



Laboratoire
Méthodes
Formelles

université
PARIS-SACLAY



école
normale
supérieure
paris—saclay

Alternating timed automata

Old story and recent developments

Patricia Bouyer

LMF, Université Paris-Saclay, CNRS, ENS Paris-Saclay

Based on recent joint work with B. Srivathsan and Vaishnavi Vishwanath (both at CMI, Chennai, India). I thank Vaishnavi for her slides.

Model-checking

Does the system satisfy the specification?



Model-checking

Does the system satisfy the specification?



Is $L(\mathcal{A} \times \mathcal{B}_{\neg\varphi})$ empty?

Model-checking

Does the system satisfy the specification?



Is $L(\mathcal{A} \times \mathcal{B}_{\neg\varphi})$ empty?

- Specifications can look like:
 - Every a is followed by a b : $\mathbf{G}(a \Rightarrow \mathbf{X}b)$
 - a occurs at least once, and until then only b 's occur: $b\mathbf{U}a$

Model-checking

Does the system satisfy the specification?



Is $L(\mathcal{A} \times \mathcal{B}_{\neg\varphi})$ empty?

- ▶ Specifications can look like:
 - Every a is followed by a b : $\mathbf{G}(a \Rightarrow \mathbf{X}b)$
 - a occurs at least once, and until then only b 's occur: $b\mathbf{U}a$
- ▶ These can be e.g. expressed in Linear Temporal Logic (LTL)

Model-checking real-time systems

Does the system satisfy the specification?

$\mathcal{A}$

Timed system

Timed spec.

φ

Model-checking real-time systems

Does the system satisfy the specification?

$\mathcal{A}$

Timed system

Timed automata

Timed spec.

φ

Timed logic

Model-checking real-time systems

Does the system satisfy the specification?

$\mathcal{A}$

Timed system

Timed automata

Timed spec.

φ

Timed logic

► Specifications can look like:

- Every a is followed by a b within 2 time units: $\mathbf{G}(a \Rightarrow \mathbf{X}_{[0,2]}b)$
- a occurs at least once before time unit 5, and until then only b 's occur: $b\mathbf{U}_{[0,5]}a$

Model-checking real-time systems

Does the system satisfy the specification?

$\mathcal{A}$

Timed system

Timed automata

Timed spec.

φ

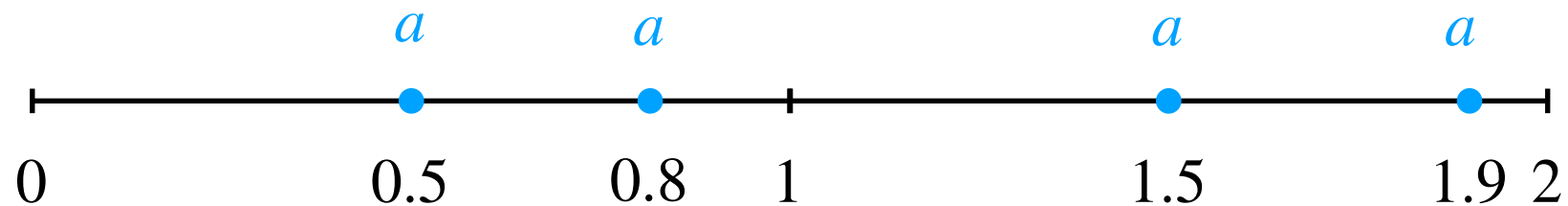
Timed logic

- ▶ Specifications can look like:
 - Every a is followed by a b within 2 time units: $\mathbf{G}(a \Rightarrow \mathbf{X}_{[0,2]}b)$
 - a occurs at least once before time unit 5, and until then only b 's occur: $b\mathbf{U}_{[0,5]}a$
- ▶ These can be e.g. expressed in Metric Temporal Logic (MTL)

Timed automata

- ▶ Timed word: $(a_1, d_1)(a_2, d_2)\dots$ where a_i are actions and d_i are delays

$$w = (a, 0.5)(a, 0.3)(a, 0.7)(a, 0.4)$$

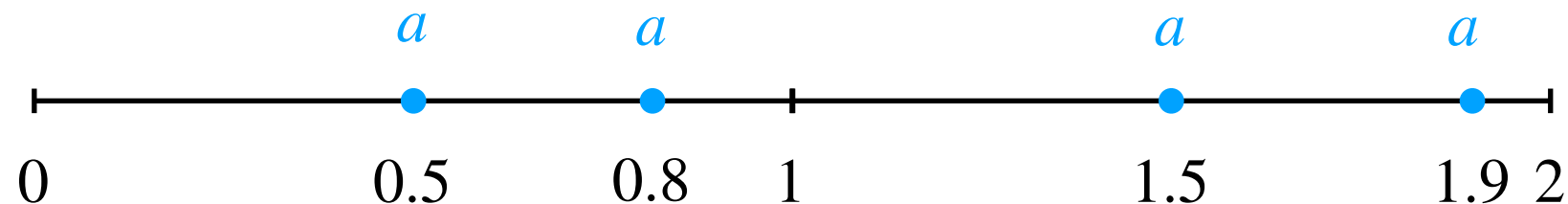


Timed automata

- ▶ Timed word: $(a_1, d_1)(a_2, d_2)\dots$ where a_i are actions and d_i are delays

$$w = (a, 0.5)(a, 0.3)(a, 0.7)(a, 0.4)$$

Could be time stamps as well



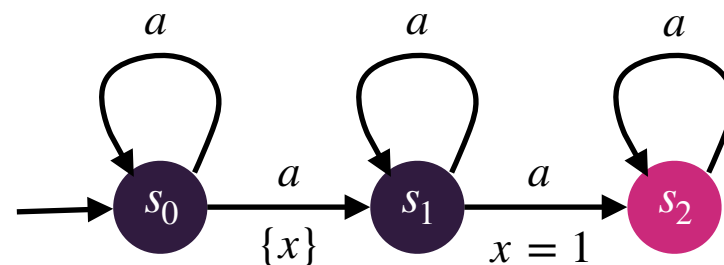
Timed automata

- ▶ Finite state automata with clocks
- ▶ Can check constraints on clock values and reset clocks to **0**

Timed automata

- ▶ Finite state automata with clocks
- ▶ Can check constraints on clock values and reset clocks to 0

Example: two a 's are 1 t.u. apart



t.u. =
time unit

Why are they so popular?

