Opacity Problems in Multi-Energy Timed Automata

Lydia Bakiri, Étienne André

Université Sorbonne Paris Nord, LIPN, CNRS UMR 7030 Villetaneuse, France

LIX, CNRS, École polytechnique, Institut Polytechnique de Paris, Palaiseau, France

Motivations

- The observation of some elements related to the functionning of a system can be enough to extract sensitive data

Motivations

- The observation of some elements related to the functionning of a system can be enough to extract sensitive data
- Pentagon Pizza theory: a spike in pizza orders from the pentagon foreshadow a major crisis

Context: Timing attacks over programs

```
1  # input pwd     : Real password
2  # input attempt: Tentative password
3  for i = 0 to min(len(pwd), len(attempt)) - 1 do
4      # comparison check takes 1 time unit
5      if pwd[i] =/= attempt[i] then
6          return false
7  done
8  return true
9
```

Context: Timing attacks over programs

```
1  # input pwd     : Real password
2  # input attempt: Tentative password
3  for i = 0 to min(len(pwd), len(attempt)) - 1 do
4      # comparison check takes 1 time unit
5      if pwd[i] =/= attempt[i] then
6          return false
7  done
8  return true
9
```

pwd	p	a	t	e	H	e	n	a	f	f
attempt	p	a	s	t	a					

Execution time:

Context: Timing attacks over programs

```
1  # input pwd     : Real password
2  # input attempt: Tentative password
3  for i = 0 to min(len(pwd), len(attempt)) - 1 do
4      # comparison check takes 1 time unit
5      if pwd[i] =/= attempt[i] then
6          return false
7  done
8  return true
9
```

pwd	p	a	t	e	H	e	n	a	f	f
attempt	p	a	s	t	a					

Execution time: 1

Context: Timing attacks over programs

```
1  # input pwd     : Real password
2  # input attempt: Tentative password
3  for i = 0 to min(len(pwd), len(attempt)) - 1 do
4      # comparison check takes 1 time unit
5      if pwd[i] =/= attempt[i] then
6          return false
7  done
8  return true
9
```

pwd	p	a	t	e	H	e	n	a	f	f
attempt	p	a	s	t	a					

Execution time: $1 + 1$

Context: Timing attacks over programs

```
1  # input pwd     : Real password
2  # input attempt: Tentative password
3  for i = 0 to min(len(pwd), len(attempt)) - 1 do
4      # comparison check takes 1 time unit
5      if pwd[i] =/= attempt[i] then
6          return false
7  done
8  return true
9
```

pwd	p	a	t	e	H	e	n	a	f	f
attempt	p	a	s	t	a					

Execution time: $1 + 1 + 1 = 3$

Context: Timing attacks over programs

```
1  # input pwd     : Real password
2  # input attempt: Tentative password
3  for i = 0 to min(len(pwd), len(attempt)) - 1 do
4      # comparison check takes 1 time unit
5      if pwd[i] =/= attempt[i] then
6          return false
7  done
8  return true
9
```

pwd	p	a	t	e	H	e	n	a	f	f
attempt	p	a	s	t	a					

Execution time: $1 + 1 + 1 = 3$

- Problem: The execution time is proportional to the number of consecutive correct characters from the beginning of `attempt`

Motivations

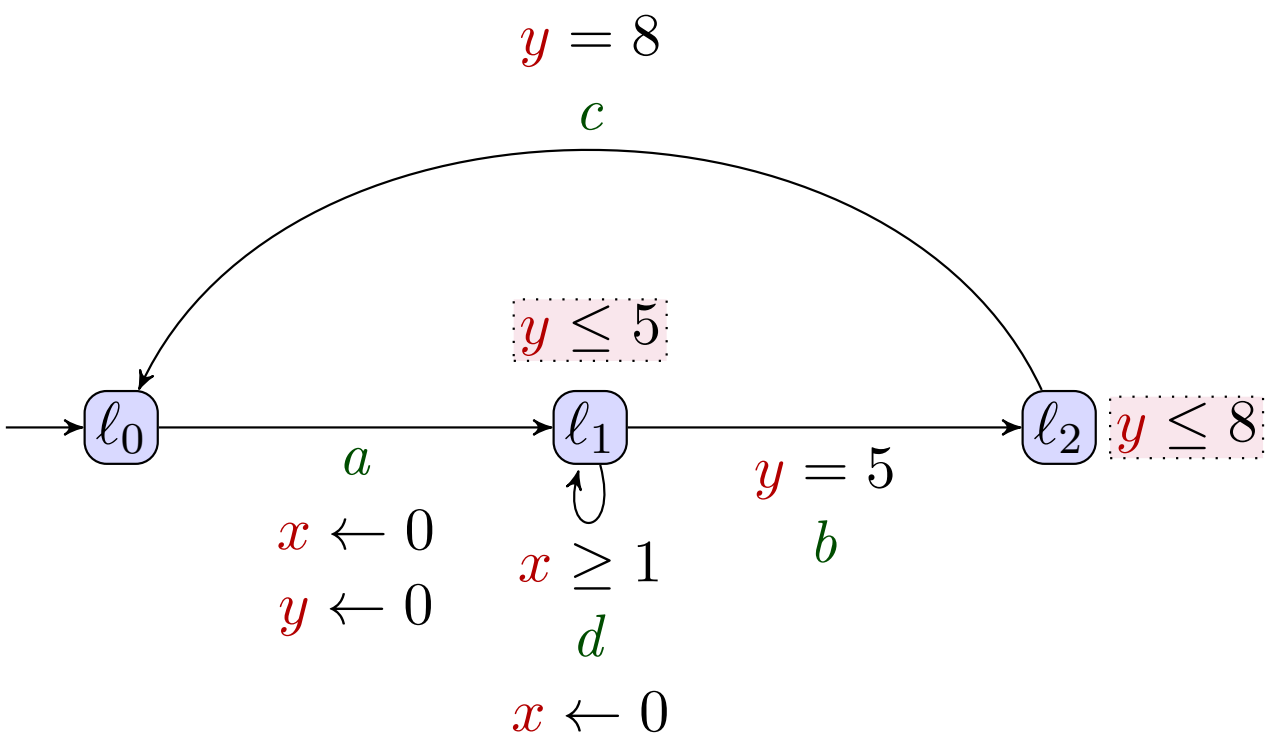
- Opacity captures the attacker's ability to infer secret information based on publicly observable behaviours
- Problems of opacity studied by Franz Cassez on Timed Automata[1]
  - Observation: set of actions together with their timestamp
- Alternative definition given by Étienne André[2][3]
  - Observation: execution time

[1] The dark side of Timed Opacity, 2009

[2] The Bright Side of Timed Opacity, 2024

[3] Configuring Timing Parameters to Ensure Execution-Time Opacity in Timed Automata, 2024.

