27 January 2026 | APR Seminar | Paris, France

Detecting Timing Leaks of Programs

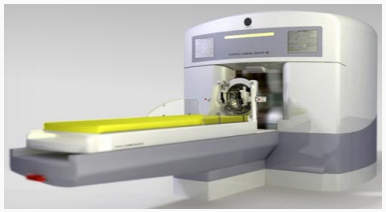
using Parametric Timed Model Checking

Dylan Marinho

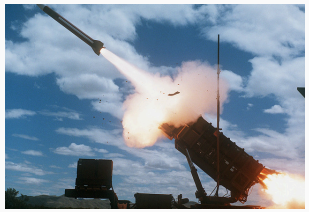
Sorbonne Université, CNRS UMR 7606, LIP6

Context: Verifying complex timed systems

- **Critical** systems: Failures may result in **dramatic** consequences
- Need for early bug detection
  - Bugs discovered when final testing: **expensive**
  - Need for a thorough **specification** and **verification** phase



Therac-25
(USA, 1980s)



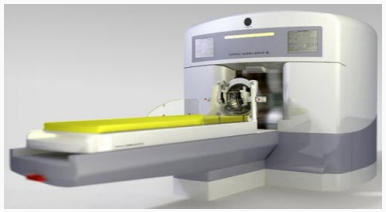
MIM-104 Pat. Mis. Fail.
(Iraq, 1991)

Sleipner A offshore platform
(Norway, 1991)

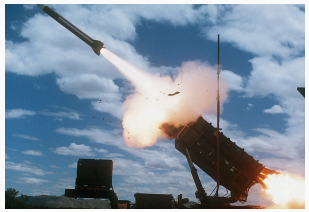
Ariane flight V88
(France, 1996)

Context: Verifying complex timed systems

- **Critical** systems: Failures may result in **dramatic** consequences
- Need for early bug detection
  - Bugs discovered when final testing: **expensive**
  - Need for a thorough **specification** and **verification** phase



Therac-25
(USA, 1980s)



MIM-104 Pat. Mis. Fail.
(Iraq, 1991)

Sleipner A offshore platform
(Norway, 1991)

Ariane flight V88
(France, 1996)

- Verification is needed to ensure the absence of bugs

Context: Side-channel attacks

Side-channel attacks

Threats to a system using non-algorithmic weaknesses

- *e.g., power consumption, electromagnetic radiation, cache usage, **timing**, accoustic emissions, temperature variations, etc.*

[1] home.xnet.com/~warinner/pizzacites.html (1990s)

Context: Side-channel attacks

Side-channel attacks

Threats to a system using non-algorithmic weaknesses

- *e.g., power consumption, electromagnetic radiation, cache usage, **timing**, accoustic emissions, temperature variations, etc.*

Example

Number of pizzas (and order time) ordered by the white house prior to major war announcements[1]

[1] home.xnet.com/~warinner/pizzacites.html (1990s)

Context: Side-channel attacks

Side-channel attacks

Th

ns,

Example

Nu

ann

house prior to major war

MIDDLE EAST

What is the Pizza Meter? The signal that spiked on Saturday during Iran's attack on Israel

On April 13 there was a surge in pizza orders from U.S. government buildings such as the White House, the Pentagon, and the Department of Defense.

[1] home.xnet.com/~warinner/pizzacites.html (1990s)

Context: Side-channel attacks

Side-channel atta

Example

[1]home.xnet.com/~warinner/pizzacites.html (1990s)

Context: Side-channel attacks

Side-channel atta

Example

Papa Johns Pizza

Overv | Overview | Menu | Reviews | Photos | Updat

Popu

Popular times

Tuesdays

Live 5 PM

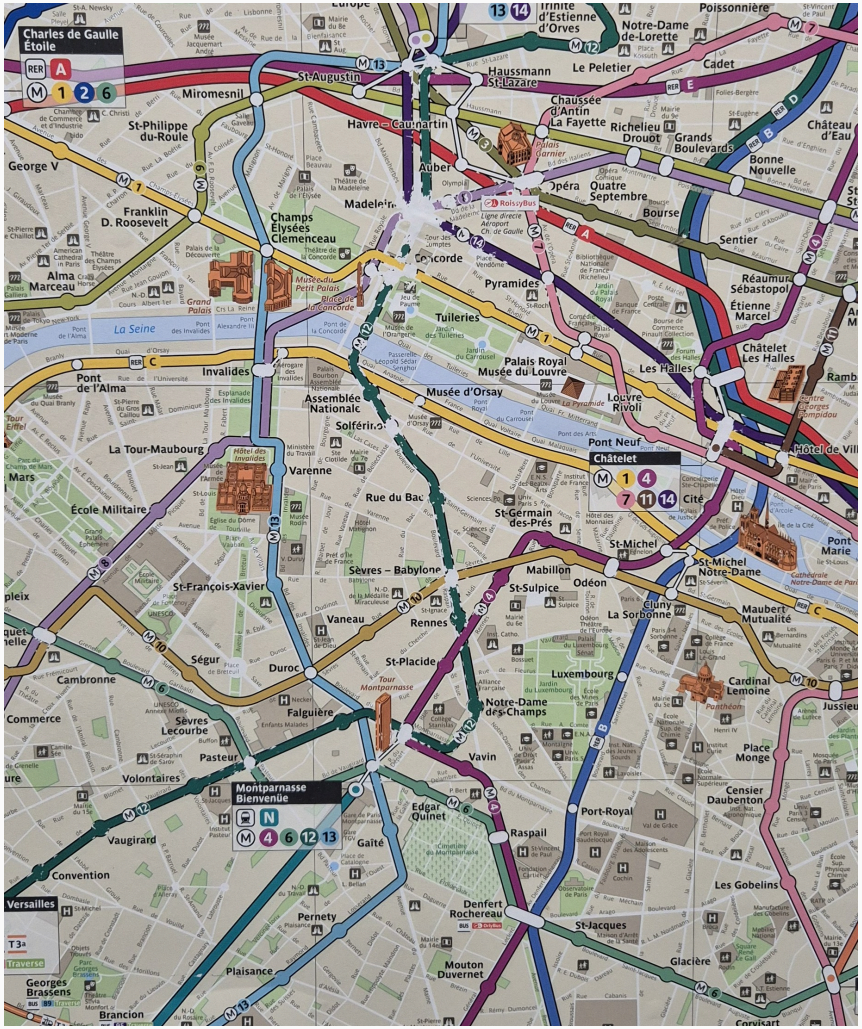
Busier than usual · Usually no wait

9a 12p 3p 6p 9p

No wait

[1]home.xnet.com/~warinner/pizzacites.html (1990s)

Your turn: where was this picture taken?



Context: Timing attacks over programs

```c
// input pwd    : Real password
// input attempt: Tentative password
for (i = 0; i < min(len(pwd), len(attempt)); i++) {
    if(pwd[i] != attempt[i]){
        return false
    }
}
return true
```

pwd	A	L	M	A	S	T	Y
attempt	A	L	S	O	C		

🕑 Execution time (ET):

Context: Timing attacks over programs

```c
// input pwd    : Real password
// input attempt: Tentative password
for (i = 0; i < min(len(pwd), len(attempt)); i++) {
    if(pwd[i] != attempt[i]){
        return false
    }
}
return true
```

pwd	A	L	M	A	S	T	Y
attempt	A	L	S	O	C		

🕒 Execution time (ET): ε

Context: Timing attacks over programs

```c
// input pwd    : Real password
// input attempt: Tentative password
for (i = 0; i < min(len(pwd), len(attempt)); i++) {
    if(pwd[i] != attempt[i]){
        return false
    }
}
return true
```

pwd	A	L	M	A	S	T	Y
attempt	A	L	S	O	C		

🕒 Execution time (ET): ε ε

Context: Timing attacks over programs

```c
// input pwd    : Real password
// input attempt: Tentative password
for (i = 0; i < min(len(pwd), len(attempt)); i++) {
    if(pwd[i] != attempt[i]){
        return false
    }
}
return true
```

pwd	A	L	M	A	S	T	Y
attempt	A	L	S	O	C		

🕒 Execution time (ET): ε ε ε

Context: Timing attacks over programs

```c
// input pwd    : Real password
// input attempt: Tentative password
for (i = 0; i < min(len(pwd), len(attempt)); i++) {
    if(pwd[i] != attempt[i]){
        return false
    }
}
return true
```

pwd	A	L	M	A	S	T	Y
attempt	A	L	S	O	C		

🕓 Execution time (ET): $\varepsilon \quad \varepsilon \quad \varepsilon \qquad = 3\varepsilon \quad \Rightarrow$ 2 correct characters

Problem: The ET is proportional to the **number of consecutive correct** characters from the beginning of attempt

Context: The use of formal methods

Need to detect **timing-leak** vulnerabilities

- We want **formal guarantees** → formal methods
  - Various methods:
    - Abstract interpretation
    - Static analysis
    - Model checking
    - Theorem proving

Context: The use of formal methods

Need to detect **timing-leak** vulnerabilities

- We want **formal guarantees** → formal methods
  - Various methods:
    - Abstract interpretation
    - Static analysis
    - **Model checking**
    - Theorem proving

Context: Model checking overview

System

Specification

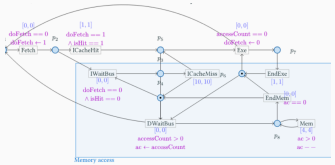
"The system must be safe"

Context: Model checking overview

System

→

Formal model



Specification

"The system must be safe"

→

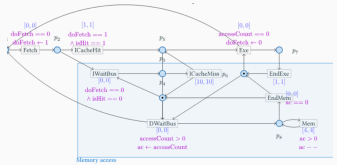
Property

$AG \quad \neg$ ○

Context: Model checking overview

System

Formal model



Specification

"The system must be safe"

Property

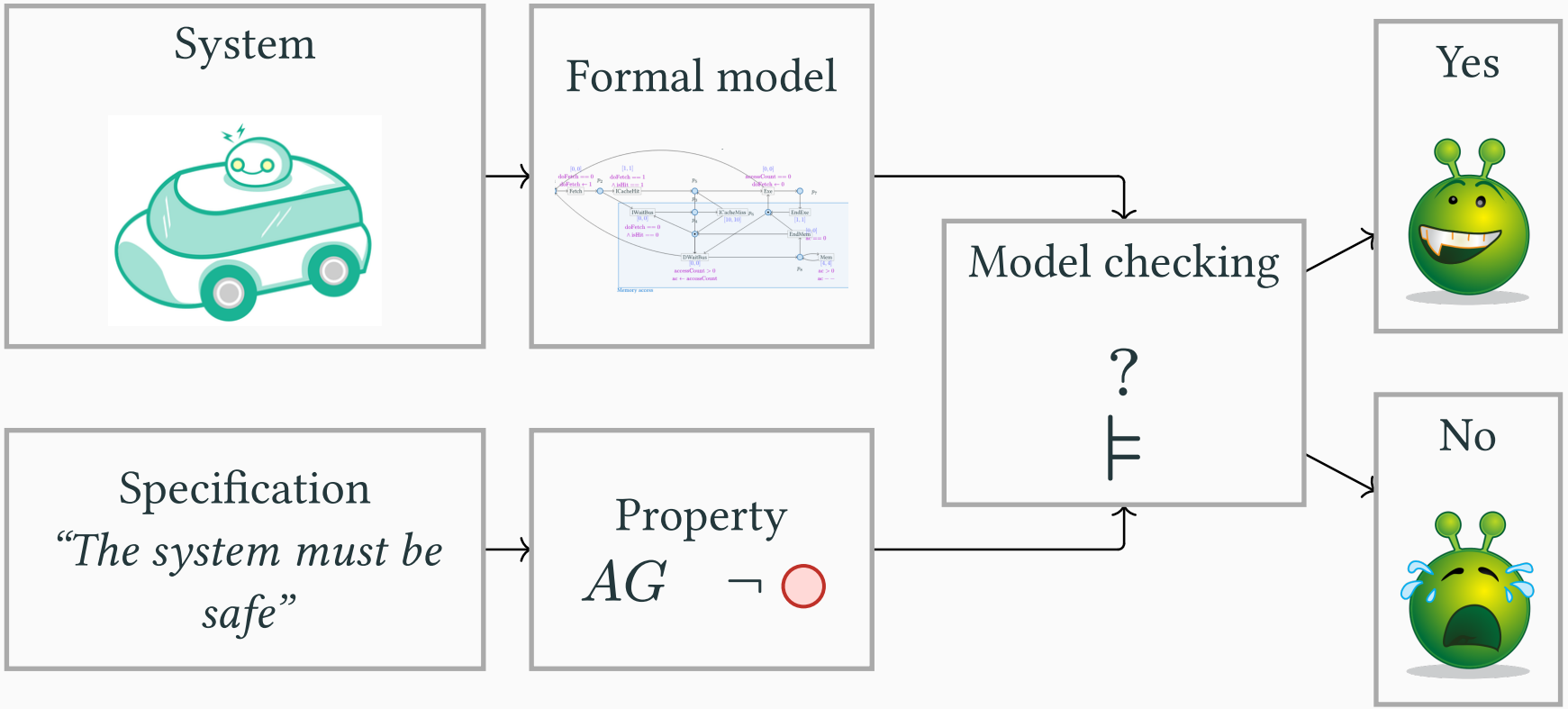
$AG \quad \neg$ 

Model checking

?

$\models$

Context: Model checking overview



Question: does the model of the system **satisfy** the property?

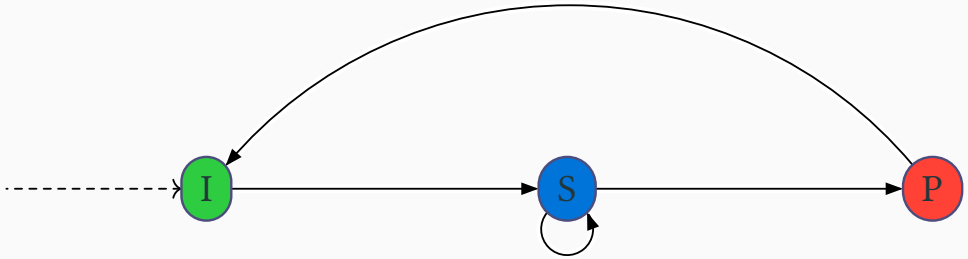
Timed automaton (TA)

- Finite-state automaton (sets of locations,)

[AD94] Rajeev Alur and David L. Dill, "A Theory of Timed Automata," *Theoretical Computer Science*, 1994.

Timed automaton (TA)

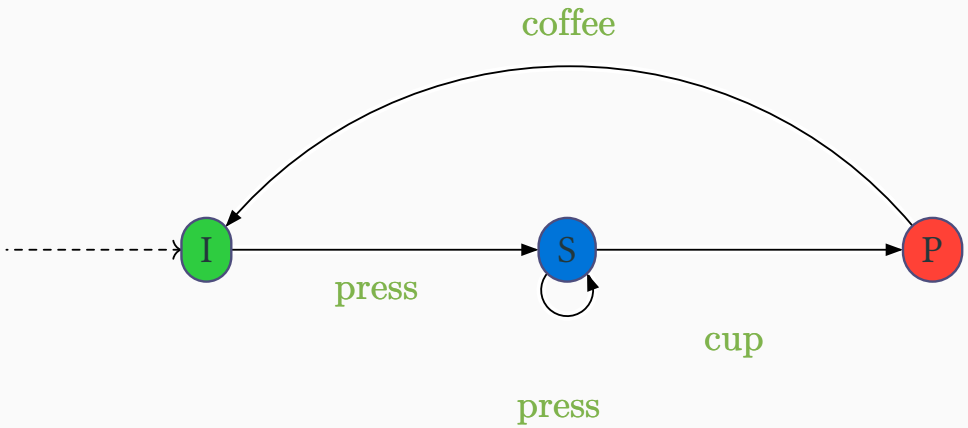
- Finite-state automaton (sets of locations, transitions,)



[AD94] Rajeev Alur and David L. Dill, “A Theory of Timed Automata,” *Theoretical Computer Science*, 1994.

Timed automaton (TA)

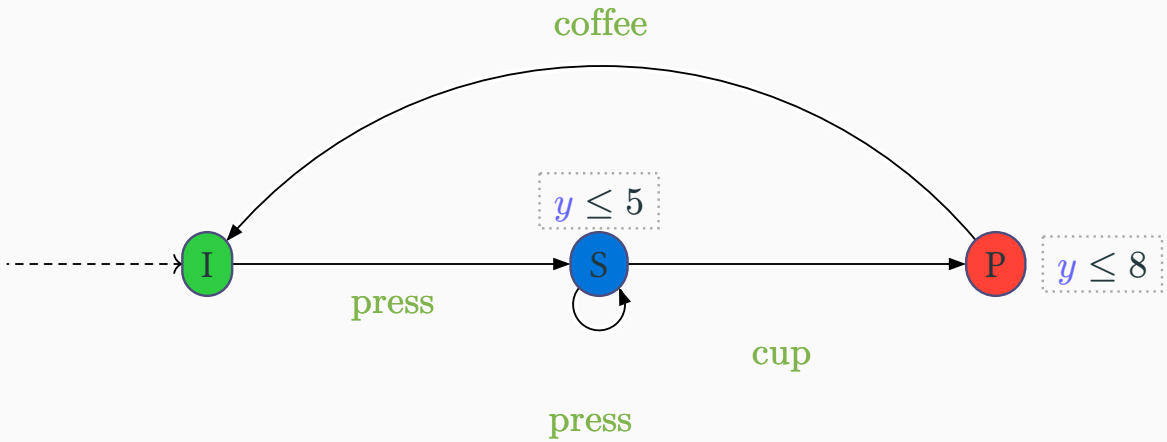
- Finite-state automaton (sets of locations, transitions, and actions)



[AD94] Rajeev Alur and David L. Dill, "A Theory of Timed Automata," *Theoretical Computer Science*, 1994.

