathcal{A}$ be a **weighted timed automaton** and $\delta > 0$.

Semantics $\mathcal{G}_\delta(\mathcal{A})$

At any state (ℓ, ν, c),

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d \models g$,
2. Environment chooses $d' \in [d - \delta, d + \delta]$,
3. New state is $(\ell', (\nu + d')[R \leftarrow 0], c + \text{Slope}(\ell) \cdot d')$





Guards may not be satisfied **after** perturbation.

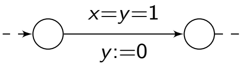
1. **Excess** Robust Game Semantics

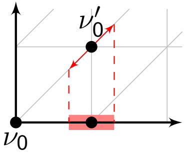
Let $\mathcal{A}$ be a **weighted timed automaton** and $\delta > 0$.

Semantics $\mathcal{G}_\delta(\mathcal{A})$

At any state (ℓ, ν, c),

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d \models g$,
2. Environment chooses $d' \in [d - \delta, d + \delta]$,
3. New state is $(\ell', (\nu + d')[R \leftarrow 0], c + \mathrm{Slope}(\ell) \cdot d')$





Guards may not be satisfied **after** perturbation.

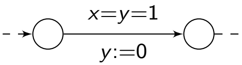
1. **Excess** Robust Game Semantics

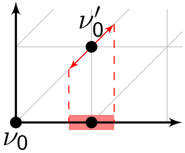
Let $\mathcal{A}$ be a **weighted timed automaton** and $\delta > 0$.

Semantics $\mathcal{G}_\delta(\mathcal{A})$

At any state (ℓ, ν, c),

1. Controller chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d \models g$,
2. Environment chooses $d' \in [d - \delta, d + \delta]$,
3. New state is $(\ell', (\nu + d')[R \leftarrow 0], c + \mathrm{Slope}(\ell) \cdot d')$





Controller's objective: reaching a given location with min cost.

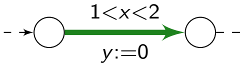
Environment's objective is avoiding the same location, or maximizing cost.

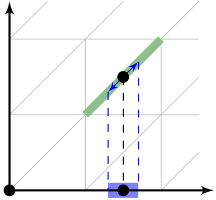
2. **Conservative** Robust Game Semantics

Game Semantics: Controller vs Environment.

Given A and $\delta > 0$, define $\mathcal{G}(A)$ as a game as follows. At any state (ℓ, ν),

1. **Controller** chooses a delay $d \geq \delta$, and an edge $\ell \xrightarrow{g,R} \ell'$, such that $\nu + d + \epsilon \models g$ for all $\epsilon \in [-\delta, \delta]$.
2. **Environment** chooses $\epsilon \in [-\delta, +\delta]$,
3. New state is $(\ell', (\nu + d + \epsilon)[R \leftarrow 0])$.



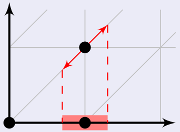


Controller's objective: reaching a given location with min cost.
Environment's objective is avoiding the same location, or maximizing cost.

Robust game semantics: Comparison

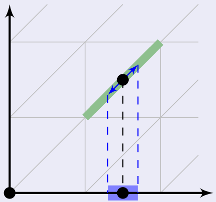
Excess

- Useful to study *perturbations* of a given model
- One can use equality guards: simpler and more abstract models



Conservative

- Consider perturbed runs inside the model
- Model already includes imprecisions as intervals



Cost and Limit-Cost

Given a target state F, the **cost** of a run is

- ∞ if F is not visited,
- the cost at the first visit at F otherwise.

Cost and Limit-Cost

Given a target state F, the **cost** of a run is

- ∞ if F is not visited,
- the cost at the first visit at F otherwise.

Cost of strategy σ

Let $\mathbf{cost}_\delta^F(\mathcal{A}, \sigma)$ be the cost Controller can ensure with strategy σ in $G_\delta(\mathcal{A})$:

- ∞ if F is not guaranteed to be reached
- Otherwise

$$\mathbf{cost}_\delta^F(\mathcal{A}, \sigma) = \sup_{\sigma'} \mathbf{cost}^F(G_\delta(\mathcal{A}), \sigma, \sigma')$$

Cost and Limit-Cost

Given a target state F, the **cost** of a run is

- ∞ if F is not visited,
- the cost at the first visit at F otherwise.