Timed automata



- Set of clocks
- Clocks can be reset
- Guards and invariants

Energy based attacks over programs

```
1   # input pwd     : Real password
2   # input attempt: Tentative password
3   for i = 0 to min(len(pwd), len(attempt)) - 1 do
4       # comparison check consumes 1 energy unit
5       if pwd[i] =/= attempt[i] then
6           wait(len(attempt) - 1 - i)
7           return false
8   done
9   return true
10
```

Execution time = relative to the attempt size

Energy based attacks over programs

```
1   # input pwd     : Real password
2   # input attempt: Tentative password
3   for i = 0 to min(len(pwd), len(attempt)) - 1 do
4         # comparison check consumes 1 energy unit
5         if pwd[i] =/= attempt[i] then
6             wait(len(attempt) - 1 - i)
7             return false
8   done
9   return true
10
```

Execution time = relative to the attempt size

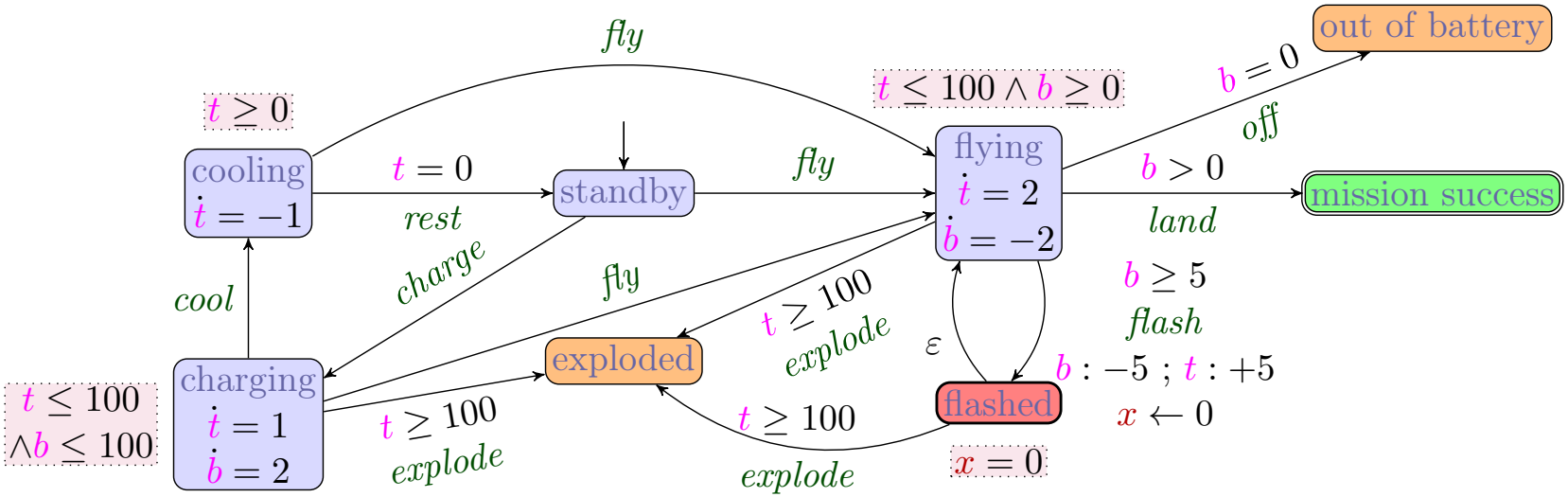
pwd	p	a	t	e	H	e	n	a	f	f
attempt	p	a	s	t	a					

Energy consumed: $1 + 1 + 1 = 3 \rightarrow$ proportional to the prefix match length

Goal of our studies

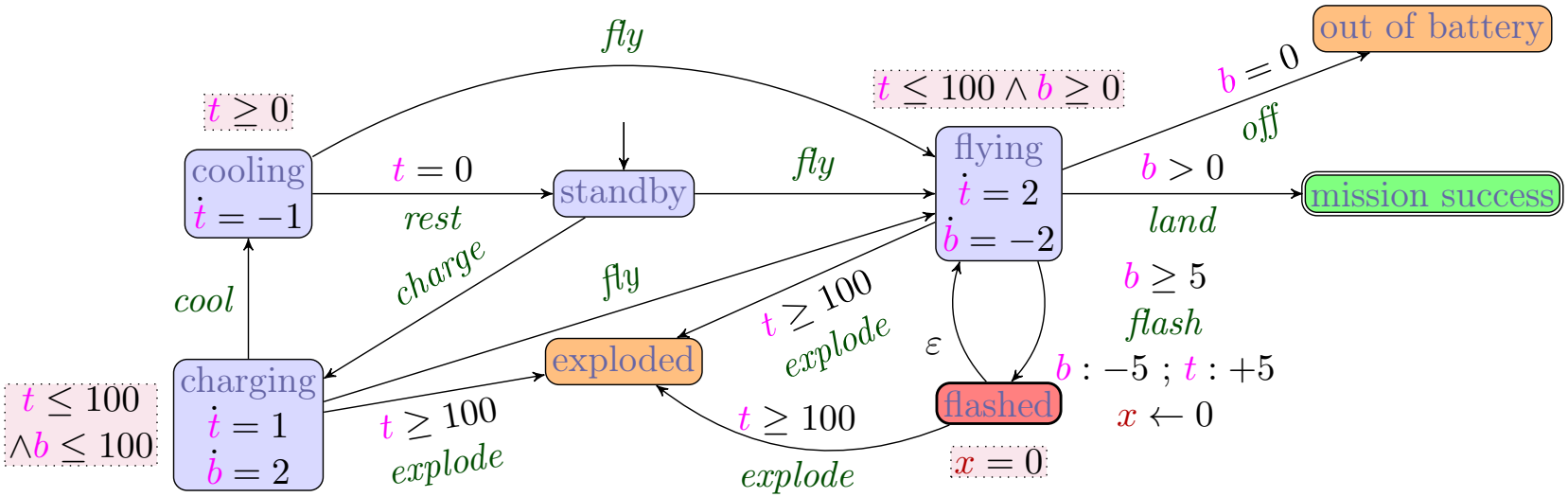
- Goal: study of opacity problems with adding observation of energy or cost constraints:
  - Introducing a more generalized model: guarded Multi-Energy Timed Automata (guarded-META)
  - Defining new opacity problems taking into account the consumption and accumulation of ressources
  - The attacker has access to the model and final valuation of energy variables