Timed automaton (TA)

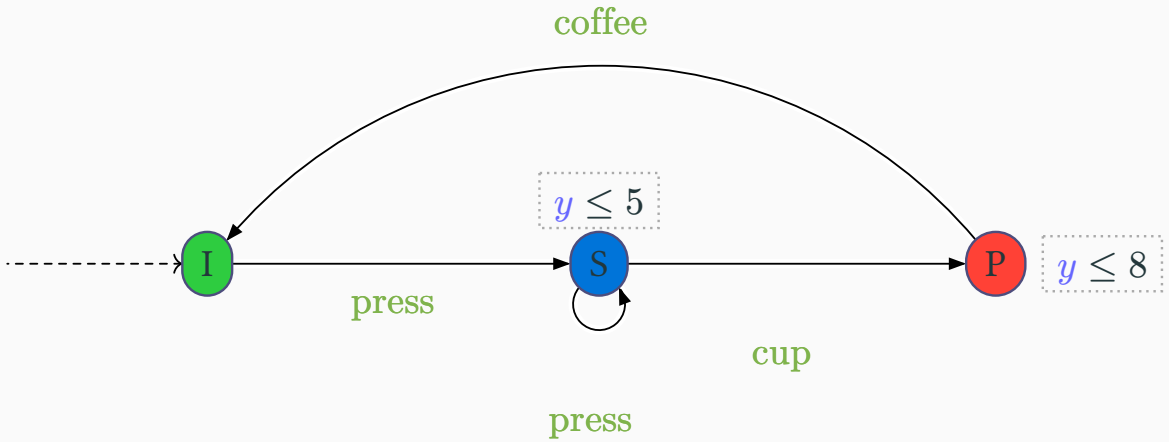
- Finite-state automaton (sets of locations, transitions, and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly **at the same rate**
- Features:



[AD94] Rajeev Alur and David L. Dill, "A Theory of Timed Automata," *Theoretical Computer Science*, 1994.

Timed automaton (TA)

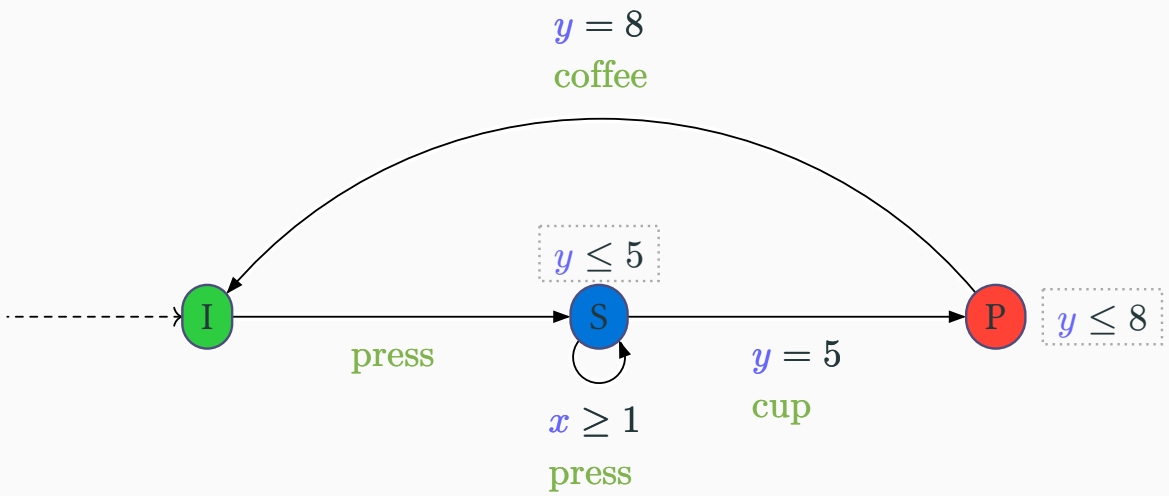
- Finite-state automaton (sets of locations, transitions, and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly **at the same rate**
- Features:
  - Location **invariant**: property to be verified to stay at a location



[AD94] Rajeev Alur and David L. Dill, "A Theory of Timed Automata," *Theoretical Computer Science*, 1994.

Timed automaton (TA)

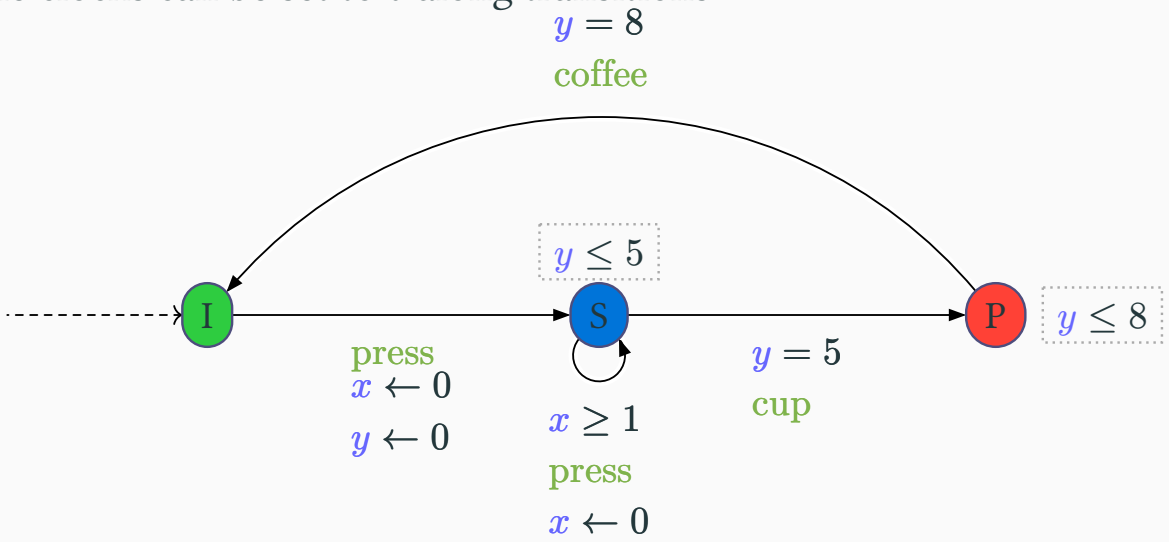
- Finite-state automaton (sets of locations, transitions, and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly **at the same rate**
- Features:
  - Location **invariant**: property to be verified to stay at a location
  - Transition **guard**: property to be verified to enable a transition



[AD94] Rajeev Alur and David L. Dill, "A Theory of Timed Automata," *Theoretical Computer Science*, 1994.

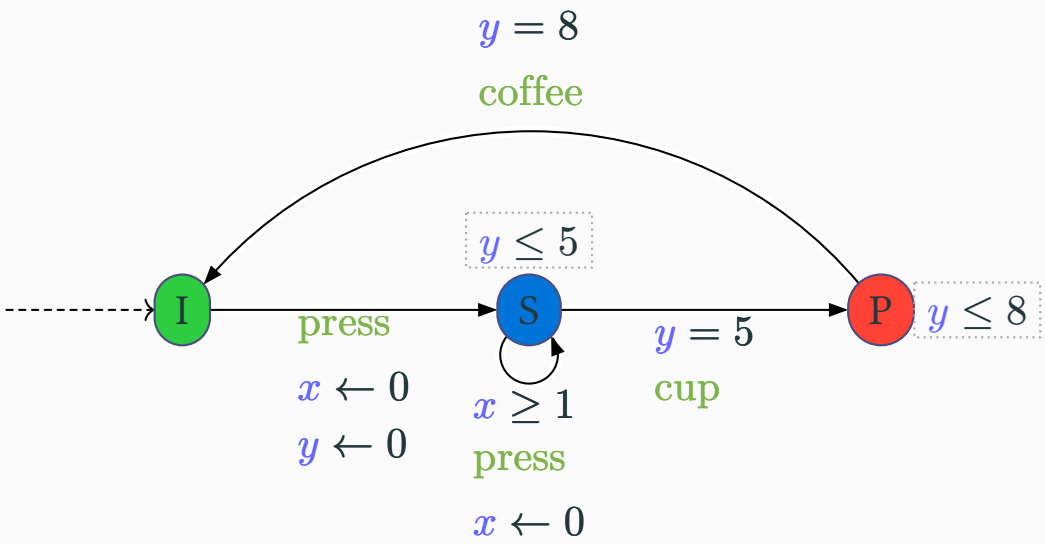
Timed automaton (TA)

- Finite-state automaton (sets of locations, transitions, and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly **at the same rate**
- Features:
  - Location **invariant**: property to be verified to stay at a location
  - Transition **guard**: property to be verified to enable a transition
  - Clock **reset**: some of the clocks can be set to 0 along transitions



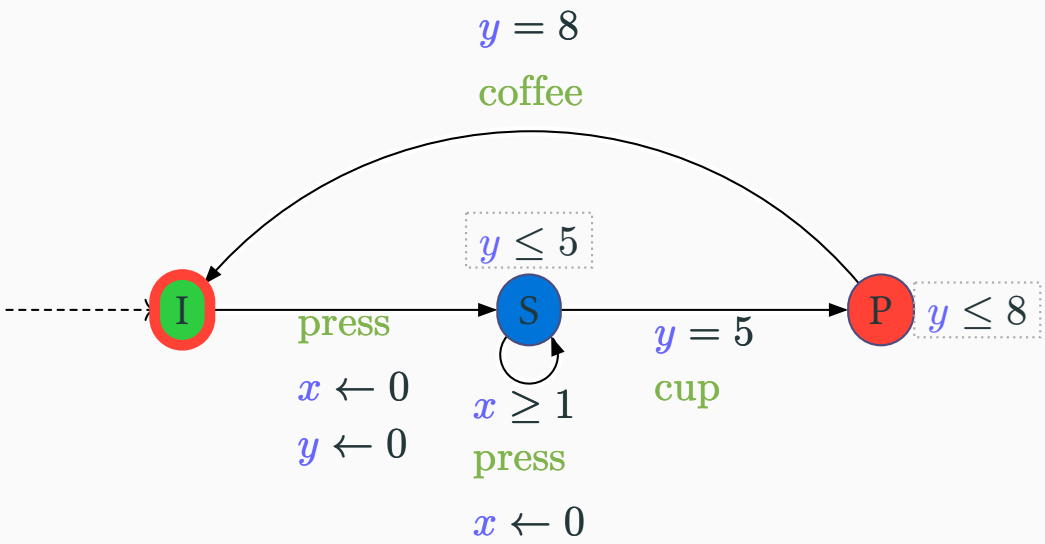
[AD94] Rajeev Alur and David L. Dill, "A Theory of Timed Automata," *Theoretical Computer Science*, 1994.

The most critical system: The coffee machine



- Coffee with two doses of sugar

The most critical system: The coffee machine



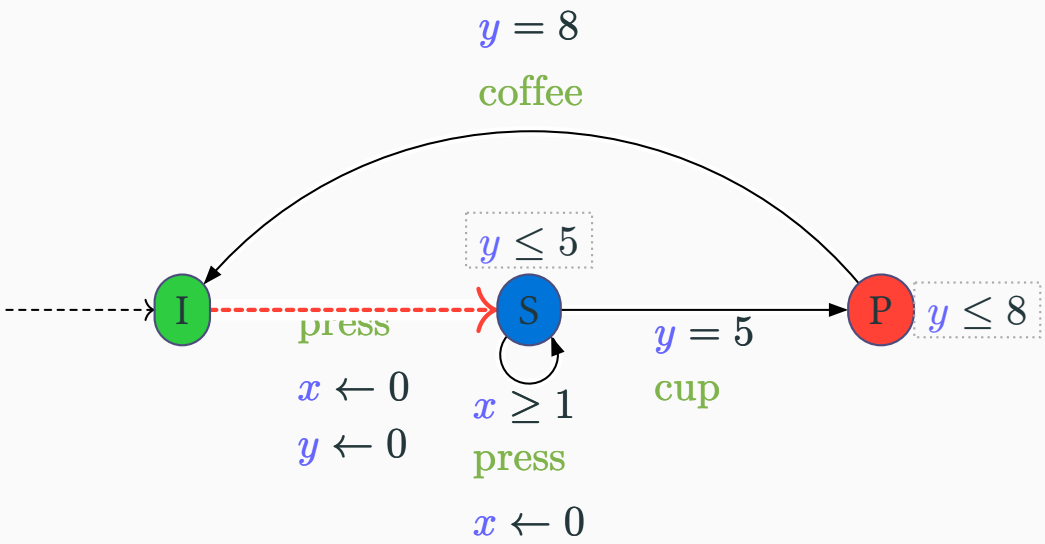
- Coffee with two doses of sugar

I

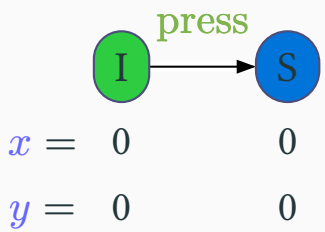
$x = 0$

$y = 0$

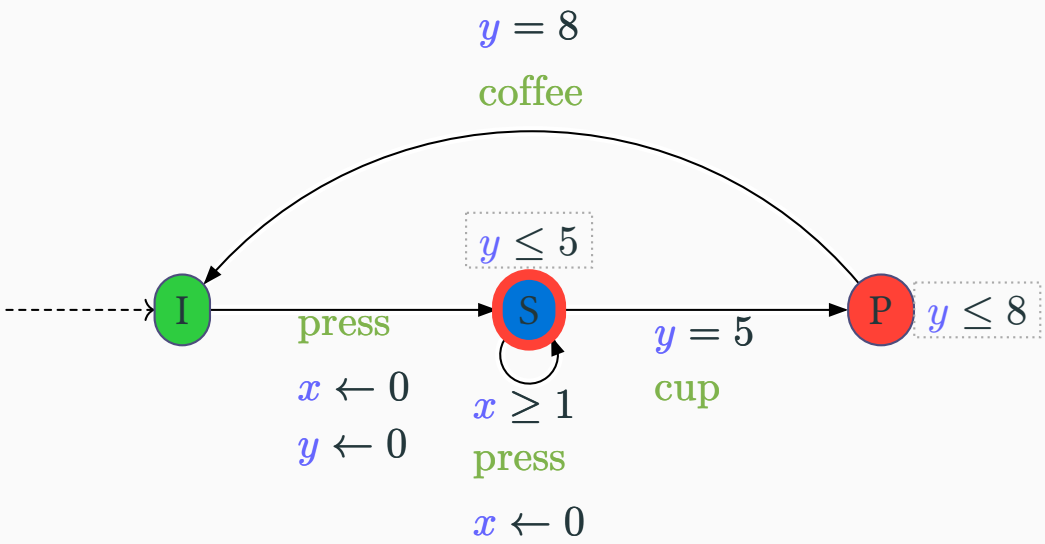
The most critical system: The coffee machine



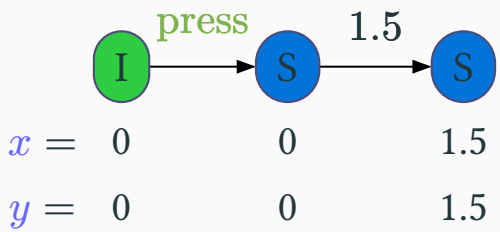
- Coffee with two doses of sugar



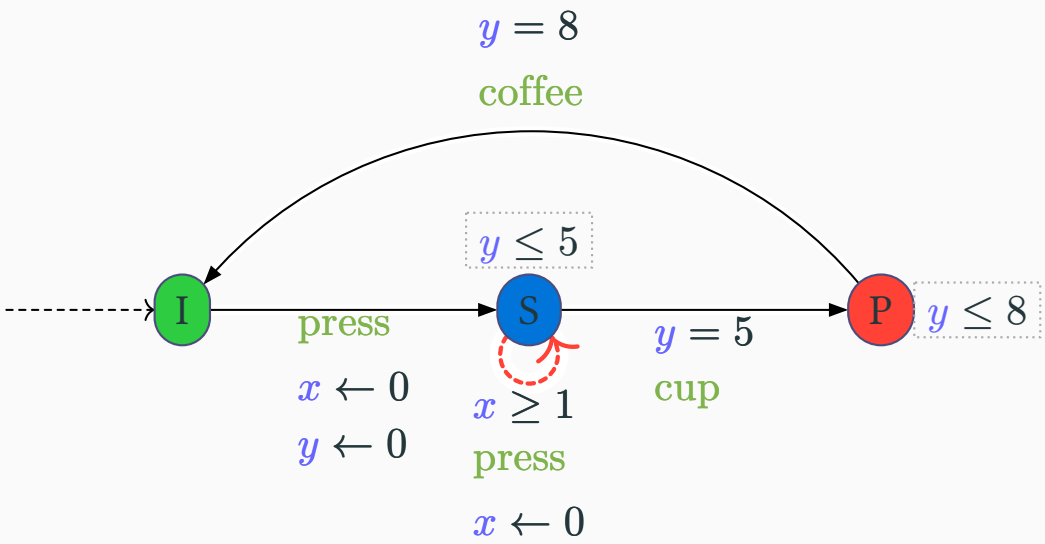
The most critical system: The coffee machine



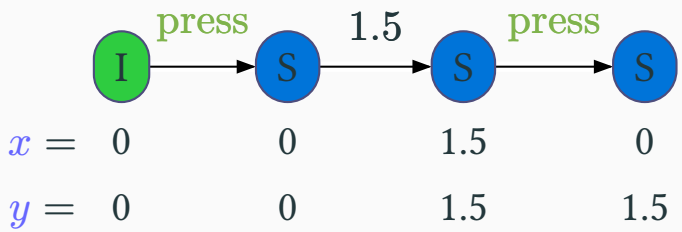
- Coffee with two doses of sugar



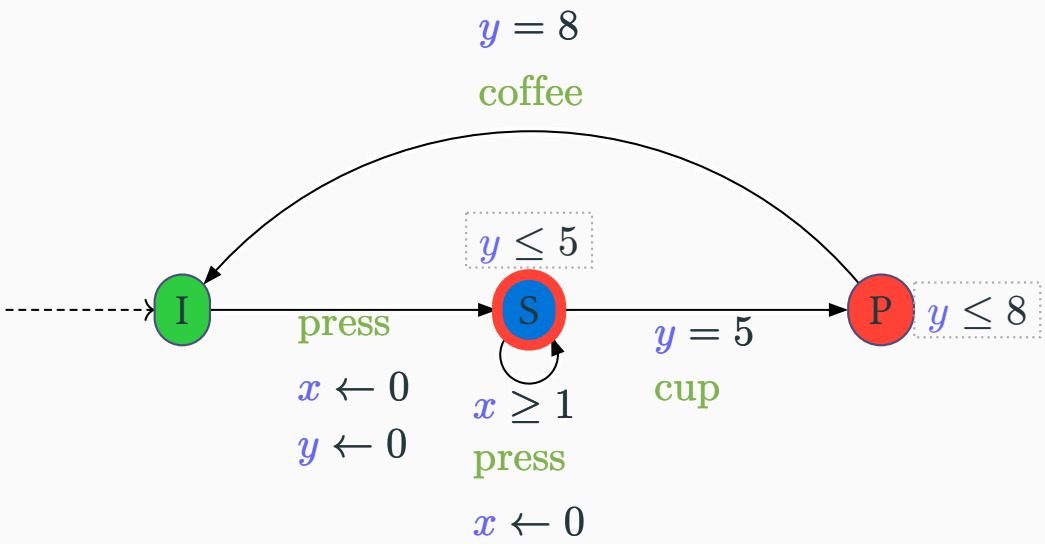
The most critical system: The coffee machine