Why are they so popular?

- ▶ Interesting expressiveness

Why are they so popular?

- ▶ Interesting expressiveness
- ▶ Reachability, (T)CTL model-checking decidable

Why are they so popular?

- ▶ Interesting expressiveness
- ▶ Reachability, (T)CTL model-checking decidable
- ▶ Region abstraction

Why are they so popular?

- ▶ Interesting expressiveness
- ▶ Reachability, (T)CTL model-checking decidable
- ▶ Region abstraction
- ▶ **Zone-based algorithms!**
 - Extrapolation vs entailment

Why are they so popular?

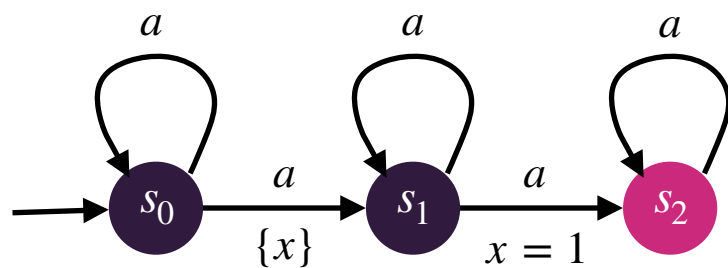
- ▶ Interesting expressiveness
- ▶ Reachability, (T)CTL model-checking decidable
- ▶ Region abstraction
- ▶ **Zone-based algorithms!**
 - Extrapolation vs entailment
- ▶ Tools exist: Uppaal, TChecker, ...

What are their limitations?

- ▶ They are not closed under complementation
- ▶ Universality of two-clock timed automata is undecidable

What are their limitations?

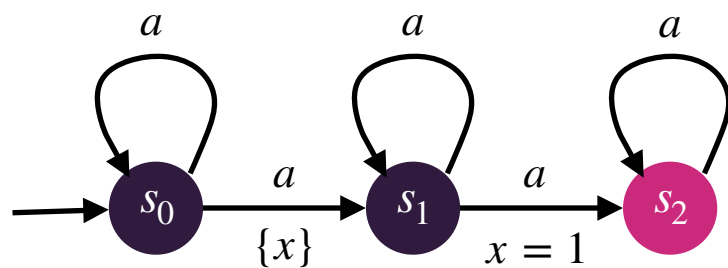
- ▶ They are not closed under complementation
- ▶ Universality of two-clock timed automata is undecidable



cannot be complemented

What are their limitations?

- ▶ They are not closed under complementation
- ▶ Universality of two-clock timed automata is undecidable



cannot be complemented

See my lecture notes if you wish

Model-checking real-time systems

Does the system satisfy the specification?

$\mathcal{A}$

Timed system

Timed automata

Timed spec.

φ

Timed logic

Model-checking real-time systems

Does the system satisfy the specification?

$\mathcal{A}$

Timed system

Timed automata

Timed spec.

φ

Timed logic

- ▶ No timed-automata-based approach can be used for analyzing a logic closed under negation

Model-checking real-time systems

Does the system satisfy the specification?

$\mathcal{A}$

Timed system

Timed automata

Timed spec.

φ

Timed logic

- ▶ No timed-automata-based approach can be used for analyzing a logic closed under negation

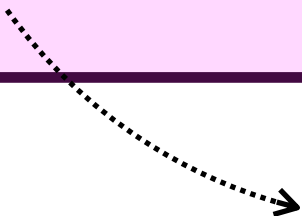
Add alternation...

Alternating timed automata

- ▶ A natural extension of timed automata
- ▶ Transitions are now formulas

Alternating timed automata

- ▶ A natural extension of timed automata
- ▶ Transitions are now formulas



Disjunction = non-determinism
Conjunction = new feature

Alternating timed automata

- ▶ A natural extension of timed automata
- ▶ Transitions are now formulas

Example: no two a 's are 1 t.u. apart

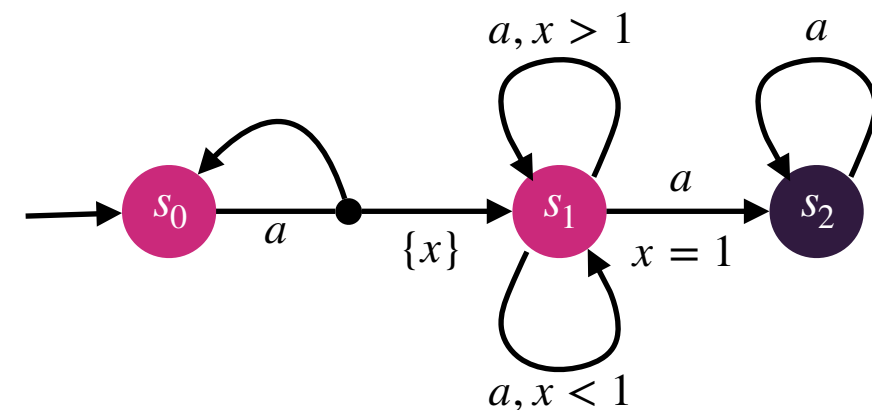
Disjunction = non-determinism

Conjunction = new feature

$$\delta(s_0, a) = s_0 \wedge x \cdot s_1$$

$$\delta(s_1, a) = (x \in [0,1) \wedge s_1) \vee (x \in (1,\infty) \wedge s_1) \vee (x \in [1,1] \wedge s_2)$$