Cost of strategy σ

Let $\mathbf{cost}_\delta^F(\mathcal{A}, \sigma)$ be the cost Controller can ensure with strategy σ in $G_\delta(\mathcal{A})$:

- ∞ if F is not guaranteed to be reached
- Otherwise

$$\mathbf{cost}_\delta^F(\mathcal{A}, \sigma) = \sup_{\sigma'} \mathbf{cost}^F(G_\delta(\mathcal{A}), \sigma, \sigma')$$

What is δ? We assume it is unknown.

How to compare strategies if δ is unknown?

Cost and Limit-Cost

Given a target state F, the **cost** of a run is

- ∞ if F is not visited,
- the cost at the first visit at F otherwise.

Cost of strategy σ

Let $\mathbf{cost}_\delta^F(\mathcal{A}, \sigma)$ be the cost Controller can ensure with strategy σ in $G_\delta(\mathcal{A})$:

- ∞ if F is not guaranteed to be reached
- Otherwise

$$\mathbf{cost}_\delta^F(\mathcal{A}, \sigma) = \sup_{\sigma'} \mathbf{cost}^F(G_\delta(\mathcal{A}), \sigma, \sigma')$$

Limit-Cost of strategy σ

$\mathbf{limcost}^F(\mathcal{A}, \sigma) = \lim_{\delta \to 0} \mathbf{cost}_\delta^F(\mathcal{A}, \sigma, \sigma')$

Optimal Limit Cost: Weighted Timed Automata

– Limit cost: δ is unknown (to be chosen later).
– The limit cost can be approached by any desired precision.

Upper bound problem

Given $\mathcal{A}$, F and a bound λ, does there exist a strategy σ for Controller such that

$$\mathbf{limcost}^F(\mathcal{A},\sigma) \leq \lambda.$$

Optimal Limit Cost: Weighted Timed Automata

Upper bound problem

Given $\mathcal{A}$, F and a bound λ, does there exist a strategy σ for Controller such that

$$\mathbf{limcost}^F(\mathcal{A}, \sigma) \leq \lambda.$$

Theorem 1

In excess robust game semantics,
the upper bound problem is undecidable on weighted timed **automata**.

Contrast: Optimal-cost reachability (without δ) is PSPACE-c for WTA.

Optimal Limit Cost: Weighted Timed Automata

Upper bound problem

Given $\mathcal{A}$, F and a bound λ, does there exist a strategy σ for Controller such that

$$\mathbf{limcost}^F(\mathcal{A}, \sigma) \leq \lambda.$$

Theorem 1

In excess robust game semantics,
the upper bound problem is undecidable on weighted timed **automata**.

Theorem 2

In conservative robust game semantics,
the upper bound problem is PSPACE-c on weighted timed **automata**.

Optimal Limit Cost: Weighted Timed Games

On weighted timed games:
– Environment also suggest delays and edges
– Smallest delay-edge is chosen
– Environment perturbs Controller's delay

Theorem 3

In both semantics,
the upper bound problem is undecidable on weighted timed **games**.

Recall: the problem is undecidable in the exact case.

Optimal Limit Cost: Weighted Timed Games

On weighted timed games:
– Environment also suggest delays and edges
– Smallest delay-edge is chosen
– Environment perturbs Controller's delay

Theorem 3

In both semantics,
the upper bound problem is undecidable on weighted timed **games**.

Recall: the problem is undecidable in the exact case.

Rest of the talk:

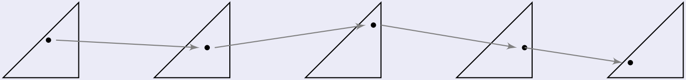
1. Decidability in WTA under conservative semantics
2. Undecidability in WTA under excess semantics

Decidable: WTA under conservative semantics

Consider the exact semantics:

Corner-point abstraction

For any run with cost c

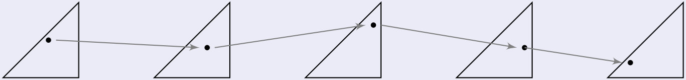


Decidable: WTA under conservative semantics

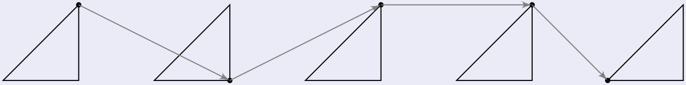
Consider the exact semantics:

Corner-point abstraction

For any run with cost c



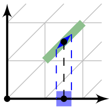
There exists an **integral** run with cost $\leq c$



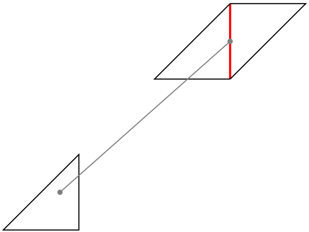
- Optimal-cost is achieved by an integral run.