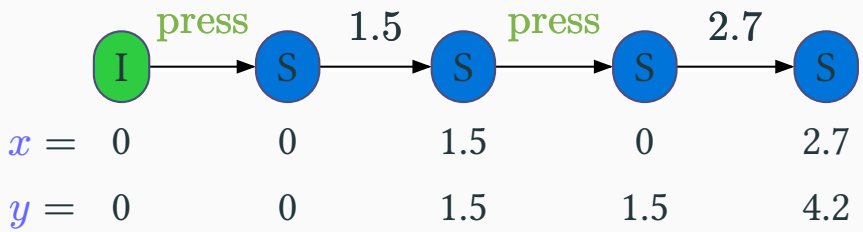
- Coffee with two doses of sugar



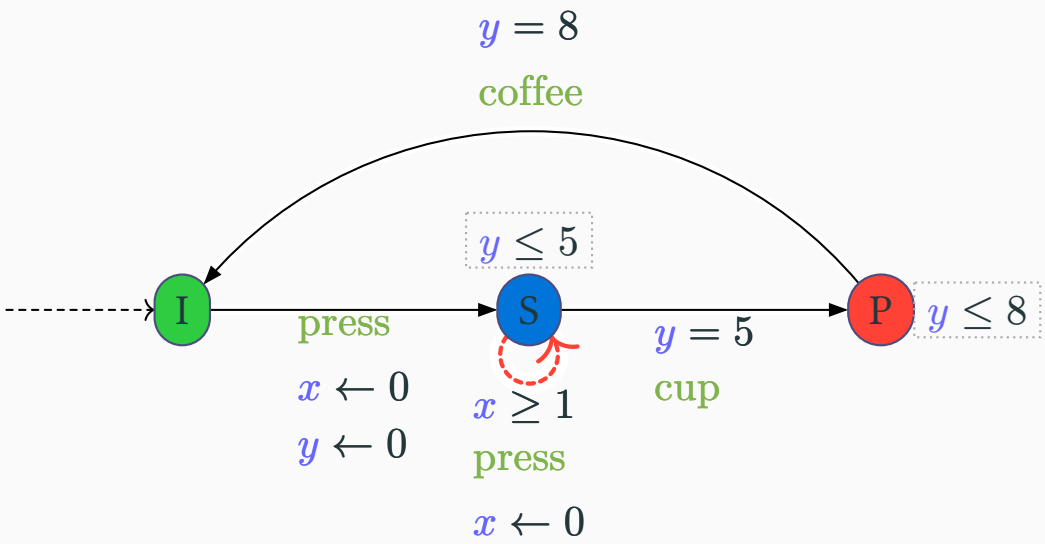
The most critical system: The coffee machine



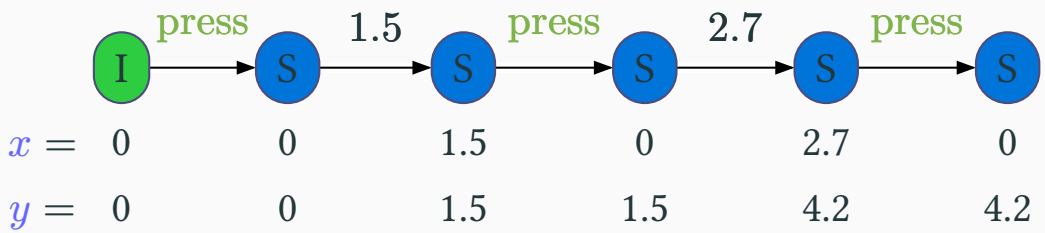
- Coffee with two doses of sugar



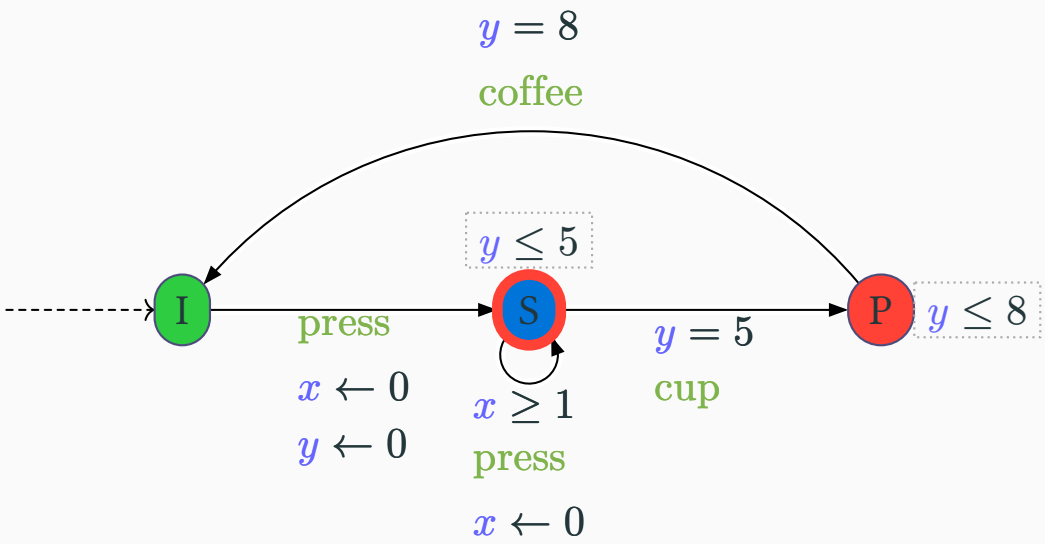
The most critical system: The coffee machine



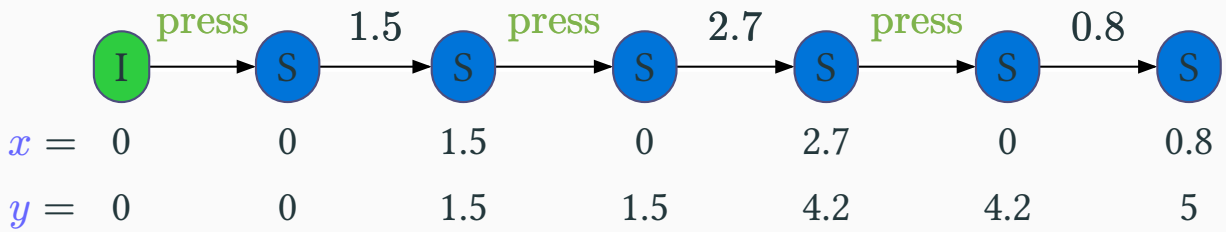
- Coffee with two doses of sugar



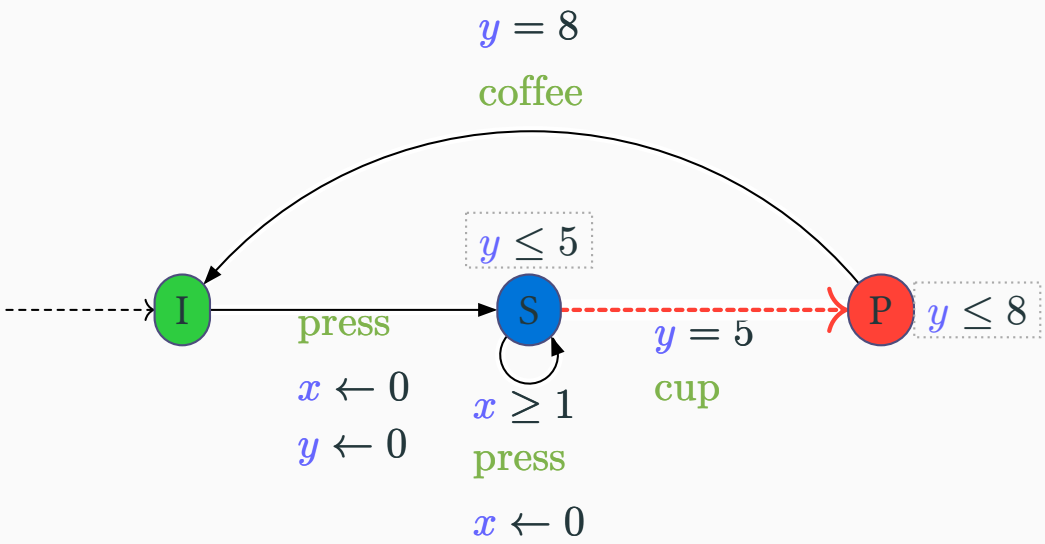
The most critical system: The coffee machine



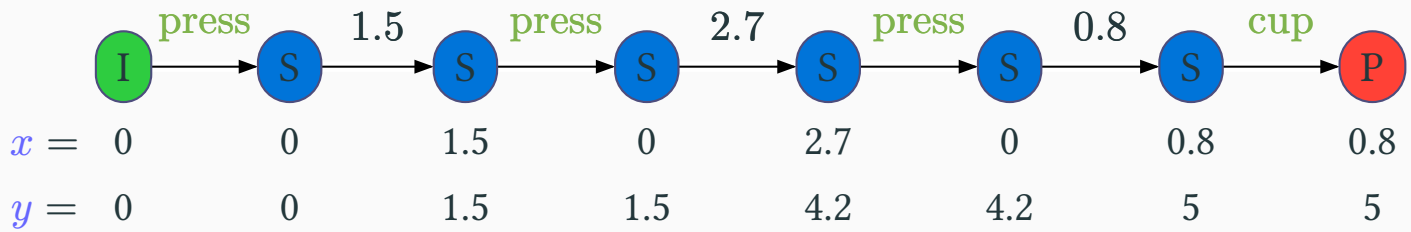
- Coffee with two doses of sugar



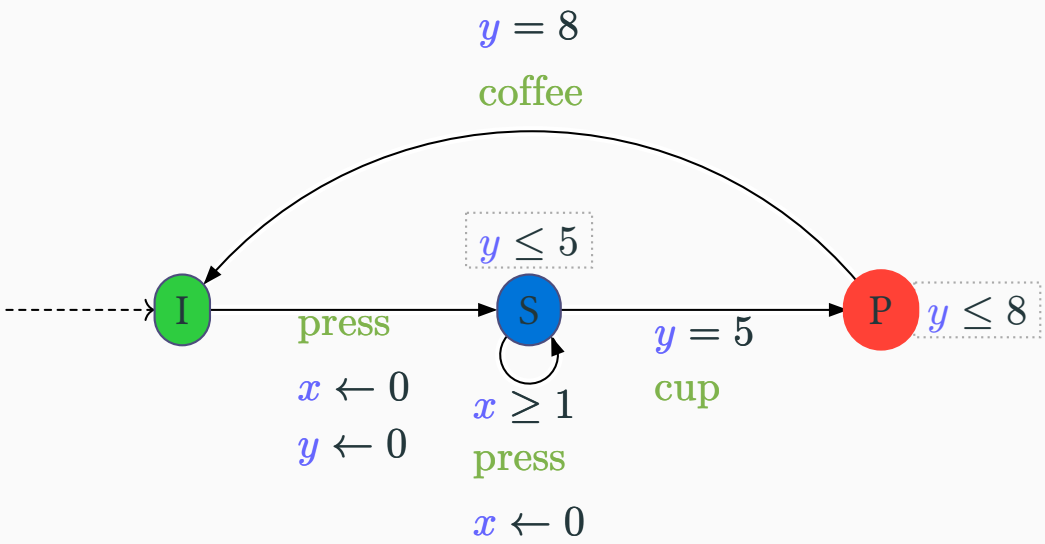
The most critical system: The coffee machine



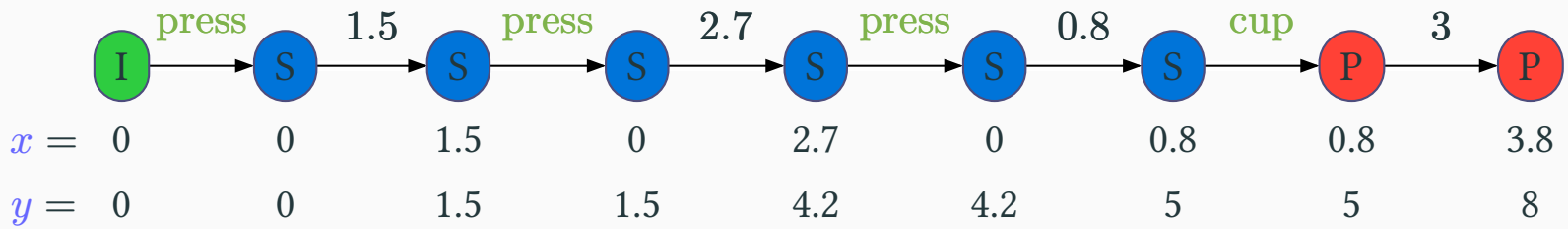
- Coffee with two doses of sugar



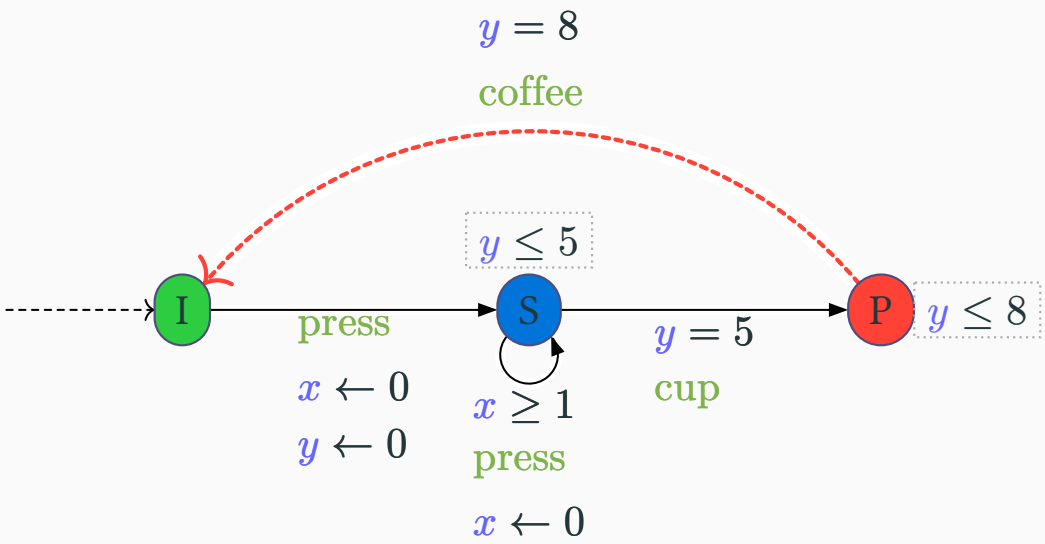
The most critical system: The coffee machine



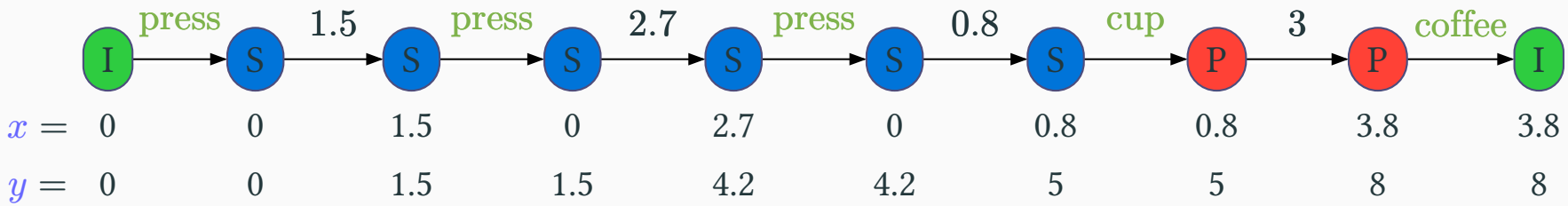
- Coffee with two doses of sugar



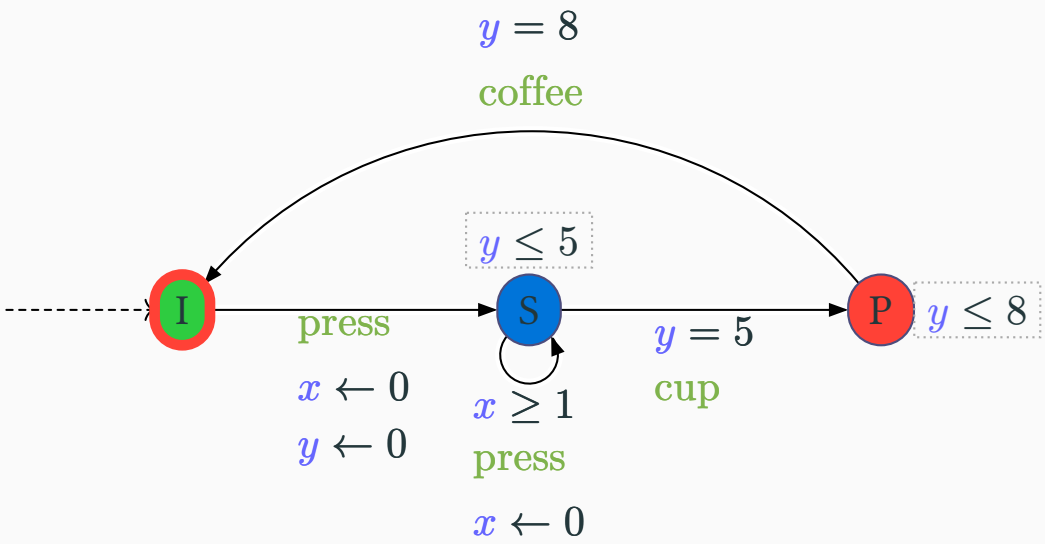
The most critical system: The coffee machine