$$\delta(s_2, a) = s_2$$



Alternating timed automata

- ▶ A natural extension of timed automata
- ▶ Transitions are now formulas

Example: no two a 's are 1 t.u. apart

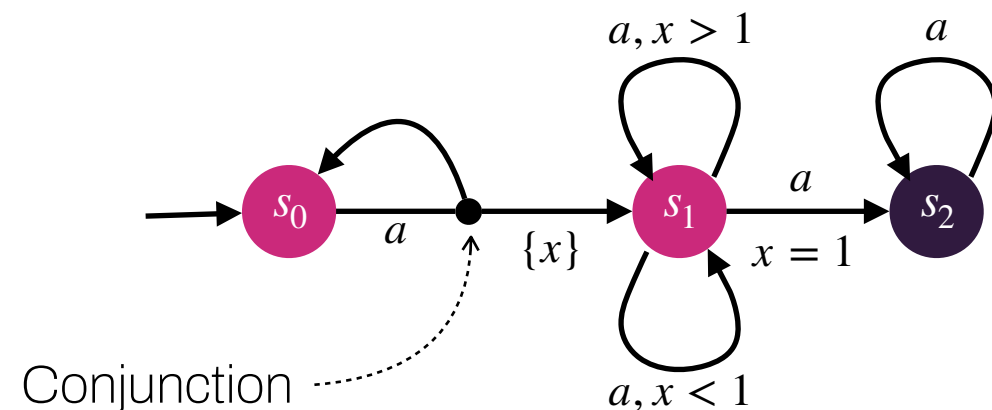
Disjunction = non-determinism

Conjunction = new feature

$$\delta(s_0, a) = s_0 \wedge x \cdot s_1$$

$$\delta(s_1, a) = (x \in [0,1) \wedge s_1) \vee (x \in (1,\infty) \wedge s_1) \vee (x \in [1,1] \wedge s_2)$$

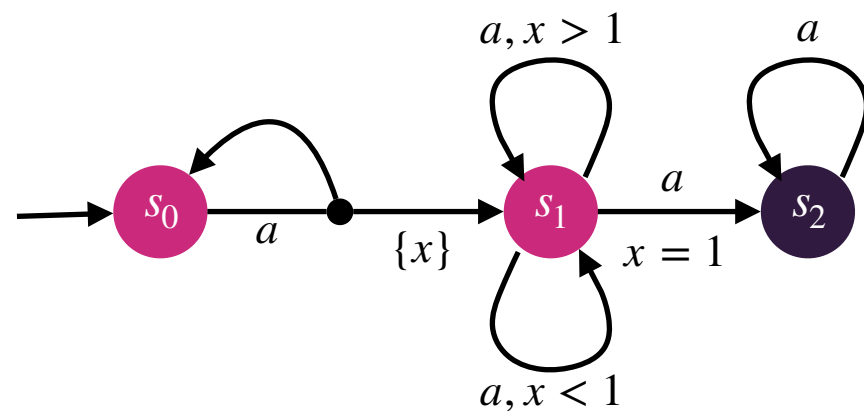
$$\delta(s_2, a) = s_2$$



An example of execution

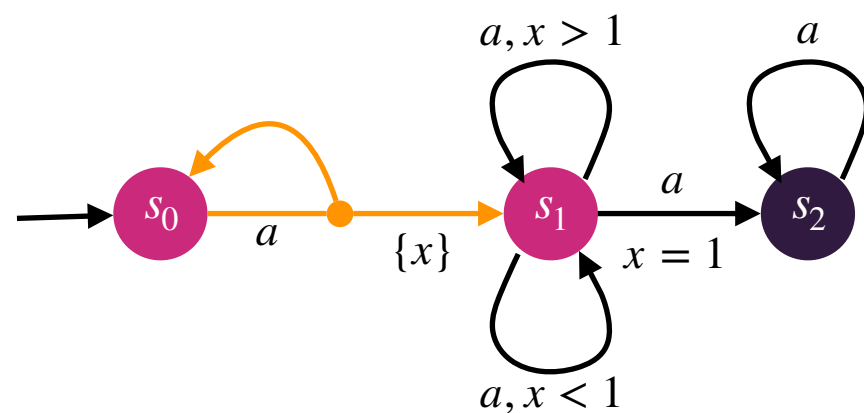
$$w = (0.7, a)(0.4, a)(0.6, a)$$

$\{(s_0, 0)\}$



An example of execution

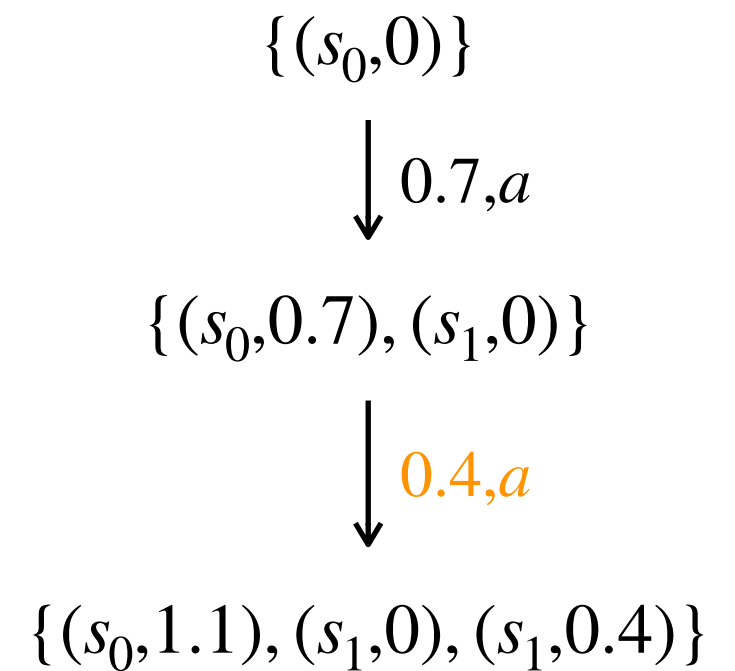
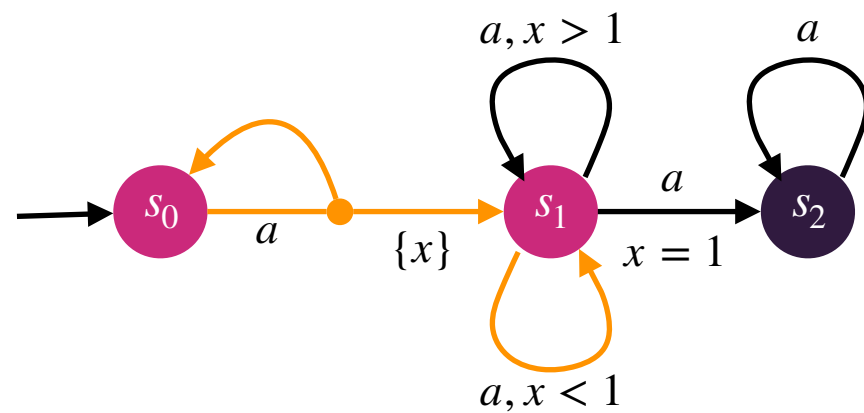
$$w = (0.7, a)(0.4, a)(0.6, a)$$