Guarded-META



- Energy variables: temperature t, battery b
  - Energy variables cannot drop below 0
- Continuous evolution in locations
  - In charging location, the battery and the temperature continuously increase at different rates.
- Discrete evolution in transitions
  - Flashing decreases the battery and increases the temperature instantaneously
- Guards and Invariants on energy variables

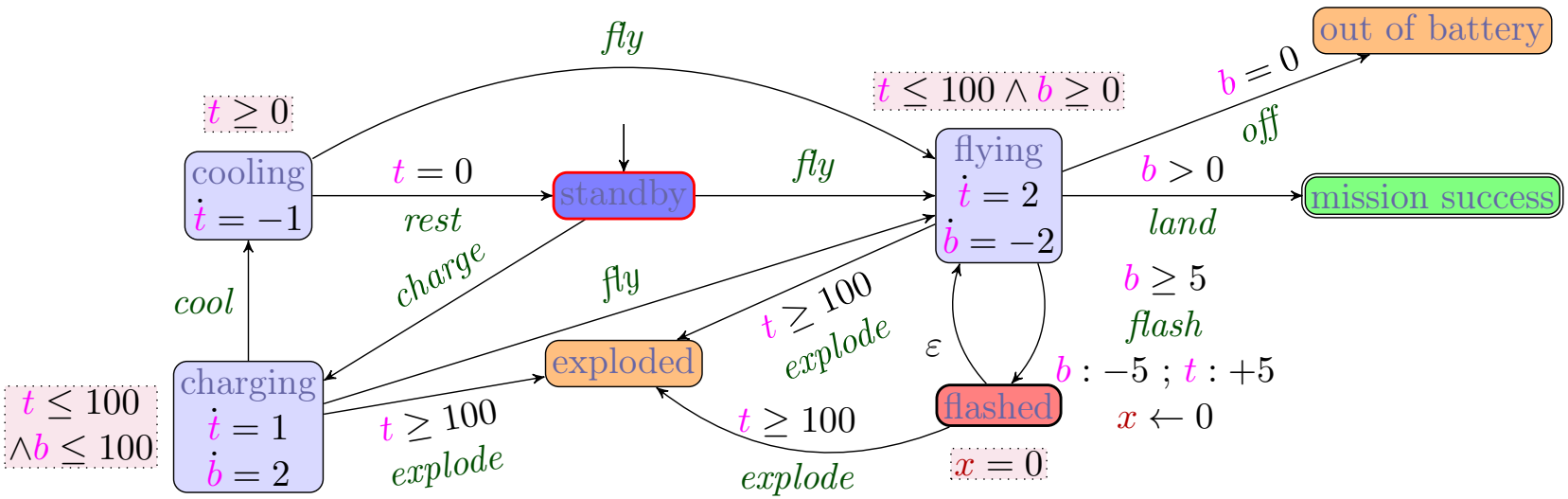
Guarded-META



State of a run: (ℓ, w, v) where

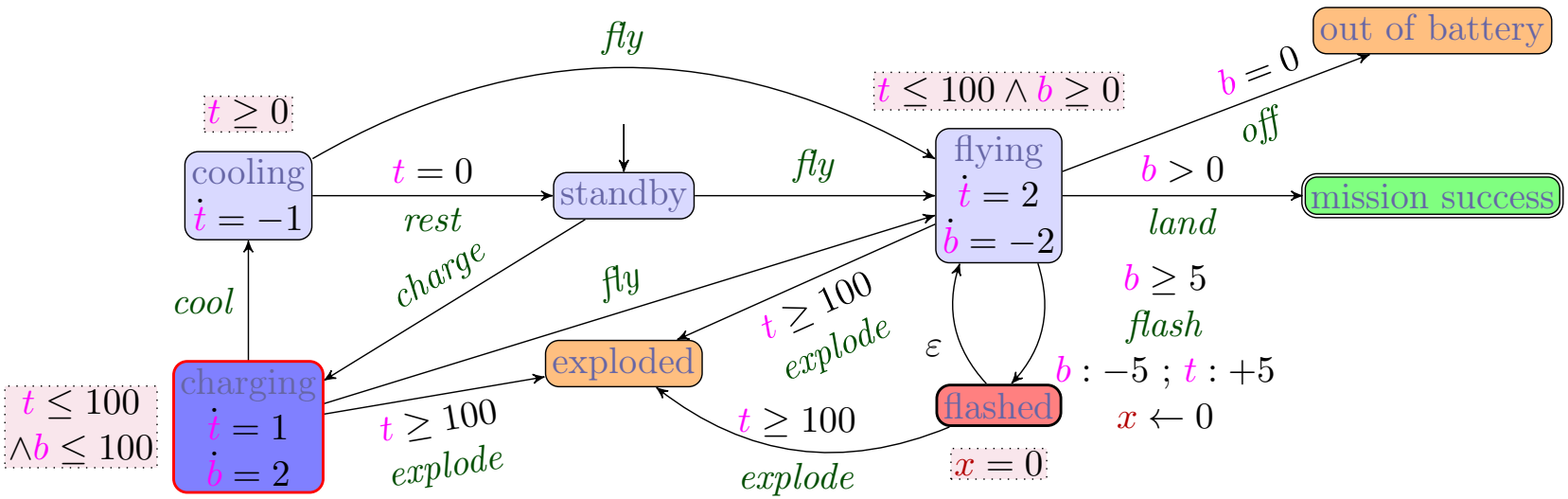
- ℓ: current location
- w: values of each clock variable
- v: values of each energy variable