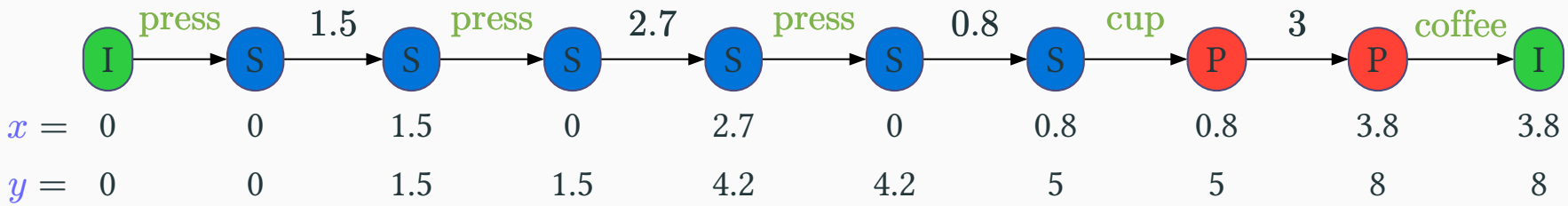
- Coffee with two doses of sugar



The most critical system: The coffee machine



- Coffee with two doses of sugar



Timed opacity

The attacker model

Attacker capabilities

- Has access to the model (white box)
- Can observe an execution

The attacker model

Attacker capabilities

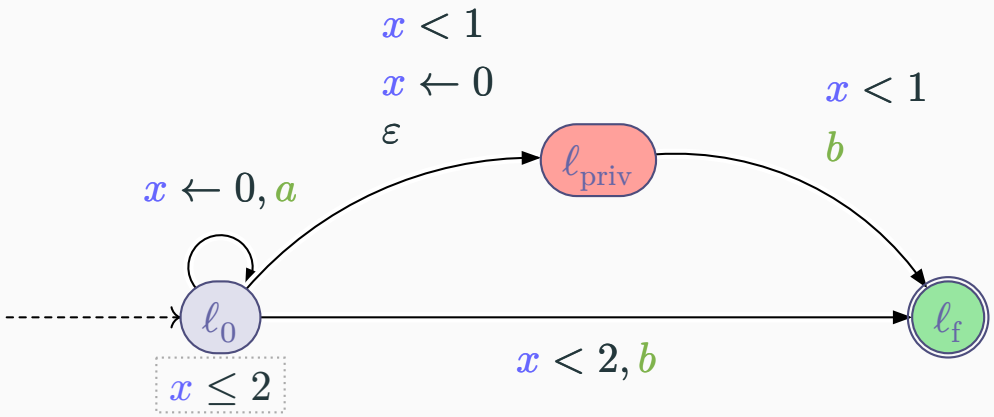
- Has access to the model (white box)
- Can observe an execution

Attacker goal

- Deduce secret information from these observations

  → *visit of a private location*

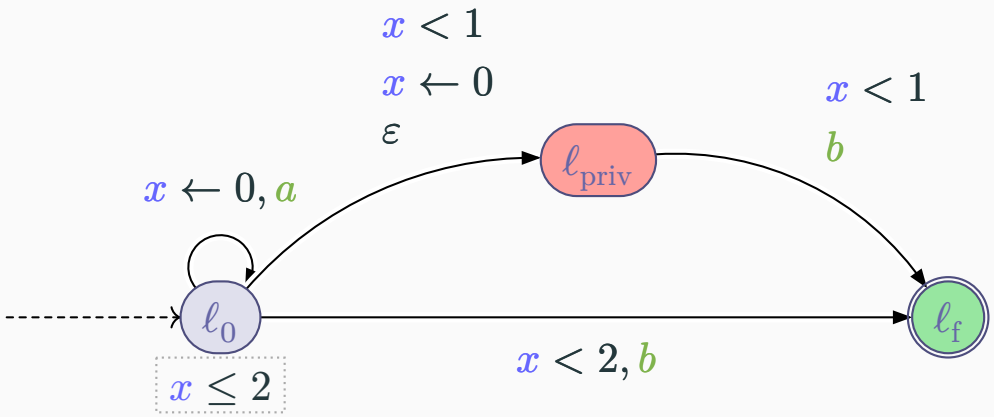
Attacker setting



Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

Attacker setting

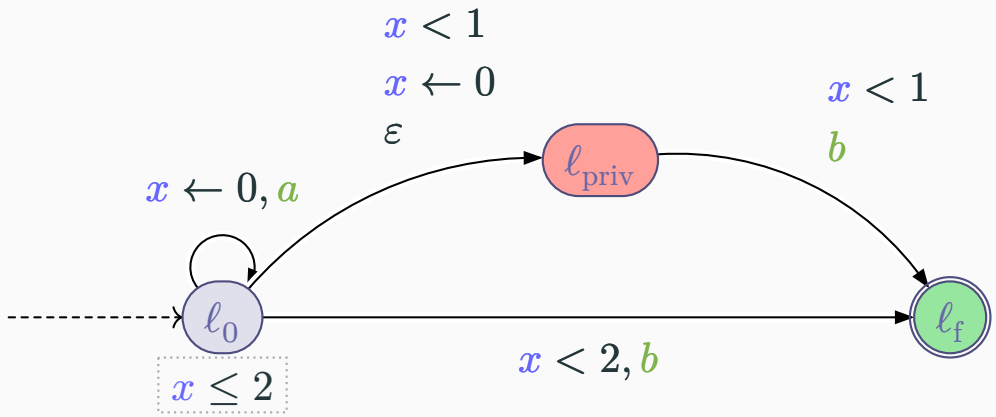


Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.

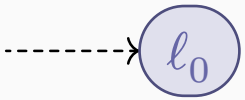
Attacker setting



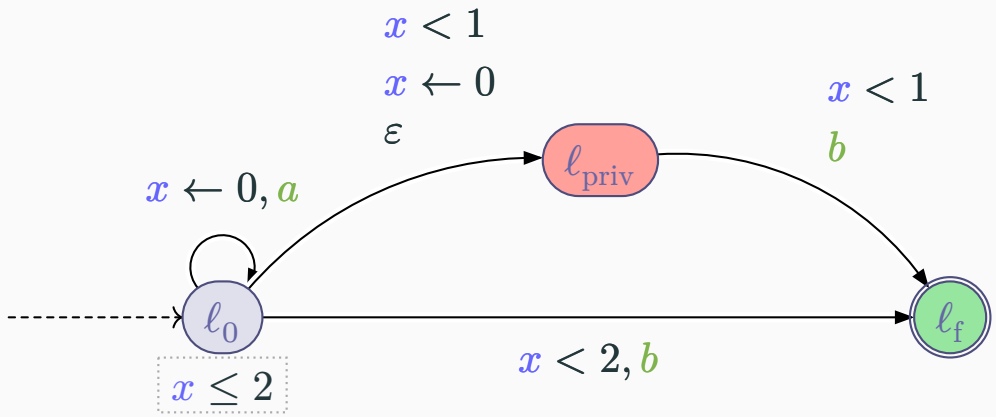
Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



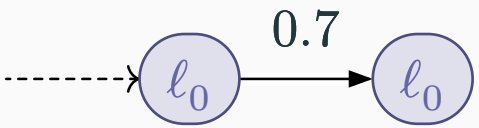
Attacker setting



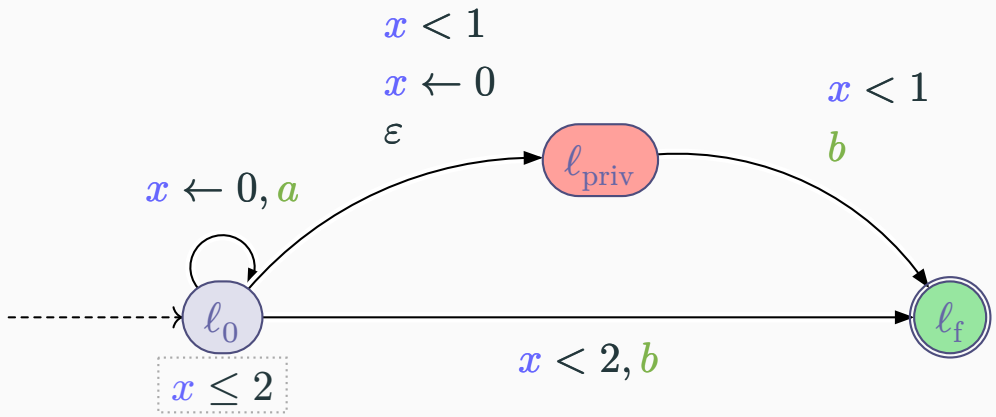
Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



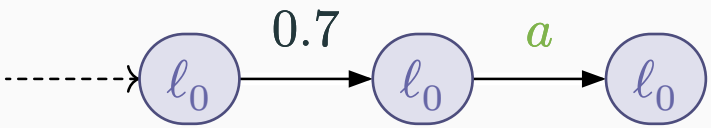
Attacker setting



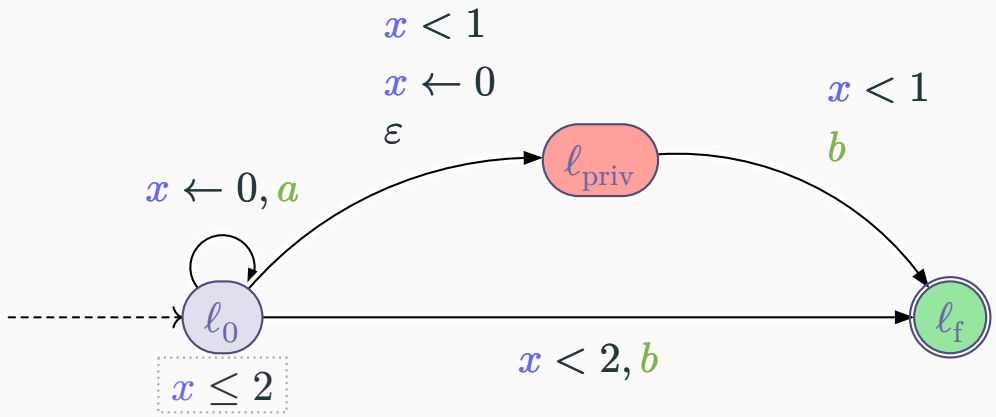
Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



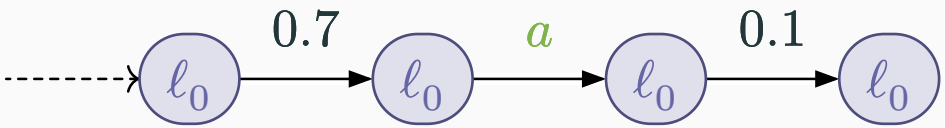
Attacker setting



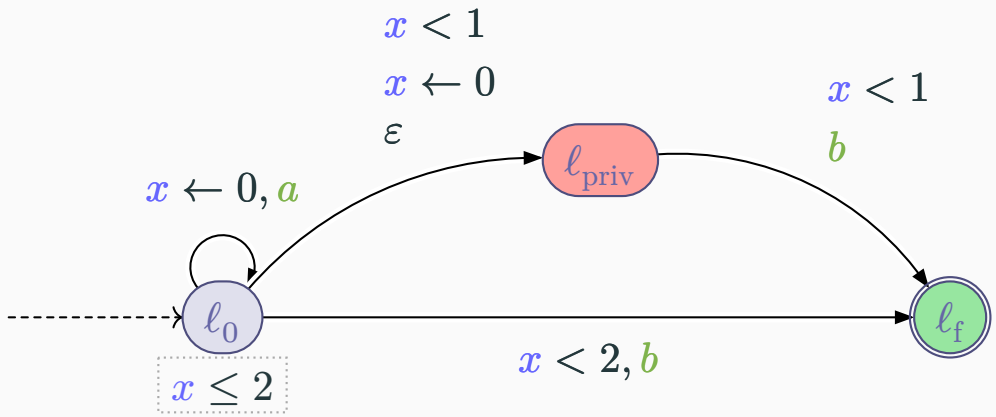
Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



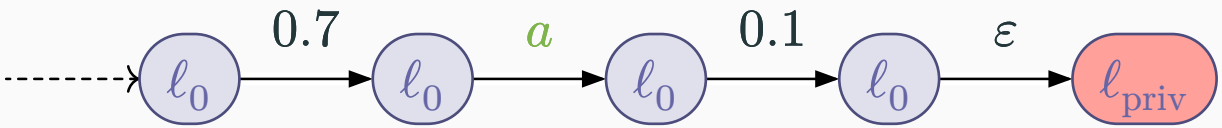
Attacker setting



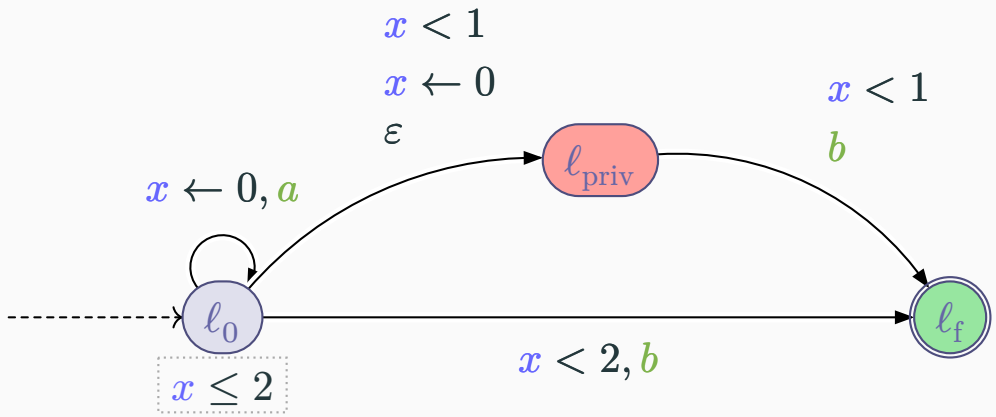
Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



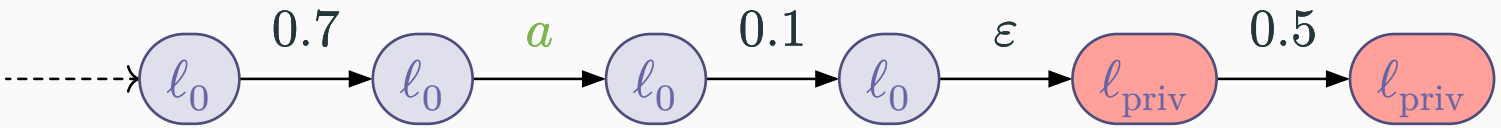
Attacker setting



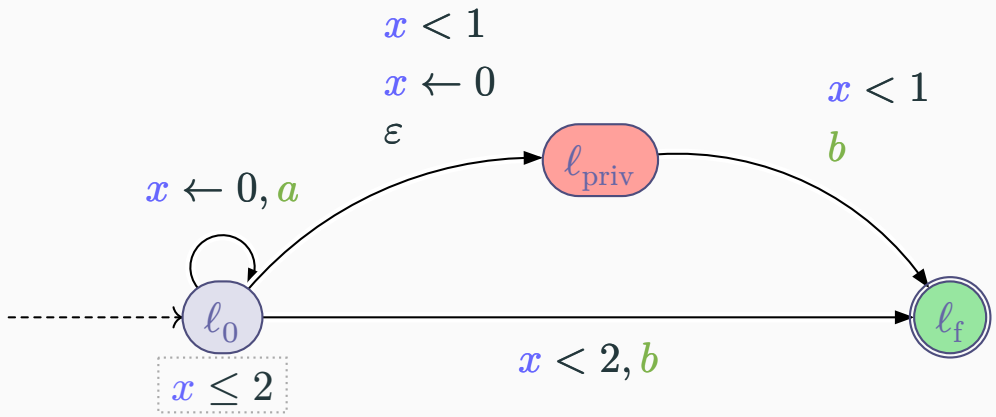
Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



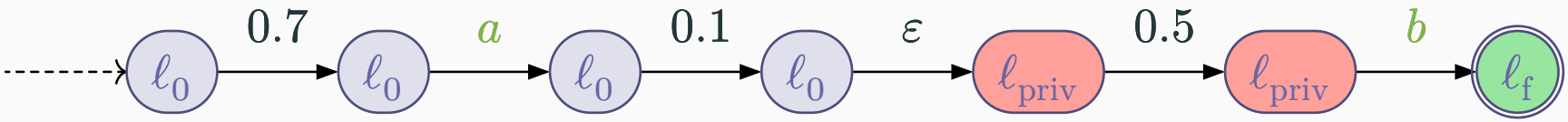
Attacker setting



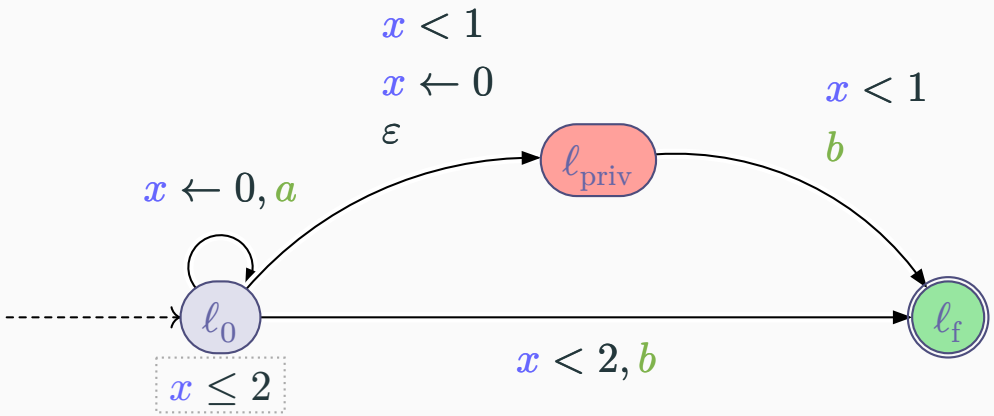
Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