$$\begin{array}{c} \{(s_0, 0)\} \\ \downarrow \text{0.7, a} \\ \{(s_0, 0.7), (s_1, 0)\} \end{array}$$

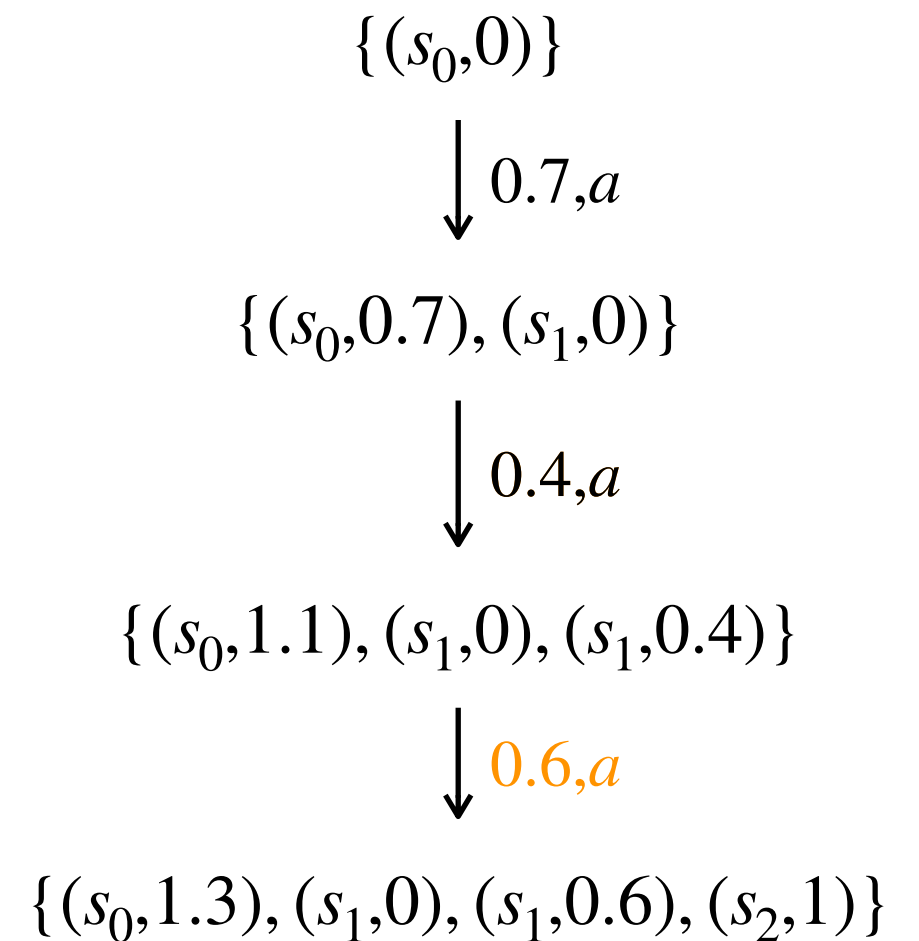
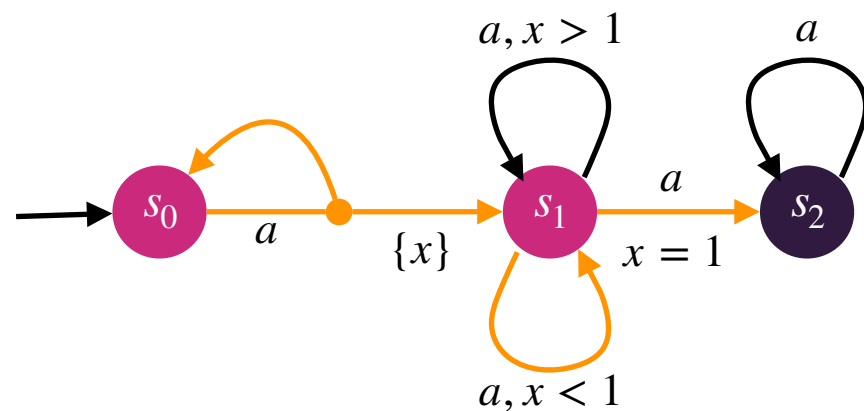
An example of execution

$$w = (0.7, a)(0.4, a)(0.6, a)$$



An example of execution

$$w = (0.7, a)(0.4, a)(0.6, a)$$



Facts

- ▶ Nice closure properties: closed under all Boolean operations
- ▶ Emptiness of universal TA = universality of standard TA

Facts

- ▶ Nice closure properties: closed under all Boolean operations
 - ▶ Emptiness of universal TA = universality of standard TA
- ▶ Emptiness of two-clock ATA is undecidable