Guarded-META



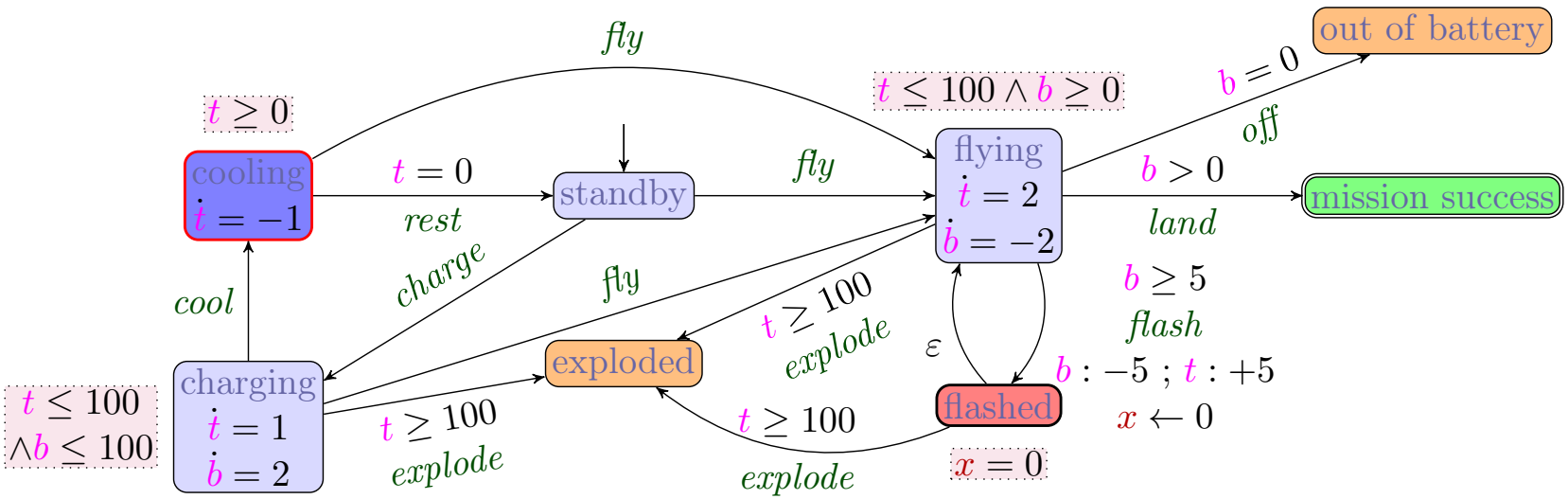
$\rho = (\text{standby}, 0, [0, \vec{0}])$

Guarded-META



$$\rho = (\text{standby}, 0, [0,\vec{0}]) \overset{(0,charge)}{\longmapsto} (\text{charging}, 0, [0,\vec{0}])$$

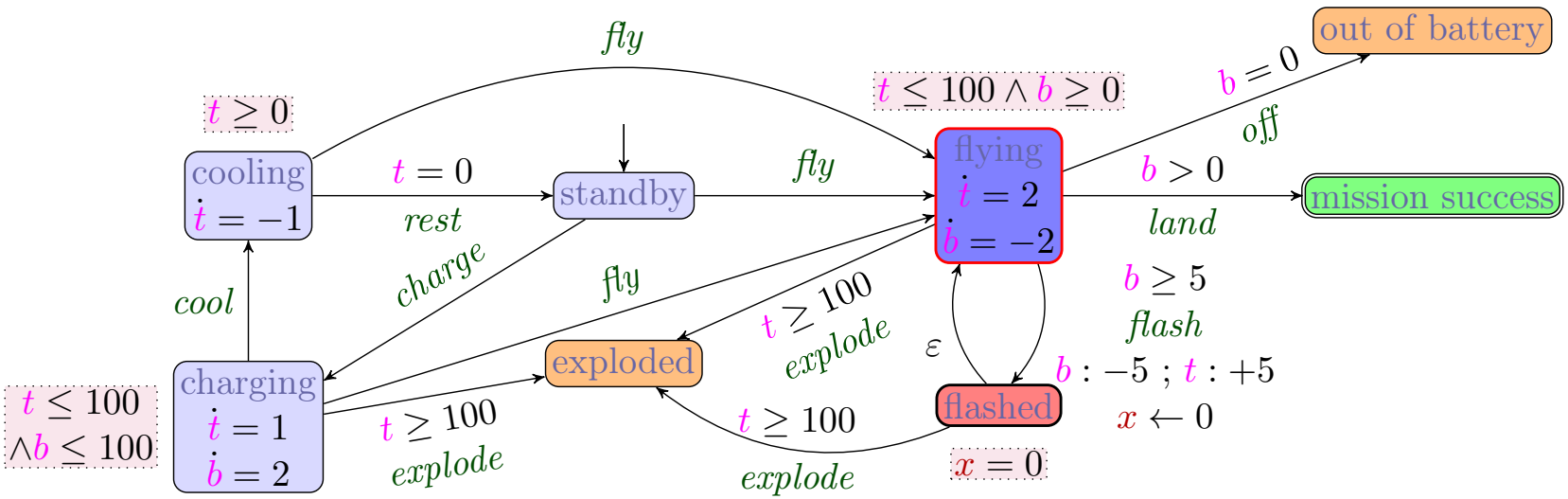
Guarded-META



$$\rho = (\text{standby}, 0, [0, \vec{0}]) \overset{(0, charge)}{\rightarrowtail} (\text{charging}, 0, [0, \vec{0}]) \overset{(10, cool)}{\rightarrowtail}$$

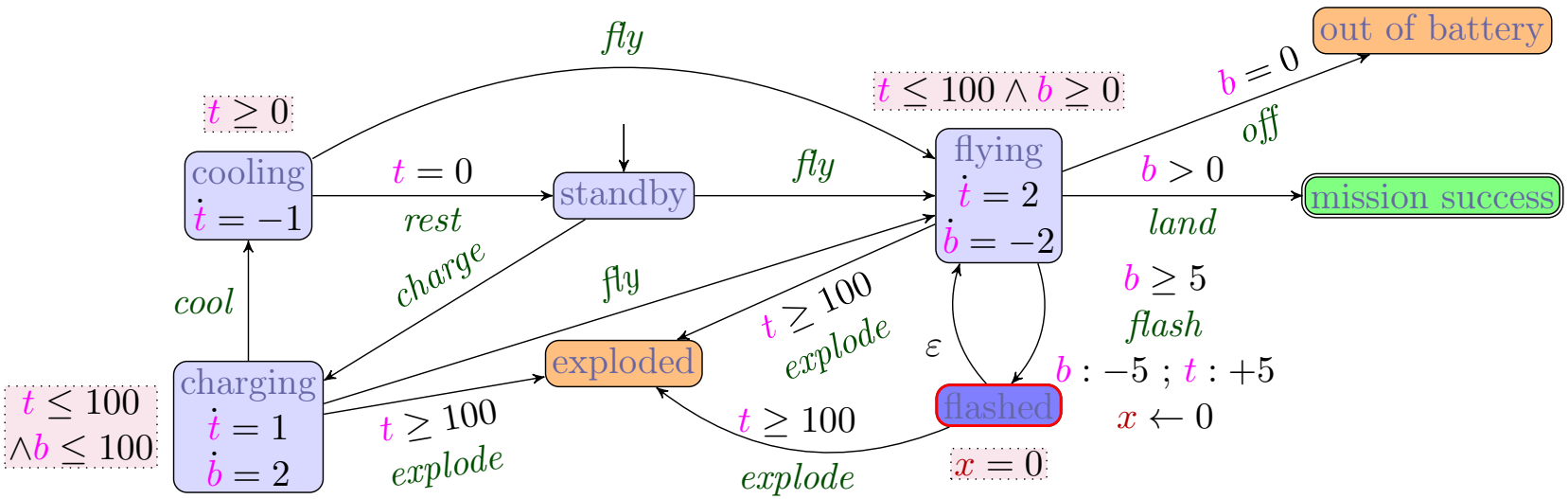
$$(\text{cooling}, 10, [10, \vec{20}])$$