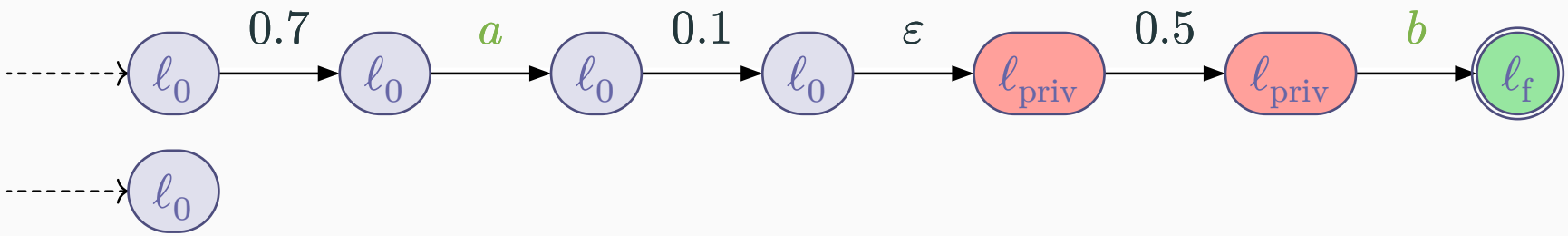
Attacker setting



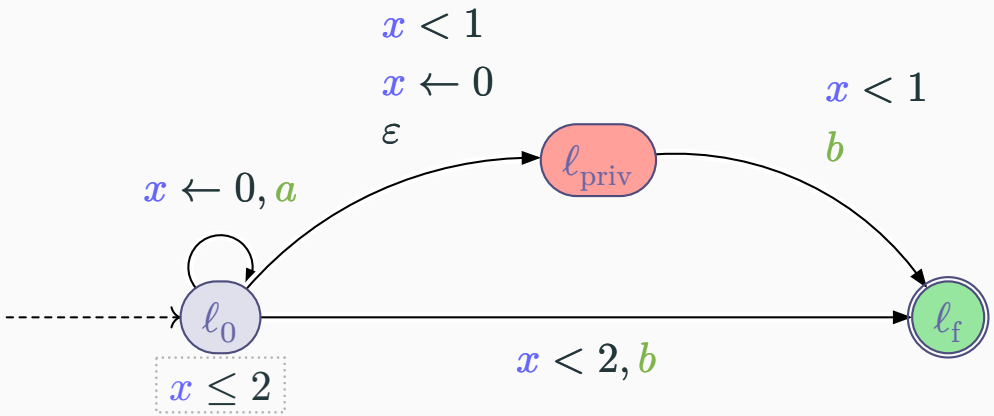
Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



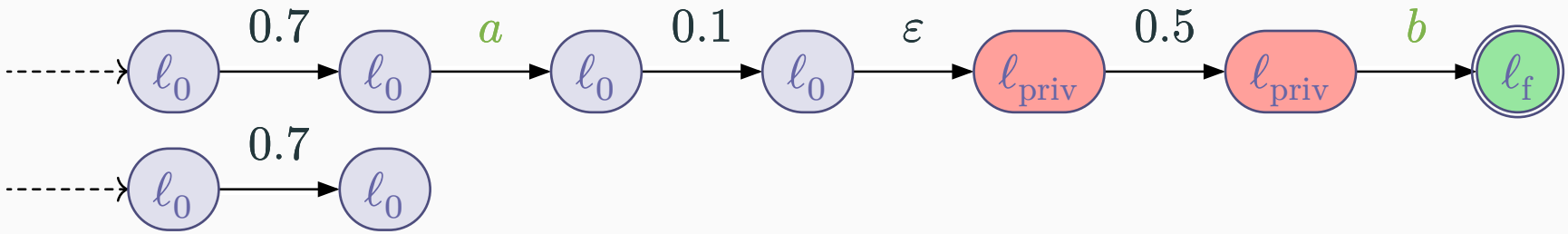
Attacker setting



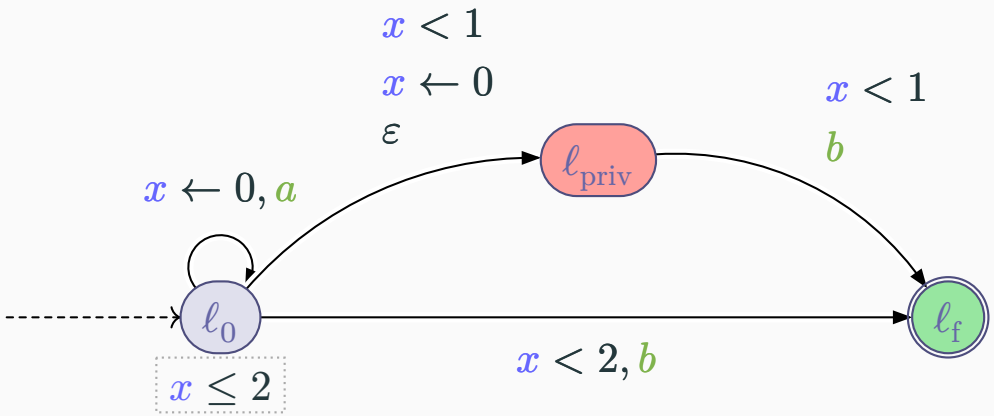
Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



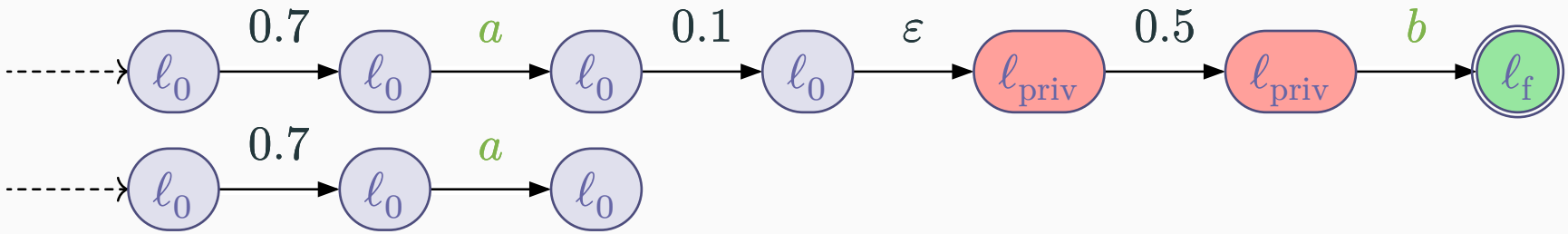
Attacker setting



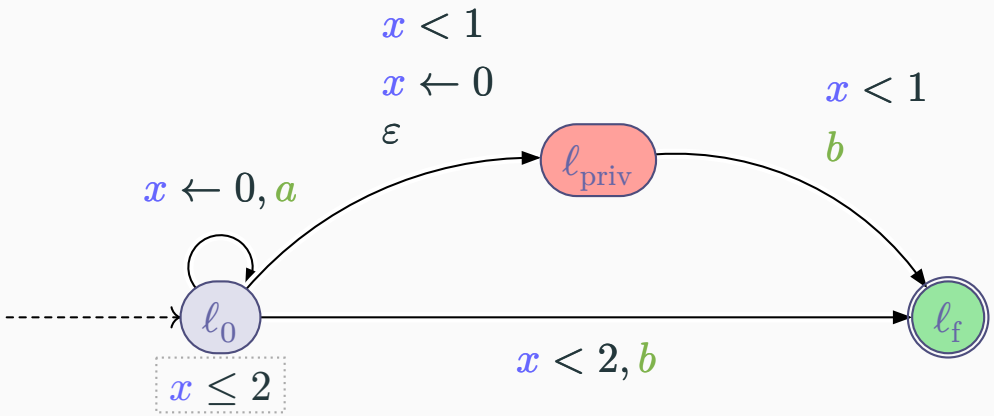
Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



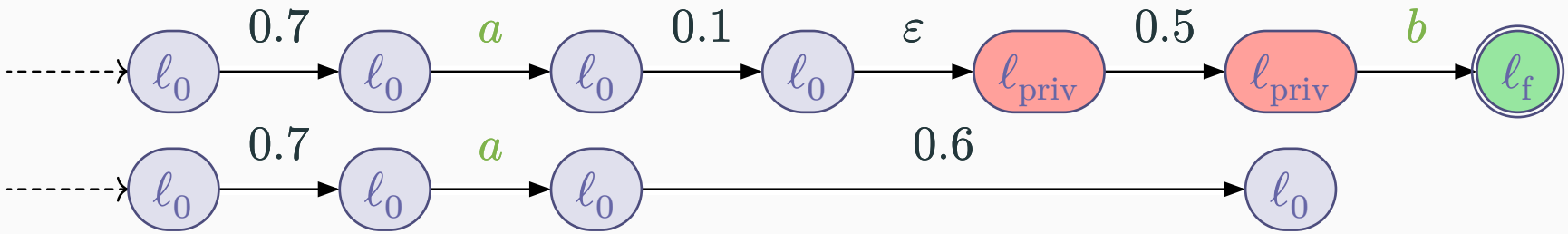
Attacker setting



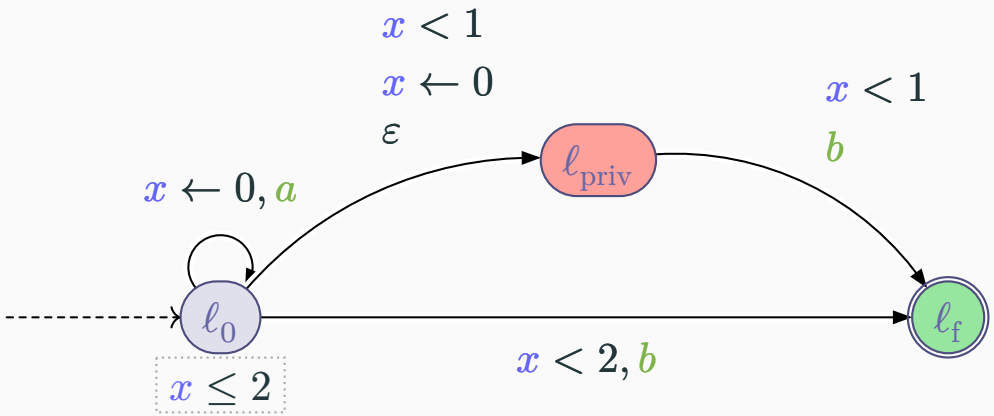
Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



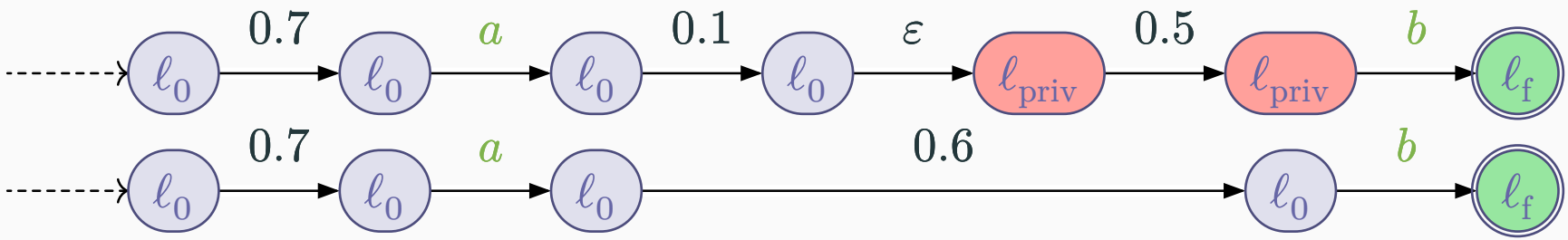
Attacker setting



Observed trace:
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



Opacity in timed automata

Definition 1 (Opacity)

A TA is **opaque** iff all *observable* traces can be obtained **both**

- by runs visiting ℓ_{priv}
- by runs **not** visiting ℓ_{priv}

♣

[Cas09] Franck Cassez, "The Dark Side of Timed Opacity," in *ISA*, 2009.

Opacity in timed automata

Definition 1 (Opacity)

A TA is **opaque** iff all *observable* traces can be obtained **both**

- by runs visiting ℓ_{priv}
- by runs **not** visiting ℓ_{priv}

♣

Opacity decision problem

Is the given timed automaton opaque?

[Cas09] Franck Cassez, "The Dark Side of Timed Opacity," in *ISA*, 2009.

Opacity in timed automata

Definition 1 (Opacity)

A TA is **opaque** iff all *observable* traces can be obtained **both**

- by runs visiting ℓ_{priv}
- by runs **not** visiting ℓ_{priv}

♣

Opacity decision problem

Is the given timed automaton opaque?

Franck Cassez, The Dark Side of Timed Opacity (2009)

→ **Opacity is undecidable for timed automata!**

[Cas09] Franck Cassez, "The Dark Side of Timed Opacity," in *ISA*, 2009.

Opacity in timed automata

Definition 1 (Opacity)

A TA is **opaque** iff all *observable* traces can be obtained **both**

- by runs visiting ℓ_{priv}
- by runs **not** visiting ℓ_{priv}

♣

So... is it the end?

Opacity decision pr

Is the given timed automaton opaque?

Franck Cassez, The Dark Side of Timed Opacity (2009)

→ **Opacity is undecidable for timed automata!**

[Cas09] Franck Cassez, "The Dark Side of Timed Opacity," in *ISA*, 2009.

Solutions

Proposed solutions

Change the system → Subclasses of TA

- restriction on the number of actions
- restriction on the number of clocks
- discrete time

Proposed solutions

Change the system → Subclasses of TA

- restriction on the number of actions
- restriction on the number of clocks
- discrete time

Change the problem → Weaker attackers

- bounded number of observations
- limited observation

Proposed solutions

Change the system → Subclasses of TA

- restriction on the number of actions
- restriction on the number of clocks
- discrete time

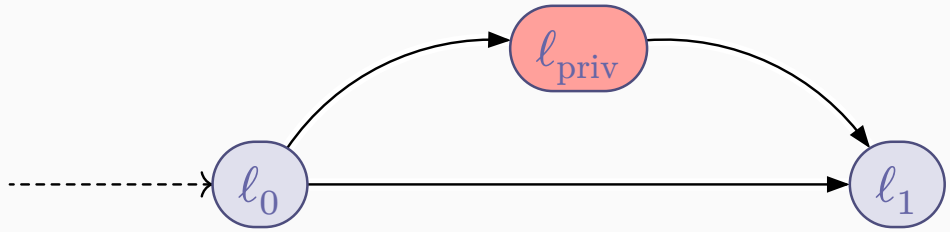
Change the problem → Weaker attackers

- bounded number of observations
- **limited observation**

Execution-time opacity

Hypothesis

- A start location and an end location
- A special private location ℓ_{priv}



Definition 1 (Execution-time opacity)

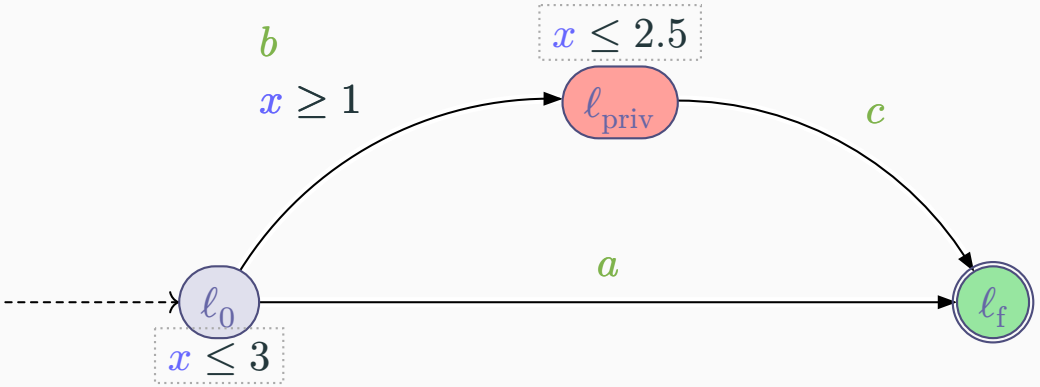
The system is **ET-opaque** for a duration d if there exist two runs of duration d

1. visiting ℓ_{priv}
2. one not visiting ℓ_{priv}

♣

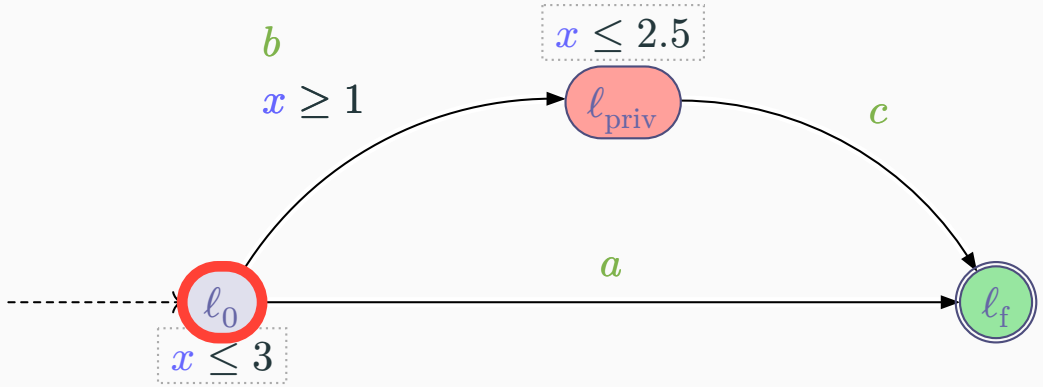
[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

Example

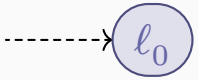


- There exist (at least) two runs of duration $d = 2$:

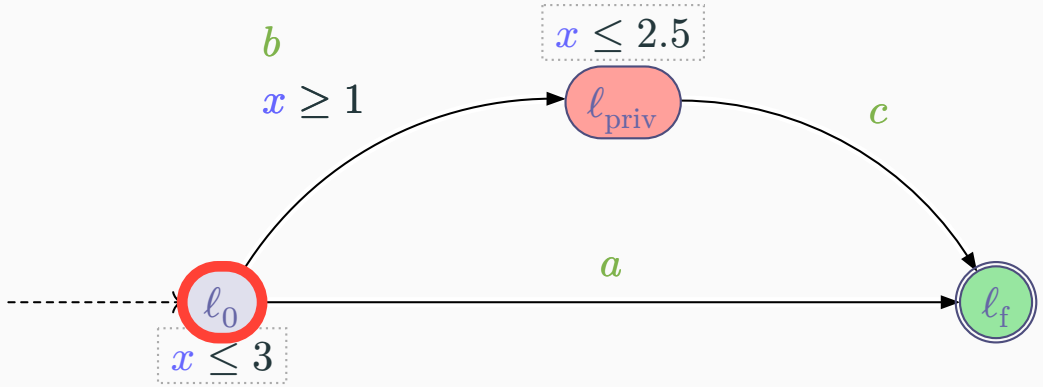
Example



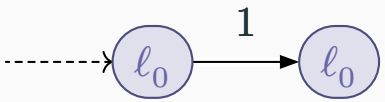
- There exist (at least) two runs of duration $d = 2$:



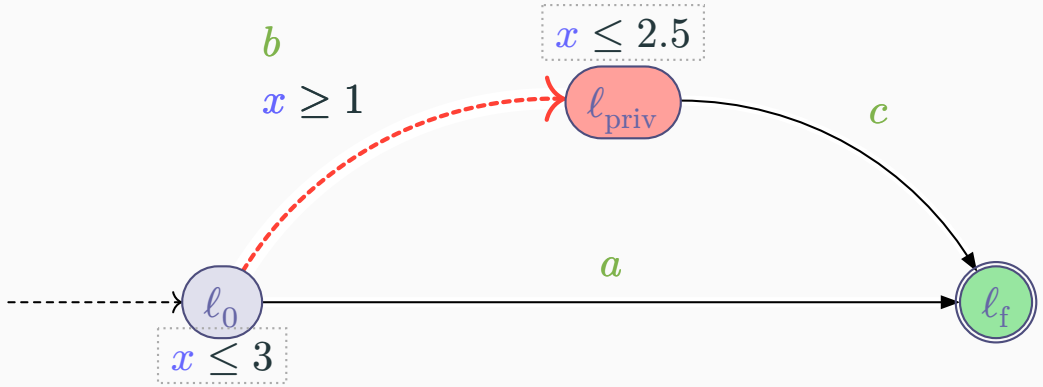
Example



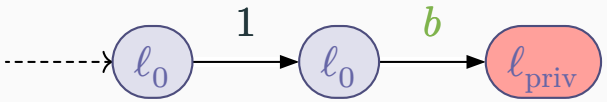
- There exist (at least) two runs of duration $d = 2$:



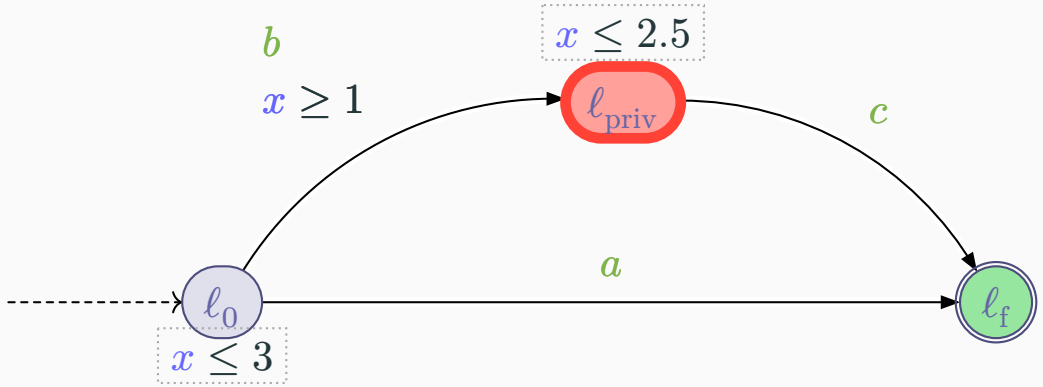
Example



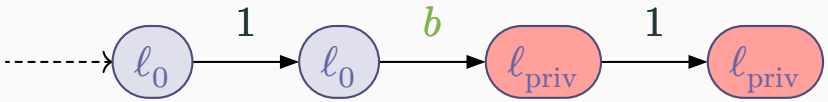
- There exist (at least) two runs of duration $d = 2$:



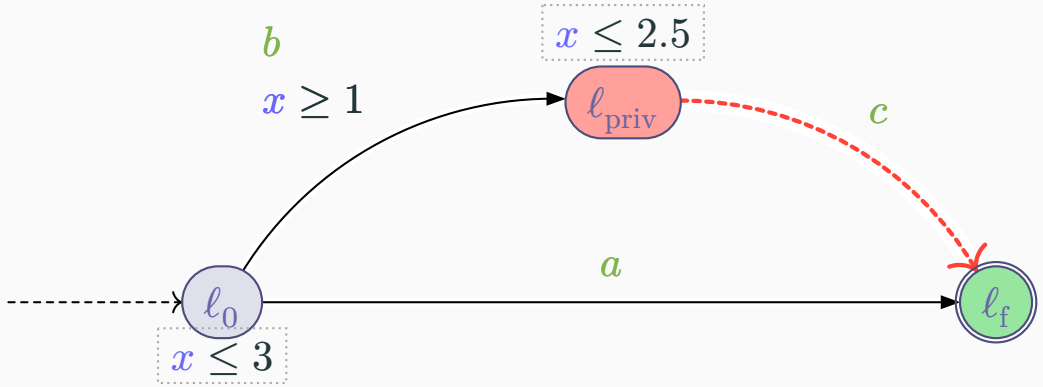
Example



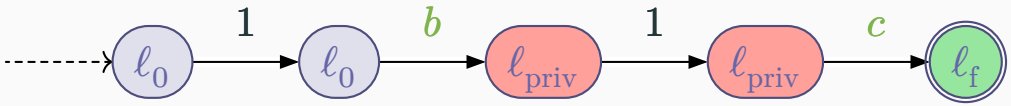
- There exist (at least) two runs of duration $d = 2$:



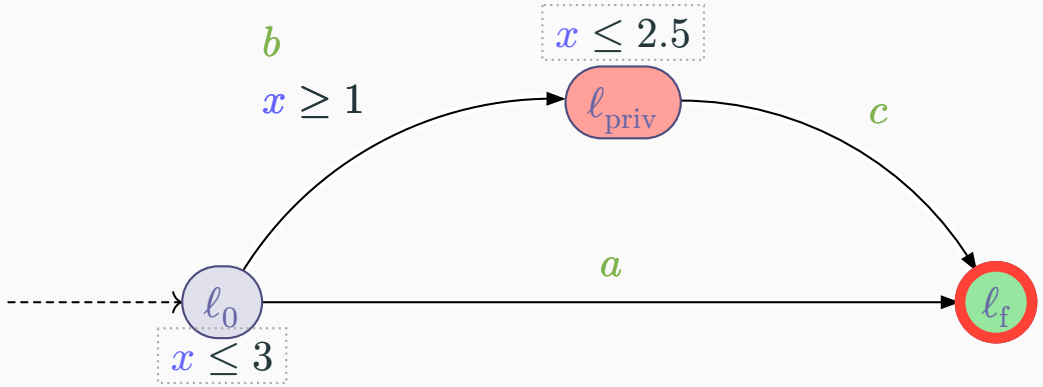
Example



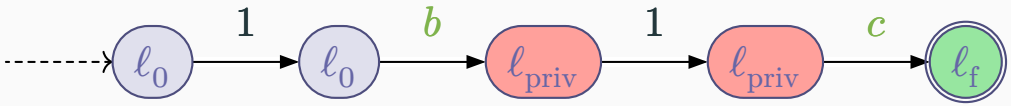
- There exist (at least) two runs of duration $d = 2$:



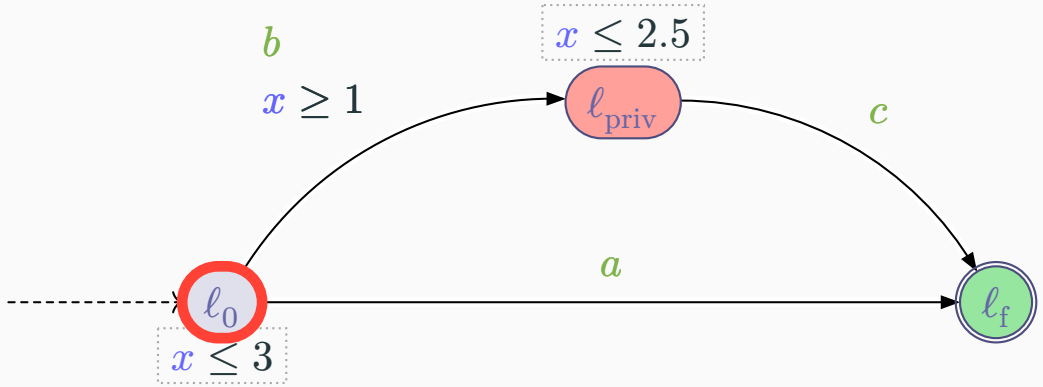
Example



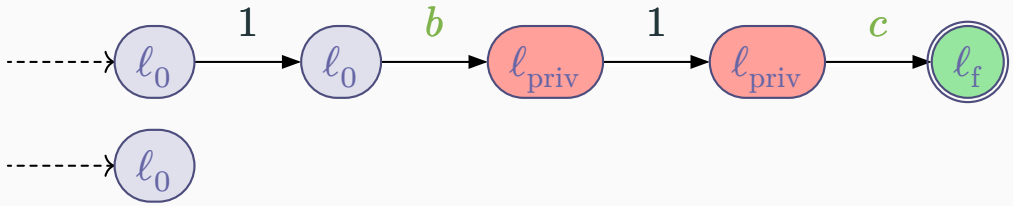
- There exist (at least) two runs of duration $d = 2$:



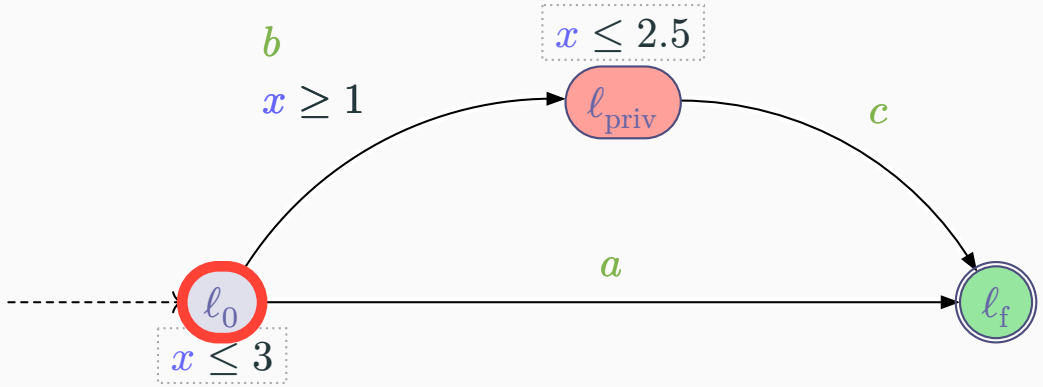
Example



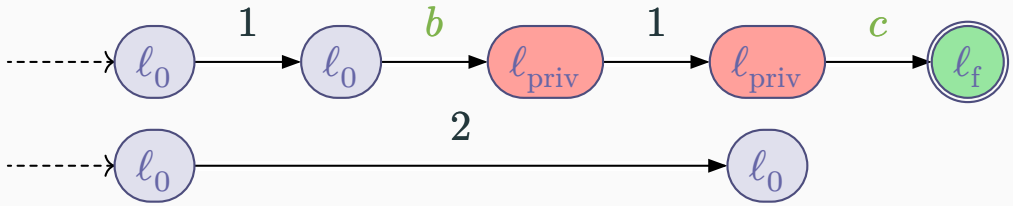
- There exist (at least) two runs of duration $d = 2$:



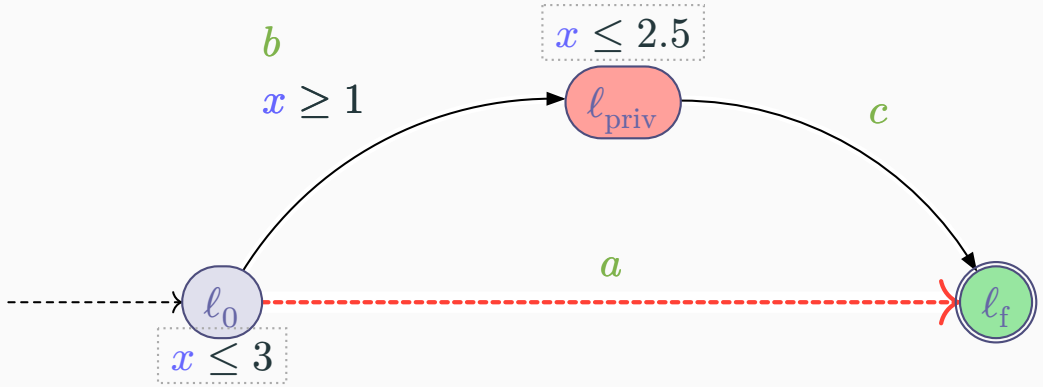
Example



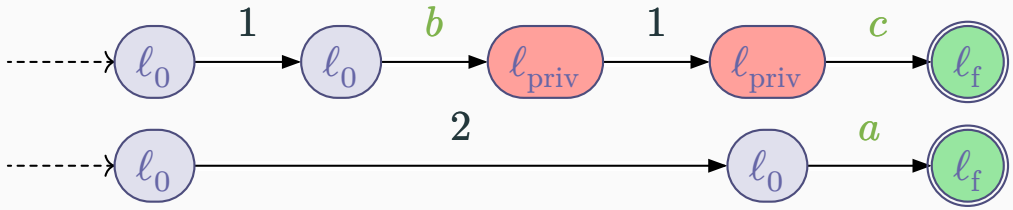
- There exist (at least) two runs of duration $d = 2$:



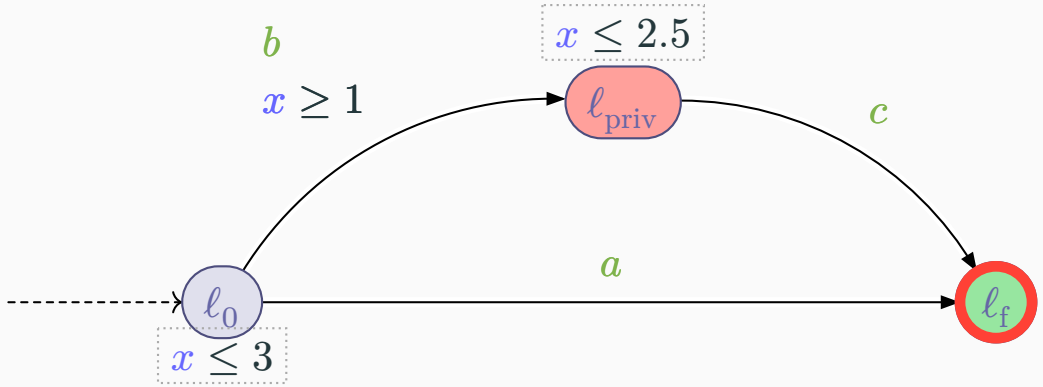
Example



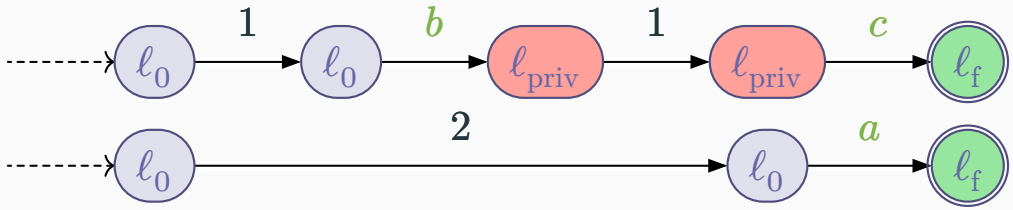
- There exist (at least) two runs of duration $d = 2$:



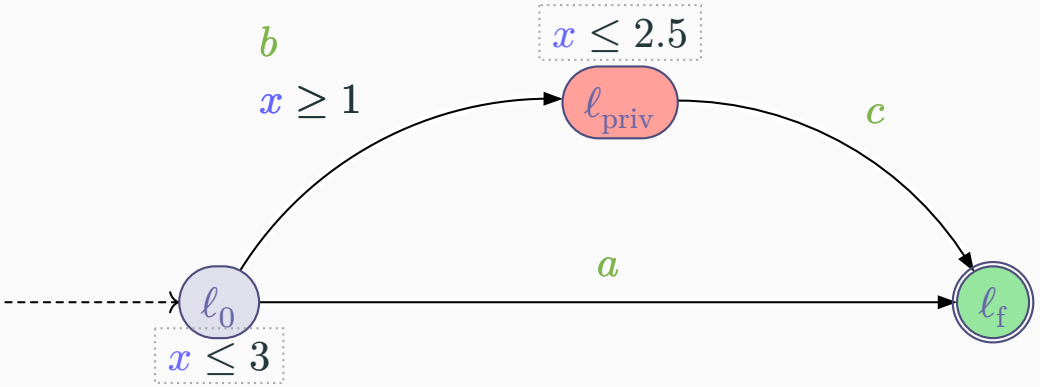
Example



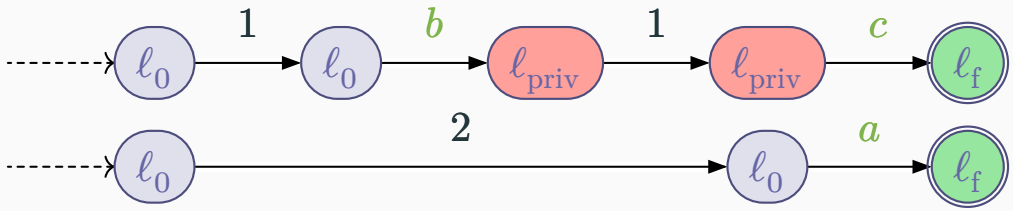
- There exist (at least) two runs of duration $d = 2$:



Example

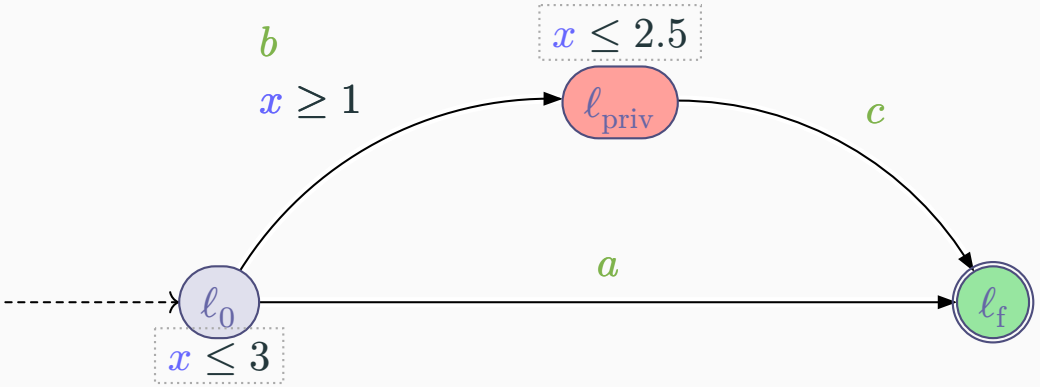


- There exist (at least) two runs of duration $d = 2$:

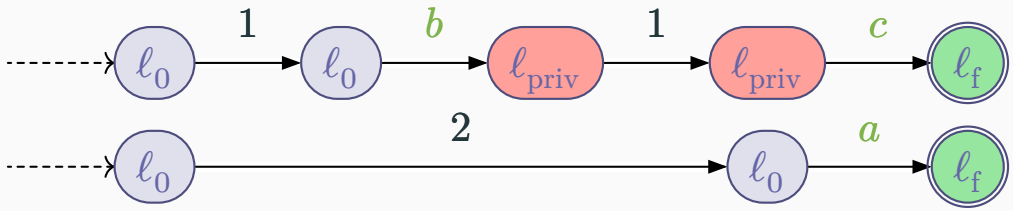


- ET-opaque for $d = 2$
- $\exists$-ET-opaque

Example



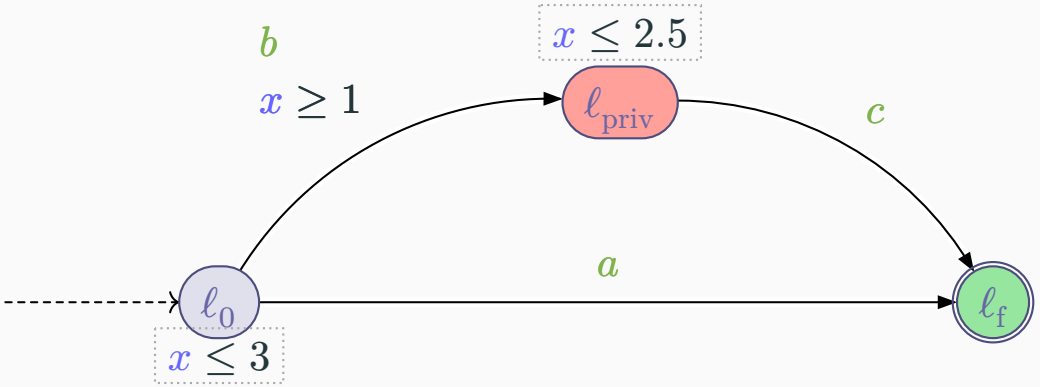
- There exist (at least) two runs of duration $d = 2$:



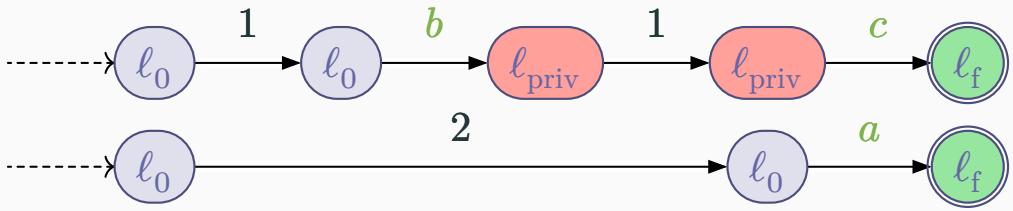
- ET-opaque for $d = 2$
- $\exists$-ET-opaque

- $D_{\ell_{\text{priv}}} = [1, 2.5] \neq D_{\neg\ell_{\text{priv}}} = [0, 3]$

Example



- There exist (at least) two runs of duration $d = 2$:



- ET-opaque for $d = 2$
- $\exists$-ET-opaque

- $D_{\ell_{\text{priv}}} = [1, 2.5] \neq D_{\neg\ell_{\text{priv}}} = [0, 3]$

- Not full-ET-opaque

ET-opacity in timed automata

ET-opacity decision problem

Is the given timed automaton ET-opaque?

[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

ET-opacity in timed automata

ET-opacity decision problem

Is the given timed automaton ET-opaque?

→ ET-opacity is decidable for timed automata!

[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

Analyzing timing behaviors of programs

Problems

Timing analysis of programs is **hard**: it depends not just on code, but also on **low-level details** of execution

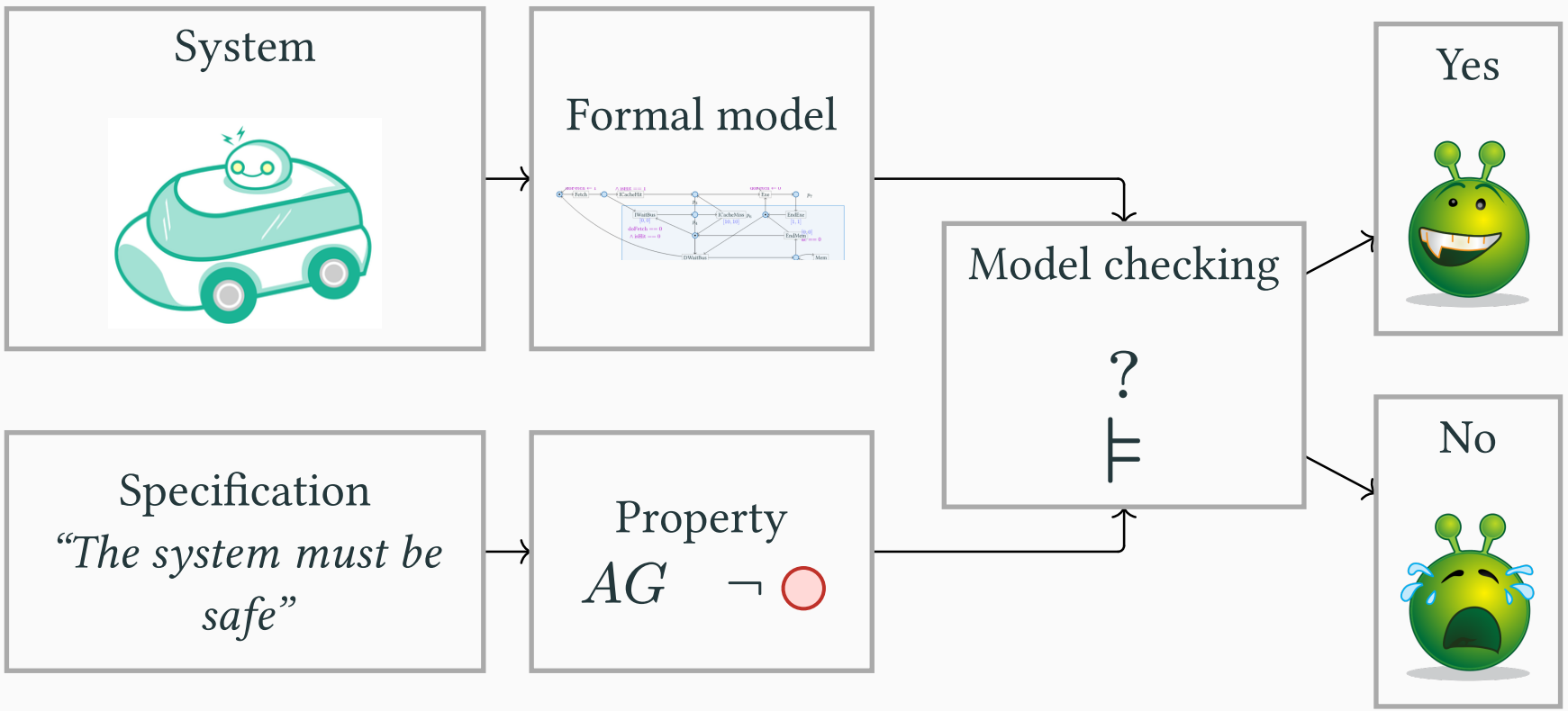
Impact of hardware

- ET is heavily influenced by the **micro-architecture**
  - Especially: pipelines, caches, memory hierarchy

Limitations of existing techniques

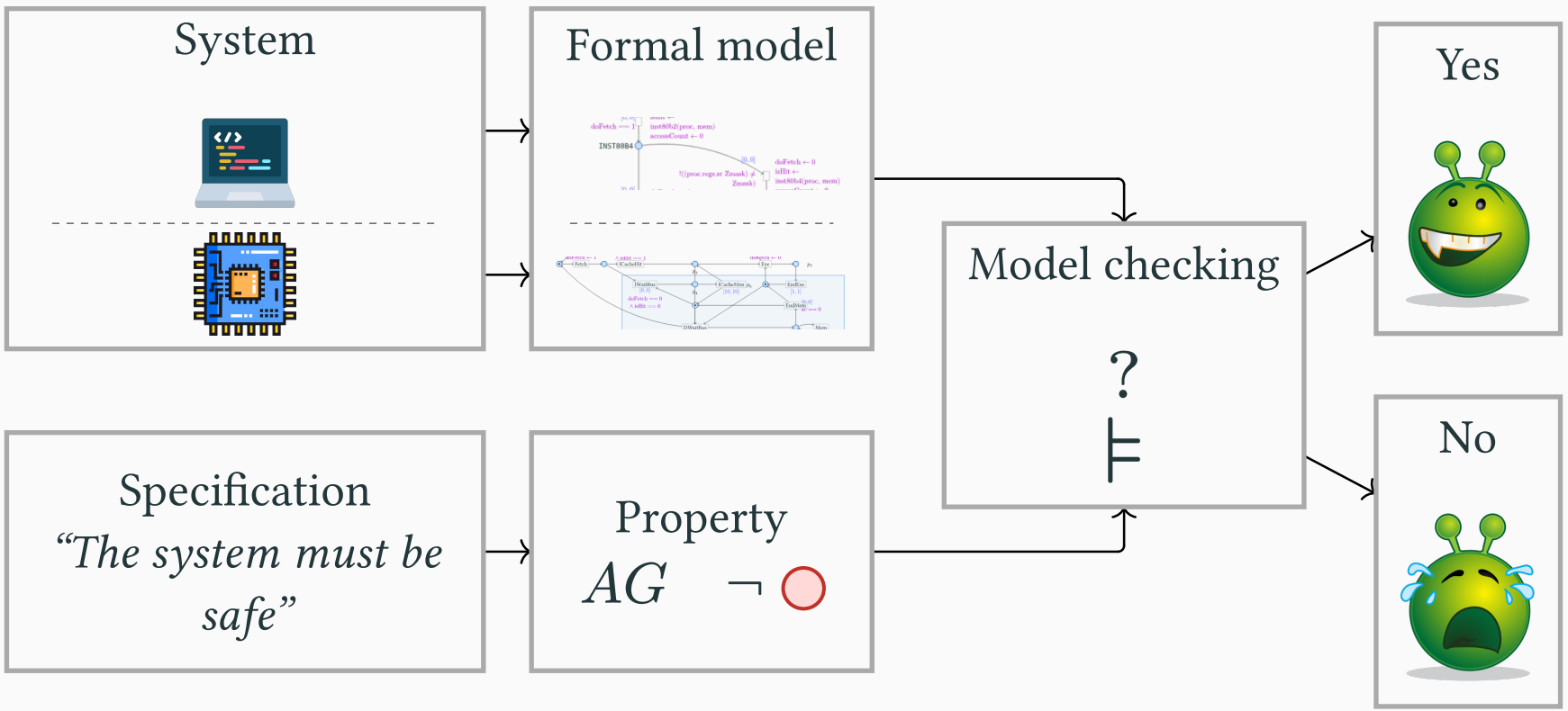
- Most abstract time away or focus on **coarse properties**
  - e.g., schedulability analysis, worst-case execution time (WCET)
- **Insufficient** for fine-grained timing behaviors
  - e.g., detecting or mitigating *timed side-channels*

Model checking



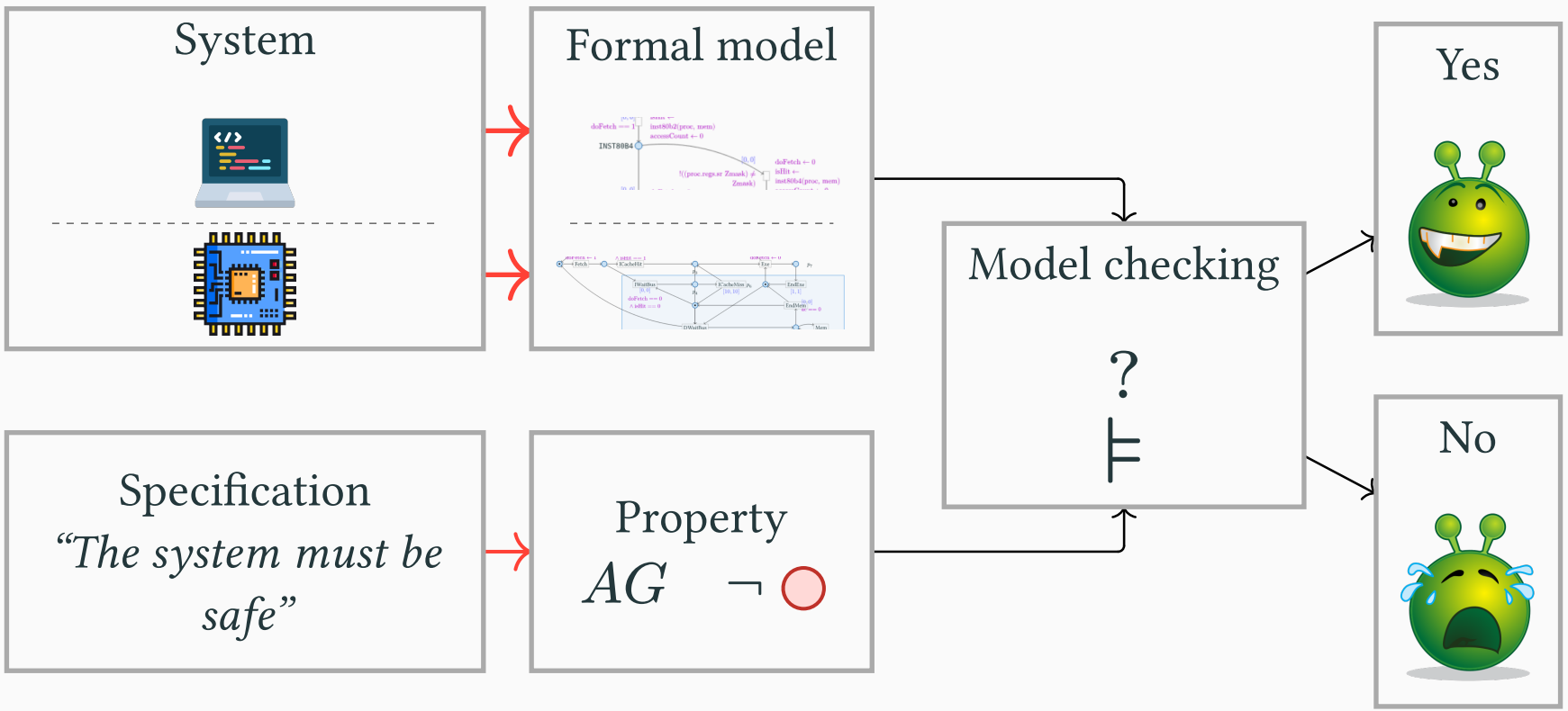
Question: does the model of the system **satisfy** the property?

Model checking



Question: does the model of the system **satisfy** the property?

Model checking



Question: does the model of the system **satisfy** the property?

Our contributions in a nutshell

[And+25]

- A **modular** and **automated** approach to build formal models to analyze **timing behaviors**
  - *binary code with the hardware*

[And+25] Étienne André *et al.*, "Verifying Timed Properties of Programs in IoT nodes using Parametric Time Petri Nets," in *SAC 2025*, 2025.

Our contributions in a nutshell

[And+25]

⚙️ A **modular** and **automated** approach to build formal models to analyze **timing behaviors**

- *binary code with the hardware*

💻 An **implementation**

- *targeting a realistic micro-architecture of a simple micro-controller*
- *producing **time Petri nets** models*

[And+25] Étienne André *et al.*, "Verifying Timed Properties of Programs in IoT nodes using Parametric Time Petri Nets," in *SAC 2025*, 2025.

Our contributions in a nutshell

[And+25]

⚙️ A **modular** and **automated** approach to build formal models to analyze **timing behaviors**
- *binary code with the hardware*

💻 An **implementation**
- *targeting a realistic micro-architecture of a simple micro-controller*
- *producing **time Petri nets** models*

⚔️ An application to **timing attacks** in **C programs** using the **Roméo** model checker

---

[And+25] Étienne André *et al.*, "Verifying Timed Properties of Programs in IoT nodes using Parametric Time Petri Nets," in *SAC 2025*, 2025.