  Follows from linear programming [Bouyer, Brihaye, Bruyère, Raskin 20007]

Decidable: WTA under conservative semantics

Now, under perturbations... 

Punctual moves (= delays into punctual regions) cannot be guaranteed by Controller:

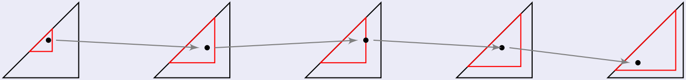


Consider non-punctual region automaton: remove punctual regions

Decidable: WTA under conservative semantics

Corner-point abstraction

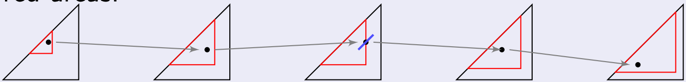
To prevent the accumulation of perturbations, a **play** needs to stay in the red areas:



Decidable: WTA under conservative semantics

Corner-point abstraction

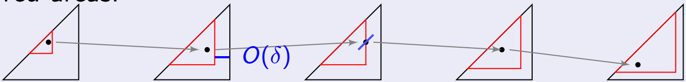
To prevent the accumulation of perturbations, a **play** needs to stay in the red areas:



Decidable: WTA under conservative semantics

Corner-point abstraction

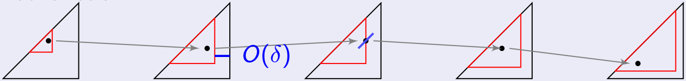
To prevent the accumulation of perturbations, a **play** needs to stay in the red areas:



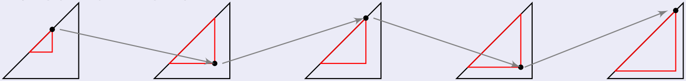
Decidable: WTA under conservative semantics

Corner-point abstraction

To prevent the accumulation of perturbations, a **play** needs to stay in the red areas:



"δ-corner" runs:



Algorithm

- Consider corner-point abstraction without punctual transitions
- Compute a symbolic path with optimal (exact) cost
- $\exists$ a strategy for small enough δ, and the optimal cost can be appoximated arbitrarily

Undecidability Proof for **WTG**

Reduction in the exact case — Bouyer, Brihaye, Markey 2006

Minsky Machine Encoding

Counter values $(n, m) \rightarrow x = \frac{1}{2^n}, y = \frac{1}{2^m}$.

Zero-test is easy: $\xrightarrow{x=1}$

Undecidability Proof for **WTG**

Reduction in the exact case — Bouyer, Brihaye, Markey 2006

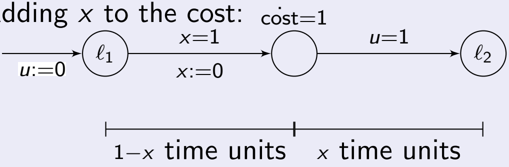
Minsky Machine Encoding

Counter values $(n, m) \to x = \frac{1}{2^n}, y = \frac{1}{2^m}$.

Zero-test is easy: $\xrightarrow{x=1}$

Some Gadgets

Adding x to the cost:



Undecidability Proof for **WTG**

Reduction in the exact case — Bouyer, Brihaye, Markey 2006

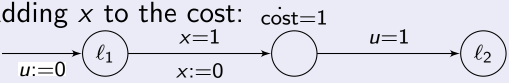
Minsky Machine Encoding

Counter values $(n, m) \rightarrow x = \frac{1}{2^n}, y = \frac{1}{2^m}$.