Guarded-META



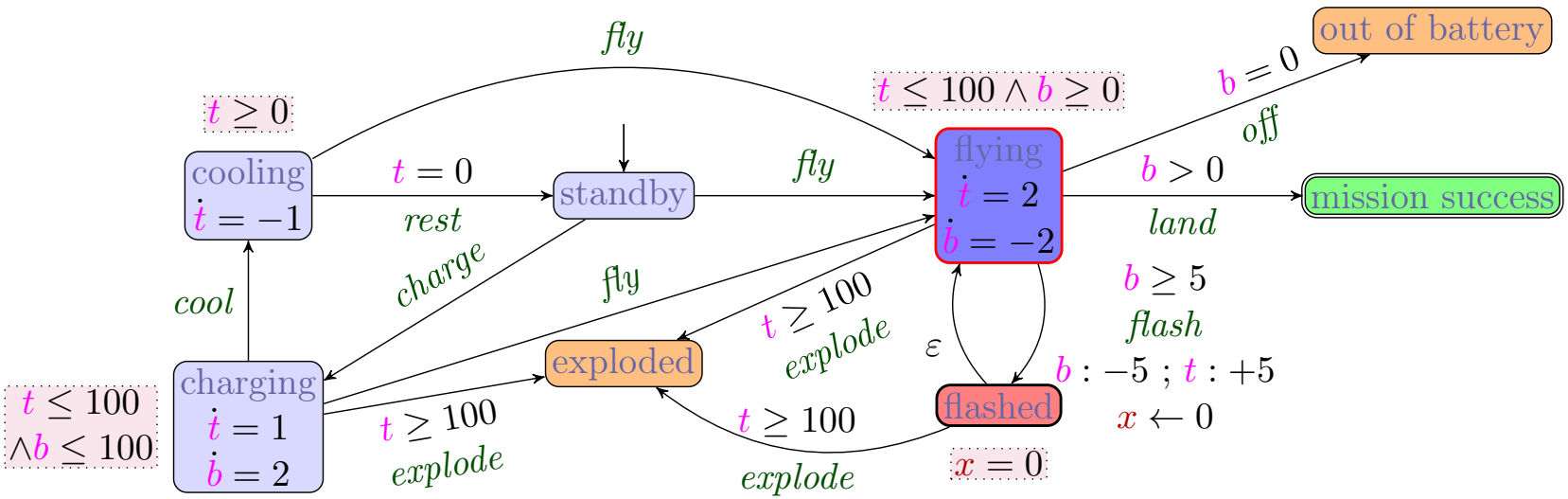
$$\rho = (\text{standby}, 0, [0, \vec{0}]) \overset{(0, charge)}{\rightarrowtail} (\text{charging}, 0, [0, \vec{0}]) \overset{(10, cool)}{\rightarrowtail}$$
$$(\text{cooling}, 10, [10, \vec{20}]) \overset{(5, fly)}{\rightarrowtail} (\text{flying}, 15, [5, \vec{20}])$$

Guarded-META



$$\rho = (\text{standby}, 0, [0, \vec{0}]) \overset{(0, charge)}{\longmapsto} (\text{charging}, 0, [0, \vec{0}]) \overset{(10, cool)}{\longmapsto}$$
$$(\text{cooling}, 10, [10, \vec{20}]) \overset{(5, fly)}{\longmapsto} (\text{flying}, 15, [5, \vec{20}]) \overset{(2, flash)}{\longmapsto} (\text{flashed}, 0, [14, \vec{11}])$$

Guarded-META

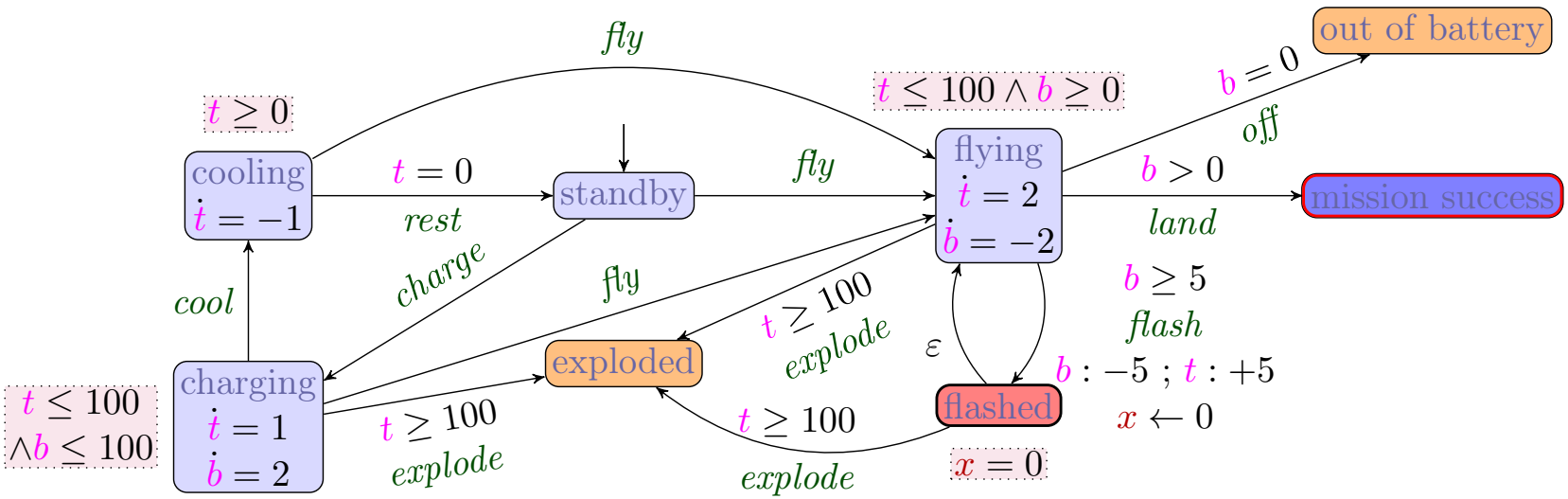


$$\rho = (\text{standby}, 0, [0, \vec{0}]) \overset{(0, charge)}{\longmapsto} (\text{charging}, 0, [0, \vec{0}]) \overset{(10, cool)}{\longmapsto}$$

$$(\text{cooling}, 10, [10, \vec{20}]) \overset{(5, fly)}{\longmapsto} (\text{flying}, 15, [5, \vec{20}]) \overset{(2, flash)}{\longmapsto} (\text{flashed}, 0, [14, \vec{11}]) \overset{(0, \varepsilon)}{\longmapsto}$$