Facts

- ▶ Nice closure properties: closed under all Boolean operations
- ▶ Emptiness of universal TA = universality of standard TA

- ▶ Emptiness of two-clock ATA is undecidable

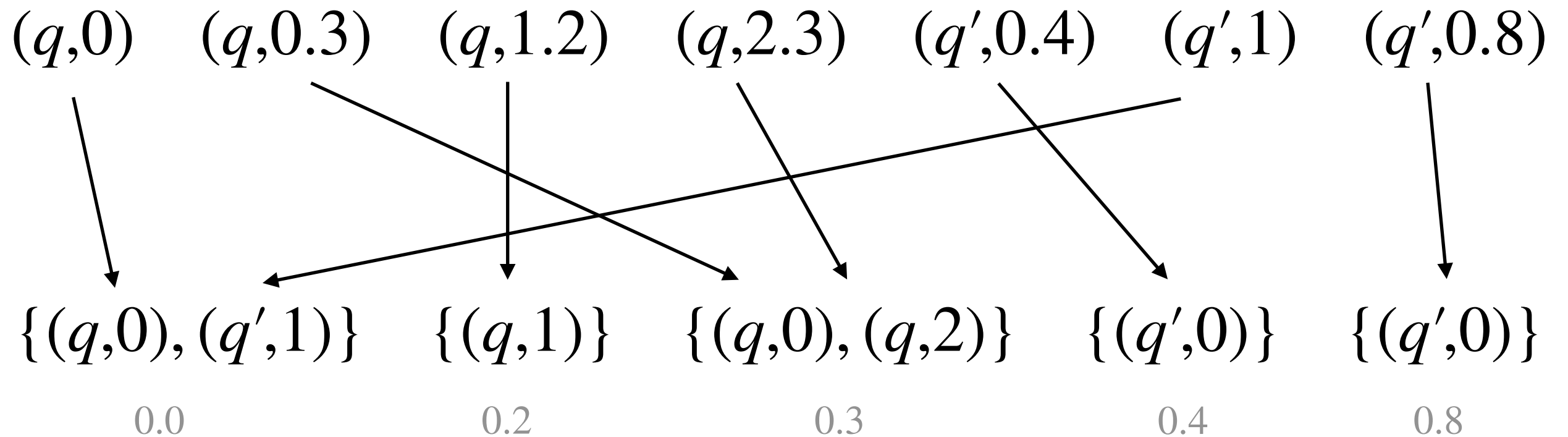
One-clock ATA

- ▶ Emptiness of 1-ATA is decidable for finite words
- ▶ Emptiness of 1-ATA is undecidable for infinite words

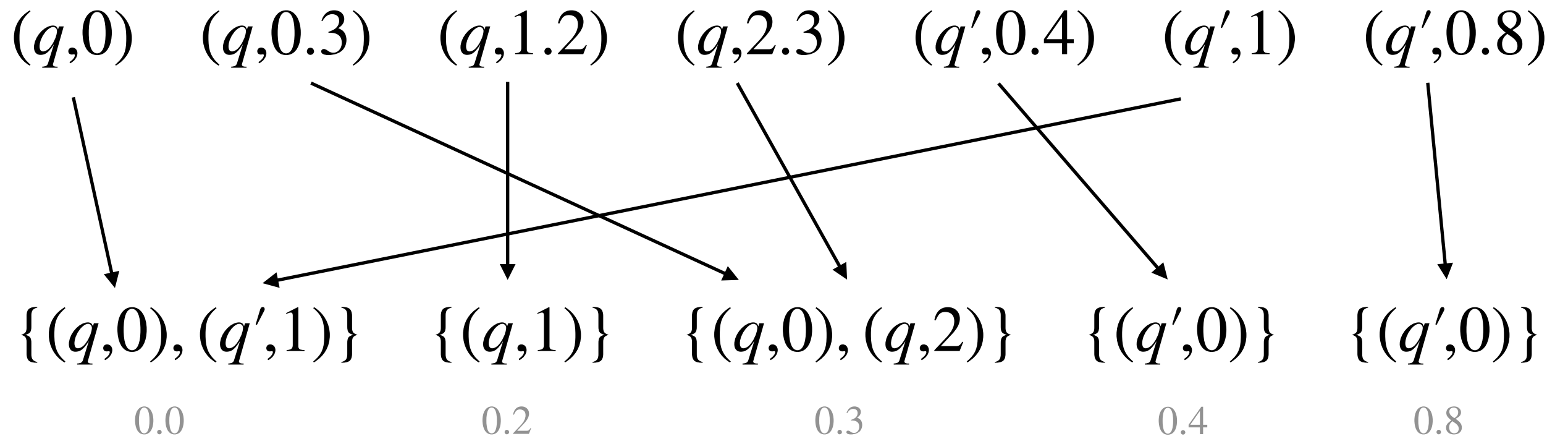
Decidability of 1-ATA (finite words)

$(q,0)$ $(q,0.3)$ $(q,1.2)$ $(q,2.3)$ $(q',0.4)$ $(q',1)$ $(q',0.8)$

Decidability of 1-ATA (finite words)



Decidability of 1-ATA (finite words)



Abstracted into the word:

$\{(q,0), (q',1)\} \cdot \{(q,1)\} \cdot \{(q,0), (q,2)\} \cdot \{(q',0)\} \cdot \{(q',0)\}$

Symbolic computation

Time elapsing

$$\{(q,0), (q',1)\} \cdot \{(q,1)\} \cdot \{(q,0), (q,2)\} \cdot \{(q',0)\} \cdot \{(q',0)\}$$