Modeling hardware

Considered hardware

Our hardware

- Model of the **processor architecture**
  - *relatively simple micro-architecture similar to ARM Cortex M0+ core, with a 2-stage pipeline (Fetch and Execute)*
- Model of the **instruction set** architecture (ISA)
  - *ARMv6-M ISA*

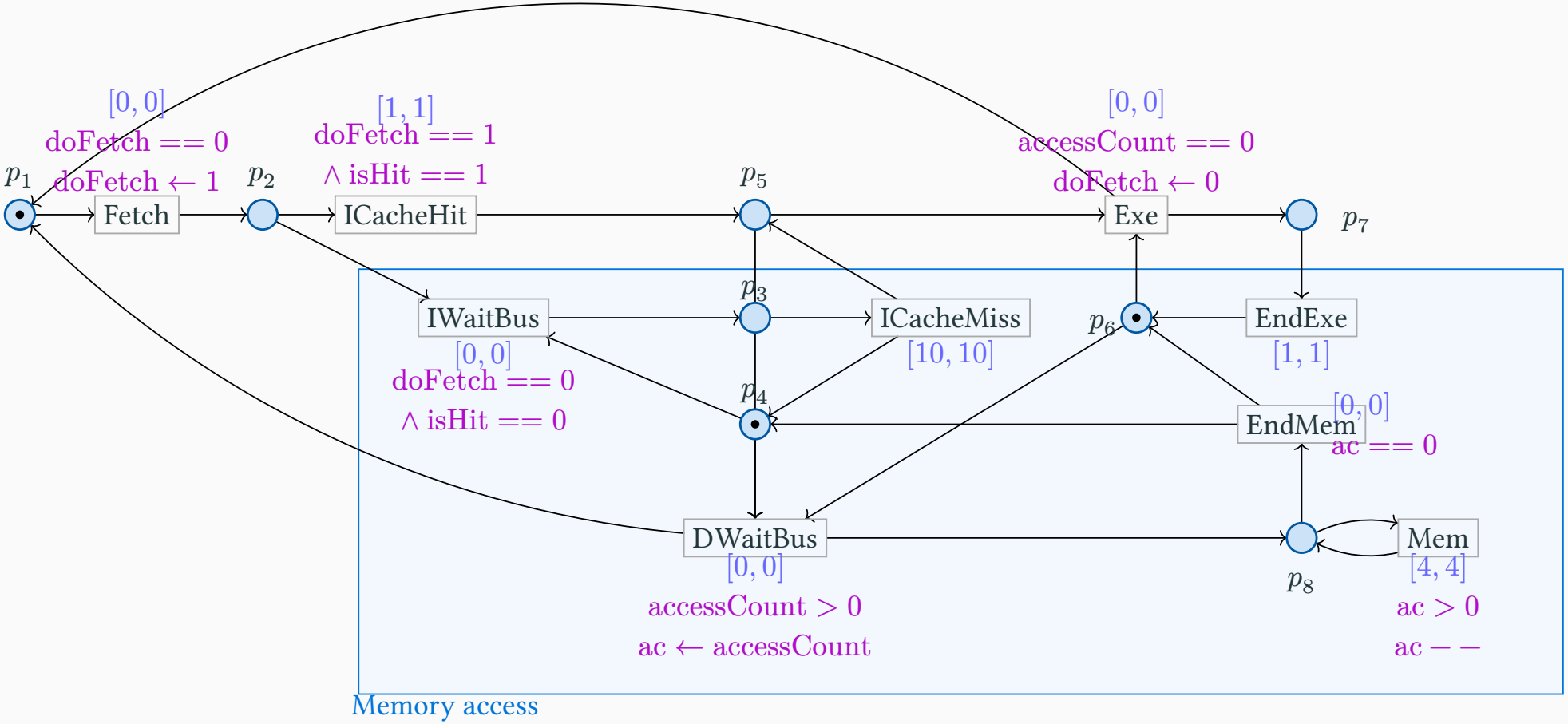
Features

- Execution pipeline of the processor
- Unique memory space
  - *(instructions and data)*
- Bus between the processor and memory
- Direct-mapped **instruction cache**
  - *with 16 lines of 32 bytes*
  - *no actual instructions, but only information about their presence*
- **No data** cache

- Among the limitations: no switch/case, function pointers. . .

PTPN hardware model

- doFetch, isHit and accessCount: variables used to synchronize with the software

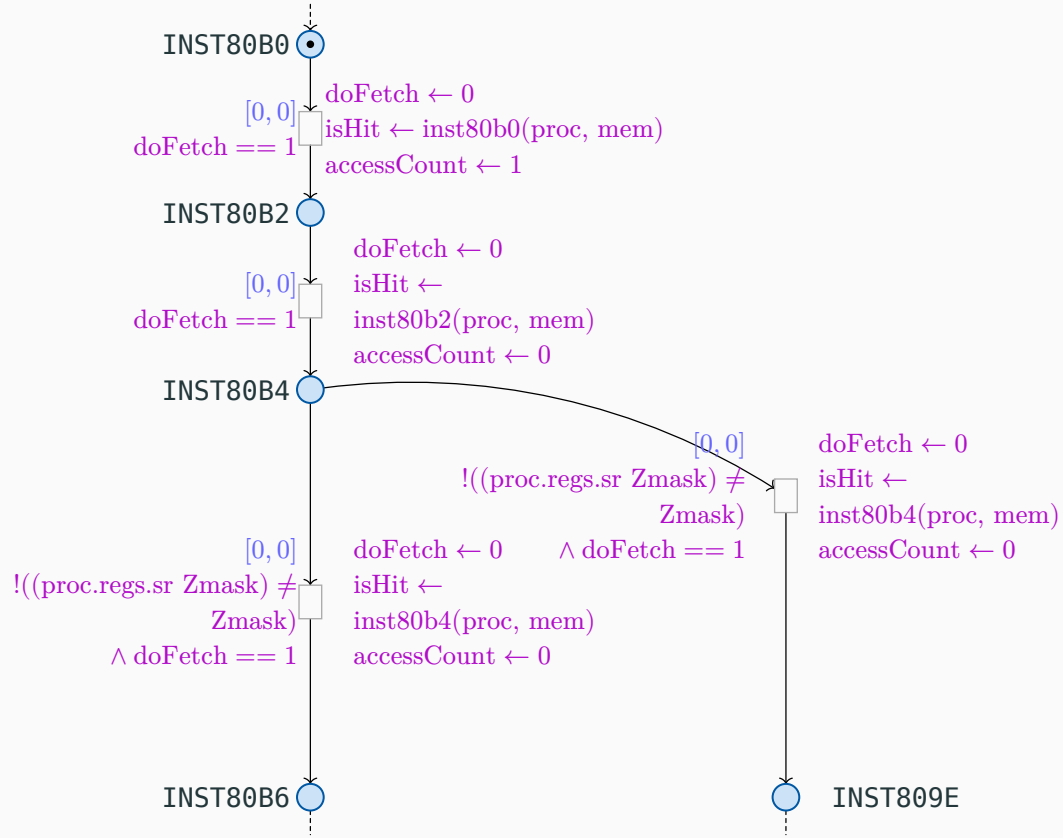


Modeling programs

PTPN software model

Software

- Captures the **binary code** of the program *(ARMv6-M)*
  - Firing a transition corresponds to executing the instruction
  - Pipeline fetch: doFetch
  - Memory access: accessCount and isHit
- Structurally identical to the **control flow graph**

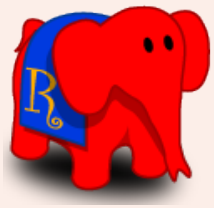


A fully automated translation

A fully automated translation

- Including the hardware and software models
- Written in C and Python
- All the way from the C source code to the PTPN model
- Entirely open source *(github.com/DylanMarinho/codeToPN/)*

Target model checker: Roméo [Lim+09]



- **Parametric timed model checker** supporting (extensions) of PTPNs
- Including **C-like code** to be executed during transitions

[Lim+09] Didier Lime, Olivier H. Roux, Charlotte Seidner, and Louis-Marie Traonouez, "Romeo: A Parametric Model-Checker for Petri Nets with Stopwatches," in *TACAS 2009*, 2009.

Application to security properties

Application to security properties

Timing attacks

- Attacker can **infer information** about the secret key by measuring the **execution time** of the program
  - *e.g., password checking program*

Execution-time opacity [And+23]

"Can the attacker deduce internal behavior by only observing the execution time?"

[And+23] Étienne André, Engel Lefaucheux, Didier Lime, Dylan Marinho, and Jun Sun, "Configuring Timing Parameters to Ensure Execution-Time Opacity in Timed Automata," in *TiCSA@ETAPS 2023*, 2023.

Application to security properties

Timing attacks

- Attacker can **infer information** about the secret key by measuring the **execution time** of the program
  - *e.g., password checking program*

Execution-time opacity [And+23]

"Can the attacker deduce internal behavior by only observing the execution time?"

- Use of timing parameters: to measure execution times

[And+23] Étienne André, Engel Lefaucheux, Didier Lime, Dylan Marinho, and Jun Sun, "Configuring Timing Parameters to Ensure Execution-Time Opacity in Timed Automata," in *TiCSA@ETAPS 2023*, 2023.

Which of the following two programs is not secure?

Program 1

C

```c
int main() {
    int i; int length = 10;
    char ca[11] = "patehenaff";
    char cb[11] = "pasta";

    for (i = 0; i < length; i++){
        if (ca[i] != cb[i]) {
            return 0; // false
        }
    }
    return 1; // true
}
```

Program 2

C

```c
int main() {
    int i;  int length = 10;
    char ca[11] = "patehenaff";
    char cb[11] = "pasta";

    int result = 1; // true

    for (i = 0; i < length; i++){
        result &= (ca[i] == cb[i]);
    }
    return result;
}
```

Which of the following two programs is not secure?

Program 1

```c
int main() {
    int i; int length = 10;
    char ca[11] = "patehenaff";
    char cb[11] = "pasta";

    for (i = 0; i < length; i++){
        if (ca[i] != cb[i]) {
            return 0; // false
        }
    }
    return 1; // true
}
```

Unsecure

Program 2

```c
int main() {
    int i;  int length = 10;
    char ca[11] = "patehenaff";
    char cb[11] = "pasta";

    int result = 1; // true

    for (i = 0; i < length; i++){
        result &= (ca[i] == cb[i]);
    }
    return result;
}
```

Secure

Which of the following two programs is not secure?

Program 1

```c
int main() {
    int i; int length = 10;
    char ca[11] = "patehenaff";
    char cb[11] = "pasta";

    for (i = 0; i < length; i++){
        if (ca[i] != cb[i]) {
            return 0; // false
        }
    }
    return 1; // true
}
```

Unsecure - ET sensitive

- 758 for the secret password
- {362, 404, 446, 488, 530, 572, 614, 656, 698, 740} for any other password

Program 2

```c
int main() {
    int i;  int length = 10;
    char ca[11] = "patehenaff";
    char cb[11] = "pasta";

    int result = 1; // true

    for (i = 0; i < length; i++){
        result &= (ca[i] == cb[i]);
    }
    return result;
}
```

Secure - Constant ET: 876

Is this third program secure?

Program 3

```c
int main () {
  int i ;
  int length = 10; // length of the strings
  char ca[11] = " patehenaff " ;
  char cb[11] = " pasta " ;

  int result = 1; // true
  for (i=0; i<length ; i++) {
    if (ca[i] == cb[i]) {
      result &= 1;
    } else {
      result &= 0;
    }
  }
  return result ;
}
```

Is this third program secure?

Program 3

```c
int main () {
  int i ;
  int length = 10; // length of the strings
  char ca[11] = " patehenaff " ;
  char cb[11] = " pasta " ;

  int result = 1; // true
  for (i=0; i<length ; i++) {
    if (ca[i] == cb[i]) {
      result &= 1;
    } else {
      result &= 0;
    }
  }
  return result ;
}
```

- It seems so: very close to the former secure program
- But it is not due to the **instruction cache**
  - 876 for the secret password
  - {816, 822, 828, 834, 840, 846, 852, 858, 864, 870} for any other password

Is this third program secure?

Program 3

```c
int main () {
  int i ;
  int length = 10; // length of the strings
  char ca[11] = " patehenaff " ;
  char cb[11] = " pasta " ;

  int result = 1; // true
  for (i=0; i<length ; i++) {
    if (ca[i] == cb[i]) {
      result &= 1;
    } else {
      result &= 0;
    }
  }
  return result ;
}
```

- It seems so: very close to the former secure program
- But it is not due to the **instruction cache**
  - 876 for the secret password
  - {816, 822, 828, 834, 840, 846, 852, 858, 864, 870} for any other password

We can **reconfigure** the program, by **making it opaque**

- *adding 6 nop instructions at the end of one branch*
- *(see paper)*

Conclusion and perspectives

Conclusion

End-to-end approach on **binary code timing analysis**, subject to micro-architectural constraints

- automated production of **timed formal models** of both the program and the hardware architecture
- using (parametric) time Petri nets

Illustrative case-study: **detection of timing leaks** in C programs

- via parameter synthesis techniques using **Roméo**
- (manual) **reconfiguration** of the program to make it opaque

Perspectives

- Modeling and analysis of programs on **multicore** architectures
- Automatic modification of a program to make it **opaque**

- Handling **more complex attacks**
  - Fault-injection
  - Cache side-channels
    - *flush and reload, prime and probe*
  - Energy-based attacks

- Formal proof of our translation?

27 January 2026 | APR Seminar | Paris, France

Detecting Timing Leaks of Programs

using Parametric Timed Model Checking

Dylan Marinho

Sorbonne Université, CNRS UMR 7606, LIP6

Bibliography

Bibliography

[AD94] Rajeev Alur and David L. Dill, “A Theory of Timed Automata,” *Theoretical Computer Science*, 1994.

[Cas09] Franck Cassez, “The Dark Side of Timed Opacity,” in *ISA*, 2009.

[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, “Guaranteeing Timed Opacity using Parametric Timed Model Checking,” *TOSEM*, 2022.

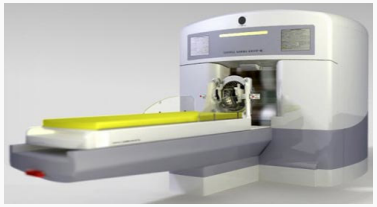
[And+25] Étienne André *et al.*, “Verifying Timed Properties of Programs in IoT nodes using Parametric Time Petri Nets,” in *SAC 2025*, 2025.

[Lim+09] Didier Lime, Olivier H. Roux, Charlotte Seidner, and Louis-Marie Traonouez, “Romeo: A Parametric Model-Checker for Petri Nets with Stopwatches,” in *TACAS 2009*, 2009.

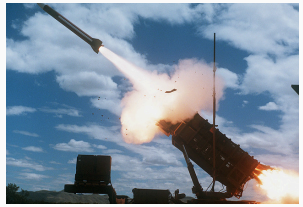
[And+23] Étienne André, Engel Lefaucheux, Didier Lime, Dylan Marinho, and Jun Sun, “Configuring Timing Parameters to Ensure Execution-Time Opacity in Timed Automata,” in *TiCSA@ETAPS 2023*, 2023.

Additional information

Explanation of the pictures



- Therac-25 bug
- Computer bug, race condition
- Consequences: multiple fatalities

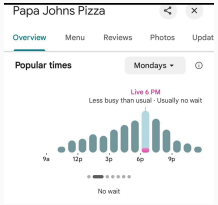


- Allusion to the MIM-104 Patriot Missile Failure (Iraq, 1991)
- 28 fatalities, hundreds of injured
- Computer bug: software error (clock drift)
- (Picture of an actual MIM-104 Patriot Missile, though not the one of 1991)

- Allusion to the sinking of the Sleipner A offshore platform (Norway, 1991)
- No fatalities
- Computer bug: inaccurate finite element analysis modeling
- (Picture actually from the Deepwater Horizon Offshore Drilling Platform)

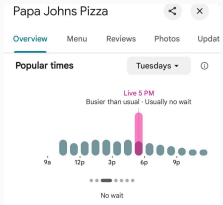
Explanation of the pictures

- Ariane flight V88 (France, 1996)
- Computer bug (notably integer overflow)
- Consequences: US$370 million



- USA, June 2025
- Empty bars during Iran's riposte against US military bases.

(Dr. Dominic Ng)

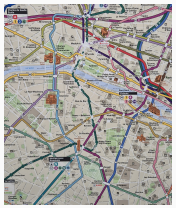


- USA, 24 June 2025
- After the Israel-Iran ceasefire

(Dr. Dominic Ng)

Explanation of the pictures

- Prefecture de Police, Paris (France, 17th July 2024 at 10:10 PM)
- Delivers in front of the Prefecture de Police, Paris
- The day before closing the center of Paris to prepare the 2024 Olympic Games



- Paris Metro map, Madeleine station (Paris, France)

Licensing

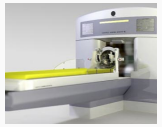
Sources of the graphics

- Title : Explosion of first Ariane 5 flight, June 4, 1996
- Author : ESA
- Source : https://www.esa.int/ESA_Multimedia/Images/2009/09/Explosion_of_first_Ariane_5_flight_
- License : ESA Standard Licence

- Title : Deepwater Horizon Offshore Drilling Platform on Fire
- Author : ideum
- Source : https://secure.flickr.com/photos/ideum/4711481781/
- License : Creative Commons cc-by-sa

- Title : DA-SC-88-01663
- Author : imcomkorea
- Source : https://secure.flickr.com/photos/imcomkorea/3017886760
- License : Creative Commons cc-by-nc-nd

Sources of the graphics



- Title : Therac-25
- Author : ?
- Source : https://arquivonuclear.blogspot.com/2011/03/therac-25.html
- License : unknown

- Title : Autonomous robot vehicle or ADV typically used for food or grocery delivery
- Author : Rlistmedia
- Source : https://commons.wikimedia.org/wiki/File:Autonomous_delivery_robot_vehicles_ADV.png
- License : Creative Commons cc-by

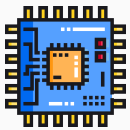
- Title : Smiley green alien big eyes (aaah)
- Author : LadyofHats
- Source : https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
- License : Public domain

Sources of the graphics

- Title : Smiley green alien big eyes (cry)
- Author : LadyofHats
- Source : https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
- License : Public domain

- Title : Smiley green alien exterminate
- Author : LadyofHats
- Source : https://commons.wikimedia.org/wiki/File:Smiley_green_alien_exterminate.svg
- License : Public domain

- Source: Flaticon.com



- Source: Flaticon.com

License of this document

Authors: Étienne André, **Dylan Marinho**

Version: 2026-01-27