$$(\text{flying}, 0, [14, \vec{11}])$$

Guarded-META



$$\rho = (\text{standby}, 0, [0, \vec{0}]) \overset{(0, charge)}{\rightarrowtail} (\text{charging}, 0, [0, \vec{0}]) \overset{(10, cool)}{\rightarrowtail}$$
$$(\text{cooling}, 10, [10, \vec{20}]) \overset{(5, fly)}{\rightarrowtail} (\text{flying}, 15, [5, \vec{20}]) \overset{(2, flash)}{\rightarrowtail} (\text{flashed}, 0, [14, \vec{11}]) \overset{(0, \varepsilon)}{\rightarrowtail}$$
$$(\text{flying}, 0, [14, \vec{11}]) \overset{(0, land)}{\rightarrowtail} (\text{mission success}, 0, [14, \vec{11}]).$$

private and publics runs of a guarded-META

- A run ρ is *private* if it visited the private location
- A run ρ is *public* if it didn't visit the private location

private and publics runs of a guarded-META

- A run ρ is *private* if it visited the private location
- A run ρ is *public* if it didn't visit the private location

- $EVisit^{priv}(\mathcal{A})$: set of all the final energy values of the private runs of $\mathcal{A}$
- $EVisit^{pub}(\mathcal{A})$: set of all the final energy values of the public runs of $\mathcal{A}$

private and publics runs of a guarded-META

- A run ρ is *private* if it visited the private location
- A run ρ is *public* if it didn't visit the private location

- $EVisit^{priv}(\mathcal{A})$: set of all the final energy values of the private runs of $\mathcal{A}$
- $EVisit^{pub}(\mathcal{A})$: set of all the final energy values of the public runs of $\mathcal{A}$

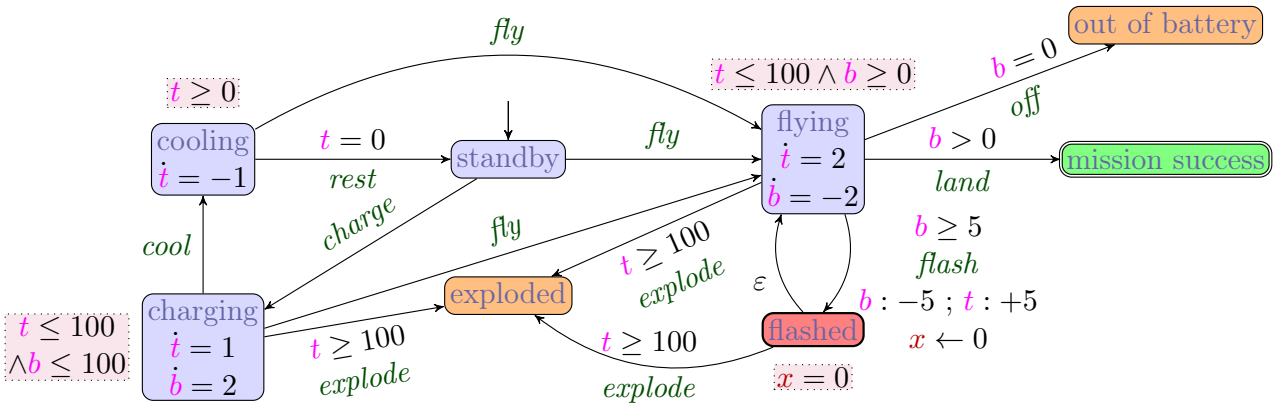
- A guarded META $\mathcal{A}$ is **fully-opaque w.r.t. energy** (*or fully-EN-opaque*) when $EVisit^{priv}(\mathcal{A}) = EVisit^{pub}(\mathcal{A})$.
- In other terms: for every final energy values of a run, it is indistiguishable whether the private location is visited.

Opacity problem

We define the following decision problem:

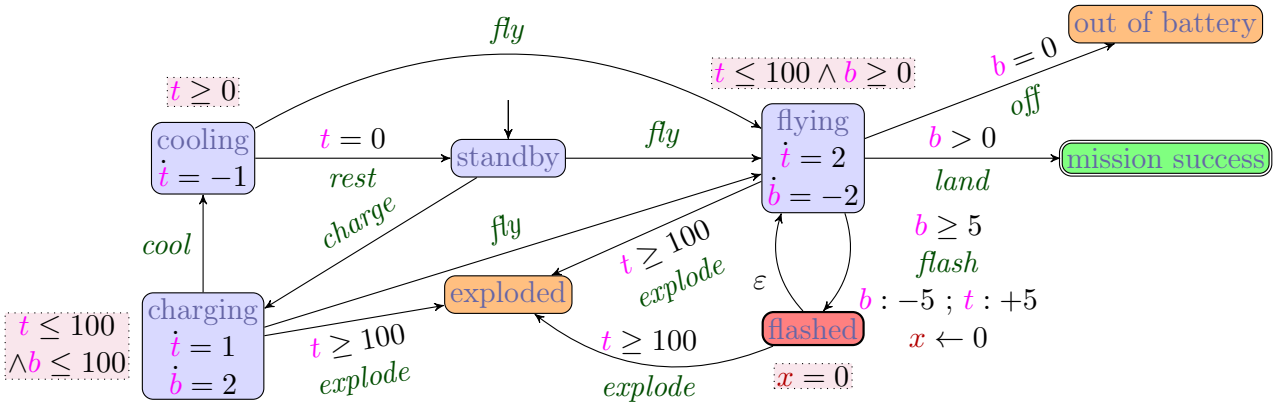
Full-EN-opacity problem:
INPUT: A guarded META $\mathcal{A}$
PROBLEM: Is $\mathcal{A}$ fully-EN-opaque?

Example



- Full-EN-Opacity ?

Example



- Full-EN-Opacity ? ✗

$$\rho_{pub} = (\text{standby}, 0, [0,\vec{0}]) \overset{(0,charge)}{\rightarrowtail} (\text{charging}, 0, [0,\vec{0}]) \overset{(0.5,cool)}{\rightarrowtail}$$
$$(\text{cooling}, 0.5, [0.5,\vec{1}]) \overset{(0.5,fly)}{\rightarrowtail} (\text{flying}, 1, [0,\vec{1}]) \overset{(0,land)}{\rightarrowtail} (\text{mission success}, 1, [0,\vec{1}])$$

- If a run is private, then the final value of temperature is higher than 5.

Contributions: Undecidability

Theorem

full-EN-opacity for discrete guarded METAs[a] *with 2 energy variables and 1 clock is undecidable.*

[a] discrete guarded-META: guarded-META with only discrete evolution of energy

Proof.