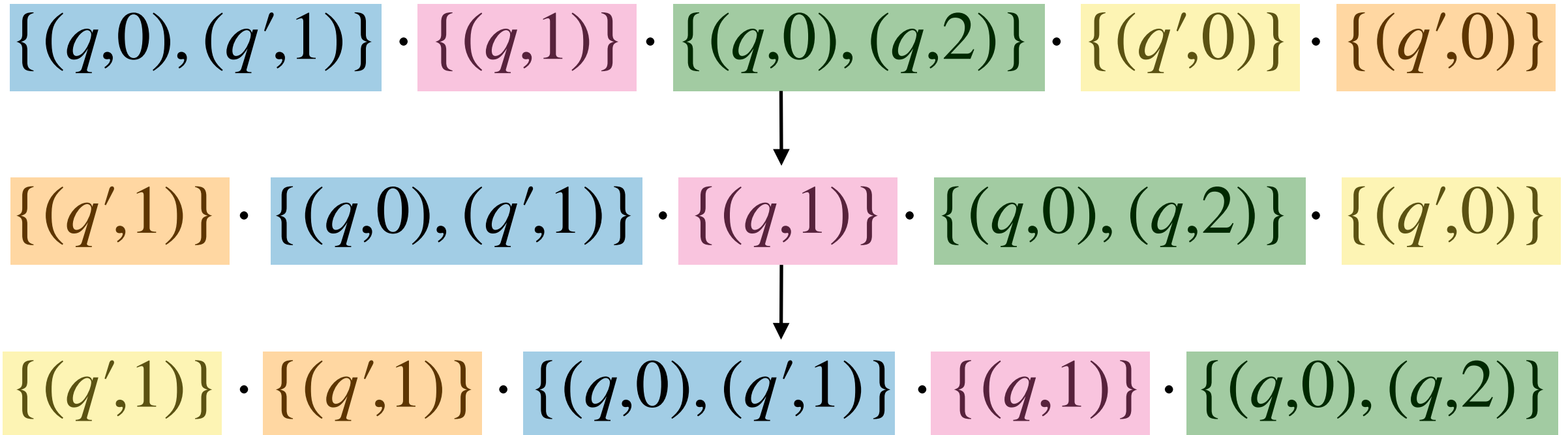
Symbolic computation

Time elapsing

$$\begin{array}{c} \{(q,0), (q',1)\} \cdot \{(q,1)\} \cdot \{(q,0), (q,2)\} \cdot \{(q',0)\} \cdot \{(q',0)\} \\ \downarrow \\ \{(q',1)\} \cdot \{(q,0), (q',1)\} \cdot \{(q,1)\} \cdot \{(q,0), (q,2)\} \cdot \{(q',0)\} \end{array}$$