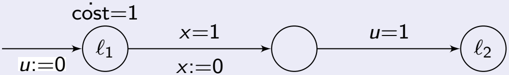
Zero-test is easy: $\xrightarrow{x=1}$

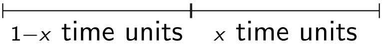
Some Gadgets

Adding x to the cost:



Adding $1-x$ to the cost:

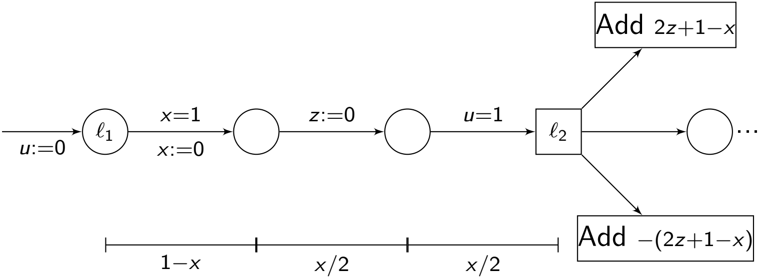




Undecidability Proof for **WTG** - 2

Reduction in the exact case — Bouyer, Brihaye, Markey 2006

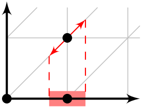
Counter incrementation ($x = \frac{1}{2^n}$ should become $x = \frac{1}{2^{n+1}}$).



At ℓ_2, we are supposed to have $z = x/2$, after which we switch $x \leftrightarrow z$
Otherwise, Player 2 can end the game with cost $|z - x/2|$.

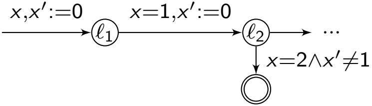
$\Rightarrow$ Player 1 should obey encoding and not cheat

The **Robust** Reduction



- Excess semantics allows Controller to enforce **exact delays**.

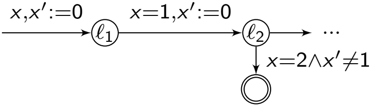
In order to enforce $x = 1$ with no perturbation, we use a duplicate x' of x



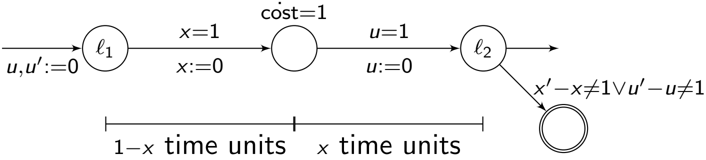
The **Robust** Reduction

- Excess semantics allows Controller to enforce **exact delays**.

In order to enforce $x = 1$ with no perturbation, we use a duplicate x' of x

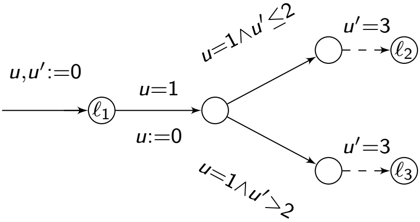


Adding x to the cost: Assume $x = x'$



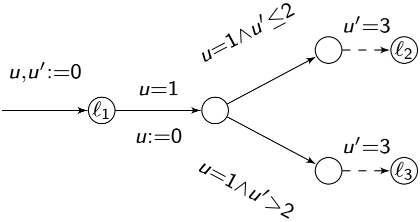
The **Robust** Reduction - 2

- Excess semantics may allow Environment to chose the successor state.

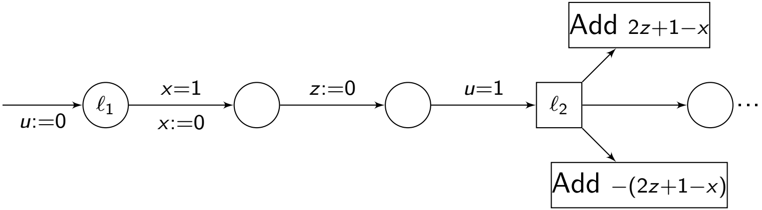


The **Robust** Reduction - 2

- Excess semantics may allow Environment to chose the successor state.



Counter incrementation ($x = \frac{1}{2^n}$ should become $x = \frac{1}{2^{n+1}}$).



Summary

Results: (Optimal-cost) Reachability

	Exact	**Robust (Excess)**	**Robust (Conservative)**
TA	PSPACE-c [AD94]	EXPTIME-c [BMS12]	PSPACE-c [SBMR13]
WTA	PSPACE-c [ALP01,BFH+01]	Undecidable	PSPACE-c
WTG	Undecidable [BBR05,BBM06]	Undecidable	Undecidable

- Robust optimal-cost reachability: Achieved (for conserv. sem.)
- Obtain decidability for games: Failed

Open

- WTA and WTG with closed guards?
- Study other robust semantics, or semantic restrictions to obtain decidability for games