Reduction from the halting problem for a two-counter machine, which is undecidable (see paper). □

→ restrict ourselves to a single energy or to only positive rate energy

Decidability for positive discrete META

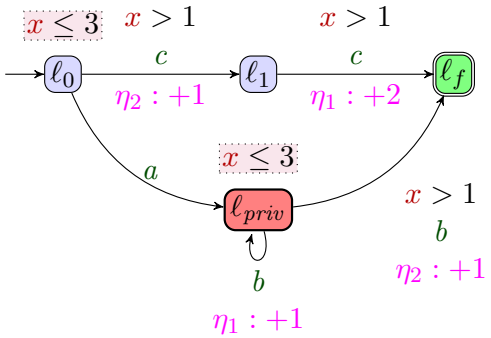
Theorem

Full-EN-opacity is decidable in 2-EXPSPACE for positive discrete META[a].

[a] positive discrete guarded-META: discrete guarded-META where the energy variable can only increase

Proof idea

Given a positive discrete META



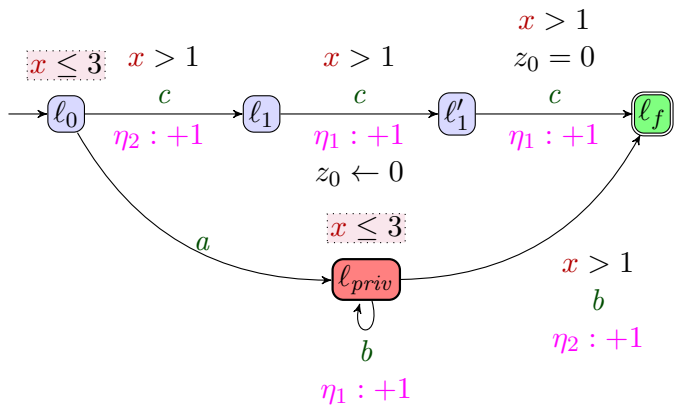
[a] R. Alur and D. L. Dill., A theory of timed automata, 1994

[b] Kopczynski, Eryk, Parikh Images of Grammars: Complexity and Applications, 2010

Proof idea

Given a positive discrete META

1. Construct of two non-deterministic finite automata:
   - Remove updates "$+N$"



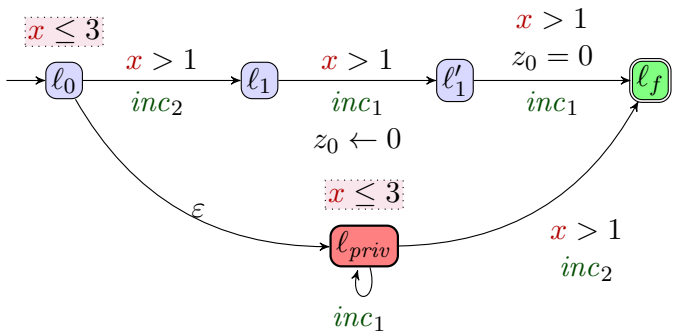
[a] R. Alur and D. L. Dill., A theory of timed automata, 1994

[b] Kopczynski, Eryk, Parikh Images of Grammars: Complexity and Applications, 2010

Proof idea

Given a positive discrete META

1. Construct of two non-deterministic finite automata:
   - Remove updates "$+N$"
   - Add increment symbols inc_i (denote an increment of energy i)



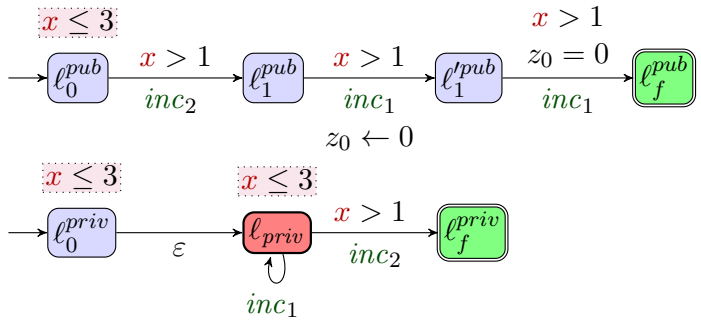
[a] R. Alur and D. L. Dill., A theory of timed automata, 1994

[b] Kopczynski, Eryk, Parikh Images of Grammars: Complexity and Applications, 2010

Proof idea

Given a positive discrete META

1. Construct of two non-deterministic finite automata:
   - Remove updates "$+N$"
   - Add increment symbols inc_i (denote an increment of energy i)
   - Construct the untimed language of private runs (using region graph[a])
   - Construct the untimed language of public runs



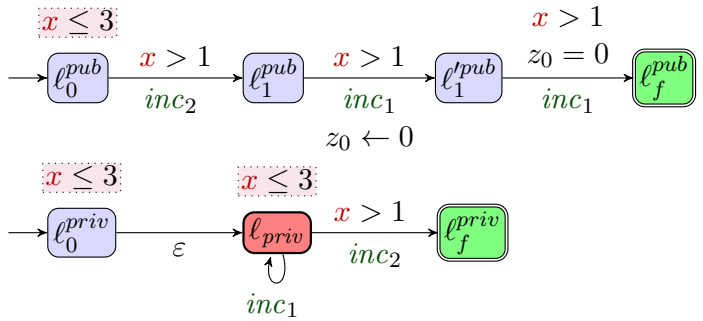
[a] R. Alur and D. L. Dill., A theory of timed automata, 1994

[b] Kopczynski, Eryk, Parikh Images of Grammars: Complexity and Applications, 2010

Proof idea

Given a positive discrete META

1. Construct of two non-deterministic finite automata:
   - Remove updates "$+N$"
   - Add increment symbols inc_i (denote an increment of energy i)
   - Construct the untimed language of private runs (using region graph[a])
   - Construct the untimed language of public runs
2. Compare the number occurences of each symbols inc_i using Parikh's image [b]



$$EVisit^{pub}(\mathcal{A}) = \text{Parikh}(\mathcal{L}(\mathcal{RA}(\mathcal{A}_{pub}))) = \{(2,1)\}$$

$$EVisit^{priv}(\mathcal{A}) = \text{Parikh}(\mathcal{L}(\mathcal{RA}(\mathcal{A}_{priv}))) = \{(n,1), n \in \mathbb{N}\}$$

$Parikh(\mathcal{L}) = \{(|\omega|_{inc_1}, |\omega|_{inc_2}, ..., |\omega|_{inc_n}), \omega \in \mathcal{L}\}$