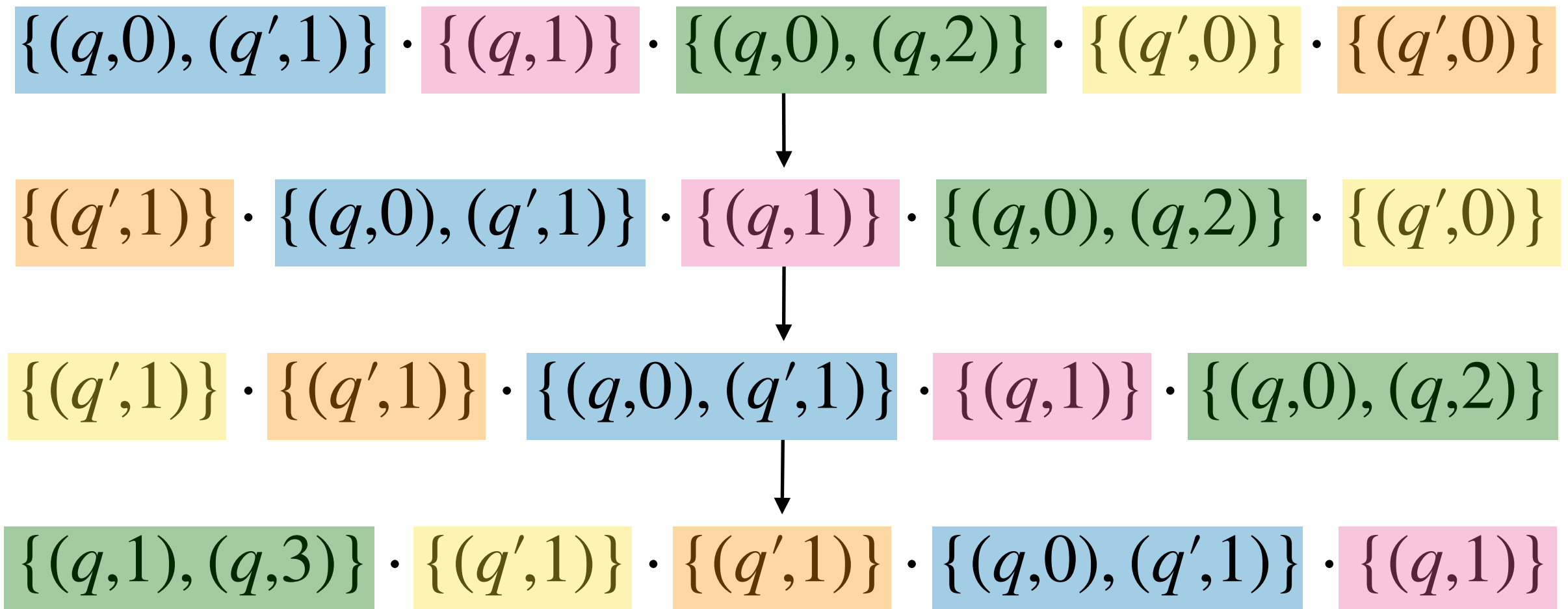
Symbolic computation

Time elapsing



Symbolic computation

Time elapsing



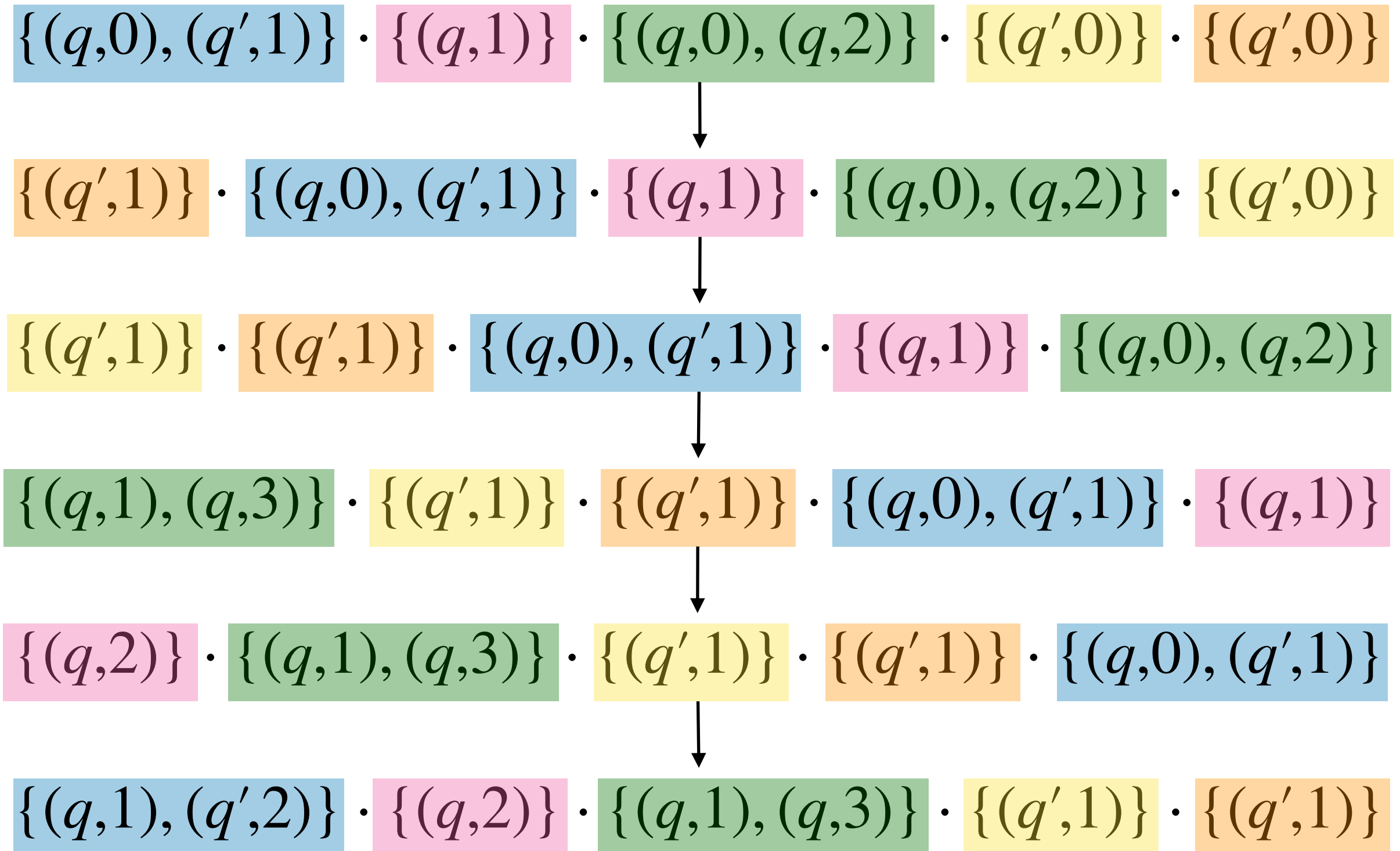
Symbolic computation

Time elapsing



Symbolic computation

Time elapsing



Symbolic computation

Transition $q \xrightarrow{x>2, x:=0} q''$

$$\{(q,1), (q',2)\} \cdot \{(q,2)\} \cdot \{(q,1), (q,3)\} \cdot \{(q',1)\} \cdot \{(q',1)\}$$

Symbolic computation

Transition $q \xrightarrow{x>2, x:=0} q''$

$\{(q,1), (q',2)\} \cdot \{(q,2)\} \cdot \{(q,1), (q,3)\} \cdot \{(q',1)\} \cdot \{(q',1)\}$



$\{(q'',0)\} \cdot \{(q,1), (q',2)\} \cdot \{(q,1)\} \cdot \{(q',1)\} \cdot \{(q',1)\}$

Abstract transition system

Abstract transition system

- ▶ This encoding can be used in an abstract transition system

Abstract transition system

- ▶ This encoding can be used in an abstract transition system
- ▶ This defines a **time-abstract bisimulation**

We
have
regions!

Abstract transition system

- ▶ This encoding can be used in an abstract transition system
- ▶ This defines a **time-abstract bisimulation**
- ▶ There is a **well-quasi-ordering** on abstract configurations, which defines a **well-structured transition system**!

We
have
regions!

Abstract transition system

- ▶ This encoding can be used in an abstract transition system
- ▶ This defines a **time-abstract bisimulation**
- ▶ There is a **well-quasi-ordering** on abstract configurations, which defines a **well-structured transition system**!

We
have
regions!