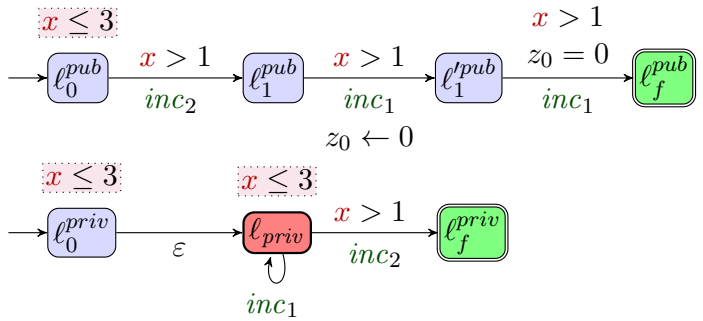
[a] R. Alur and D. L. Dill., A theory of timed automata, 1994

[b] Kopczynski, Eryk, Parikh Images of Grammars: Complexity and Applications, 2010

Proof idea

Given a positive discrete META

1. Construct of two non-deterministic finite automata:
   - Remove updates "$+N$"
   - Add increment symbols inc_i (denote an increment of energy i)
   - Construct the untimed language of private runs (using region graph[a])
   - Construct the untimed language of public runs
2. Compare the number occurences of each symbols inc_i using Parikh's image [b]



$$EVisit^{pub}(\mathcal{A}) = \text{Parikh}(\mathcal{L}(\mathcal{RA}(\mathcal{A}_{pub}))) = \{(2,1)\}$$

$$EVisit^{priv}(\mathcal{A}) = \text{Parikh}(\mathcal{L}(\mathcal{RA}(\mathcal{A}_{priv}))) = \{(n,1), n \in \mathbb{N}\}$$

$$Parikh(\mathcal{L}) = \{(|\omega|_{inc_1}, |\omega|_{inc_2}, ..., |\omega|_{inc_n}), \omega \in \mathcal{L}\}$$

$EVisit^{pub}(\mathcal{A}) \neq EVisit^{priv}(\mathcal{A})$
→ Not fully EN-Opaque

[a] R. Alur and D. L. Dill., A theory of timed automata, 1994

[b] Kopczynski, Eryk, Parikh Images of Grammars: Complexity and Applications, 2010

Decidability for discrete ETA

Theorem

Full-EN-opacity is decidable in 2-EXPSPACE for discrete ETAs[a].

[a] discrete ETA: discrete guarded-META with one energy and not energy constraint

Proof idea.

1. adding symbols *inc* et *dec* to denote energy increments and decrements,
2. construction of the untimed language of private runs and public runs
3. transformation into unary pushdown automata to simulate a one-counter machine (push for increment, pop for decrement)
4. Language inclusion of unary PDA is decidable[a]

□

[a] Chistikov and Majumdar, Unary Pushdown Automata and Straight-Line Programs, 2014

Other problems studied (see paper)

- ∃ **EN-Opacity**
  - Check if at least one final energy valuation is shared between a public run and a private run
- **Weak EN-Opacity**
  - It is OK to deduce that the secret location was NOT visited
- **ET-EN-opacity**
  - Observing the final energy valuation and duration of runs
- **DE-opacity** (Discrete-Energy opacity)
  - Observing the energy valuation every time unit
- **bDE-opacity** (buffered-DE opacity)
  - Observing the buffered energy changes every time unit

Conclusion

Contributions:

- Introduction of a new model with time and energy
- Definition of new opacity notions
- Decidability results

Future works:

- Study missing results
- Energy Parametrisation
- Expiring Opacity[4]
  - The secret can be revealed after some time has passed
- implementation in IMITATOR software

[4] André et Al., Configuring Timing Parameters to Ensure Execution-Time Opacity in Timed Automata, 2024

Our results in a nutshell

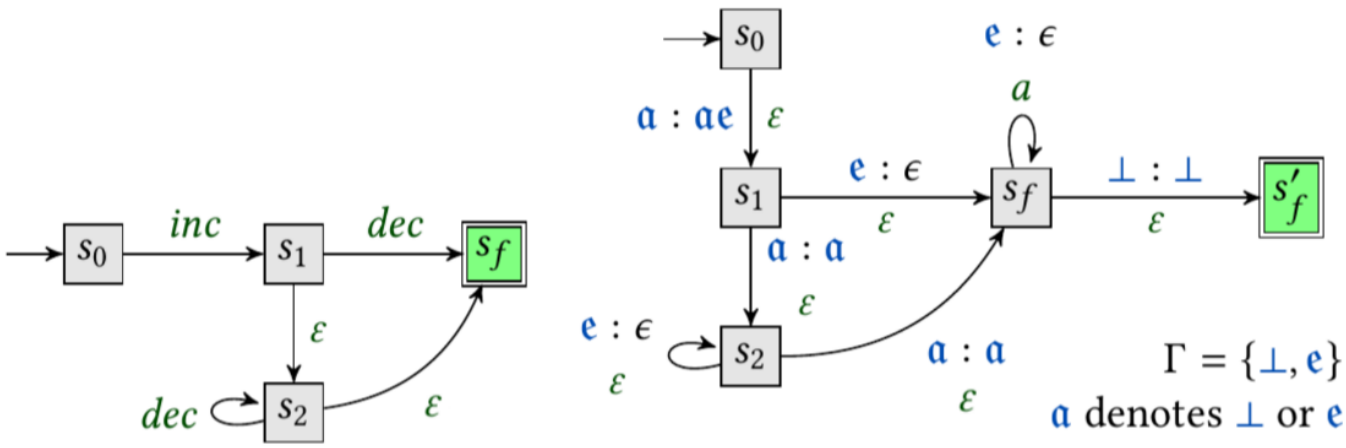
For each $\gamma \in \{\exists, \text{weak}, \text{full}\}$	γ-EN-opacity	γ-ET-EN-opacity
Discrete positive METAs	2EXPSPACE	2EXPSPACE
Discrete positive guarded METAs	3EXPSPACE	3EXPSPACE
Discrete (guarded) ETAs	decidable	
Discrete guarded METAs	undecidable	

Table 1: Decidability of (ET-)EN-opacity in discrete METAs

	DE-opacity		bDE-opacity	
	$\exists$	Weak / full	$\exists$	Weak / full
Discrete positive (guarded) ETAs	EXPSPACE		2EXPSPACE*	
Discrete positive (guarded) METAs	decidable		2EXPSPACE*	
Discrete ETAs			2EXPSPACE	
Discrete METAs				
Discrete guarded METAs	undecidable			

Table 2: Decidability of (b)DE-opacity in discrete METAs

Construction of the PDA



(a) An NFA **(b) Transformed PDA**

◂ Return