$\{(q,0), (q',1)\} \cdot \{(q,1)\} \cdot \{(q,0), (q,2)\} \cdot \{(q',0)\} \cdot \{(q',0)\}$

$\sqsubseteq$

$\{(q',1)\} \cdot \{(q,0), (q,2)\} \cdot \{(q',0)\}$

Metric Temporal Logic

Logic MTL

$$\varphi ::= a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}_I \varphi \mid \varphi \mathbf{U}_I \varphi \mid \varphi \tilde{\mathbf{U}}_I \varphi$$

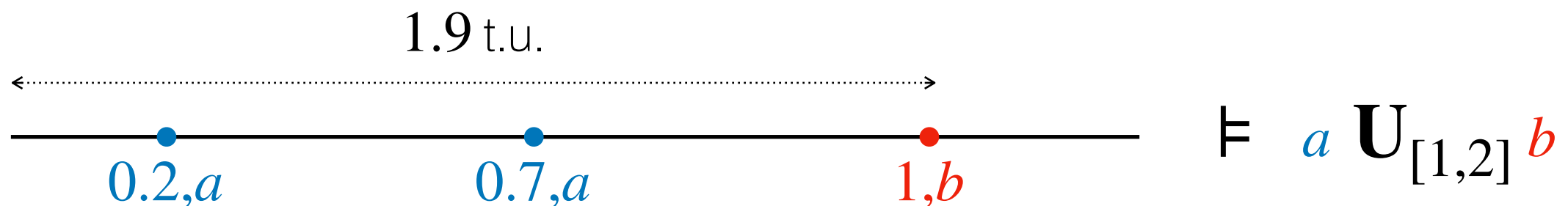
where I is an interval

Metric Temporal Logic

Logic MTL

$$\varphi ::= a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}_I \varphi \mid \varphi \mathbf{U}_I \varphi \mid \varphi \tilde{\mathbf{U}}_I \varphi$$

where I is an interval

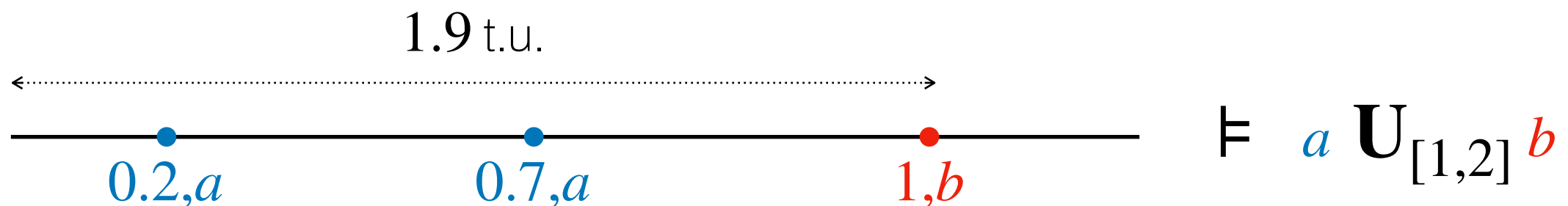


Metric Temporal Logic

Logic MTL

$$\varphi ::= a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}_I \varphi \mid \varphi \mathbf{U}_I \varphi \mid \varphi \tilde{\mathbf{U}}_I \varphi$$

where I is an interval



Logic MITL

Same as MTL, but I should be non-punctual

Some history

- ▶ [OW07] MTL satisfiability for finite words is decidable via 1ATA
- ▶ [OW06] MTL satisfiability for infinite words is undecidable
- ▶ [AFH96] MITL satisfiability for infinite words is decidable via a construction MITL to TA
- ▶ [BEG13] An alternate construction of MITL to TA via 1ATA
- ▶ [Bri+17] MightyL, a tool to convert MITL to via very weak 1ATA
- ▶ [Aks+24] A conversion from MITL to Generalized TA: A zone-based algorithm for GTA emptiness
- ▶ [Ho+26] An algorithm for MITPPL, an extension of MITL. Algo based on some assembly code

Some history

- ▶ [OW07] MTL satisfiability for finite words is decidable via 1ATA
 - ▶ [OW06] MTL satisfiability for infinite words is undecidable
 - ▶ [AFH96] MITL satisfiability for infinite words is decidable via a construction MITL to TA
 - ▶ [BEG13] An alternate construction of MITL to TA via 1ATA
 - ▶ [Bri+17] MightyL, a tool to convert MITL to via very weak 1ATA
 - ▶ [Aks+24] A conversion from MITL to Generalized TA: A zone-based algorithm for GTA emptiness
 - ▶ [Ho+26] An algorithm for MITPPL, an extension of MITL. Algo based on some assembly code
- ▶ M(I)TL verification always goes via TA: no zone-based algorithm is known for 1ATA, multiple tools for TA exist

Our recent work (FoSSaCS'25)

- ▶ Deactivation operator ($\bar{x}$)
- ▶ Zones for 1ATA and a zone-based algorithm for 1ATA
- ▶ Termination of the above due to an entailment relation on zones
 - An algorithm for $\not\models$ -check using existing techniques in TA
 - Show that the $\not\models$ -check is NP-hard in general
- ▶ The case of bounded 1ATA: improvement for $\sqsubseteq$ and some corresponding logic

Our recent work (FoSSaCS'25)

- ▶ Deactivation operator ($\bar{x}$)
- ▶ Zones for 1ATA and a zone-based algorithm for 1ATA
- ▶ Termination of the above due to an entailment relation on zones
 - An algorithm for $\not\models$ -check using existing techniques in TA
 - Show that the $\not\models$ -check is NP-hard in general
- ▶ The case of bounded 1ATA: improvement for $\sqsubseteq$ and some corresponding logic

MTL to 1ATA

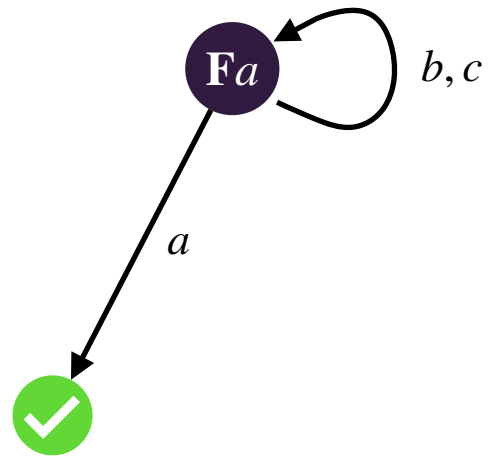
$$\varphi = (\mathbf{F}a)\mathbf{U}_{[1,2]}c$$

$$\Sigma = \{a, b, c\}$$

MTL to 1ATA

$$\varphi = (\mathbf{F}a)\mathbf{U}_{[1,2]}c$$

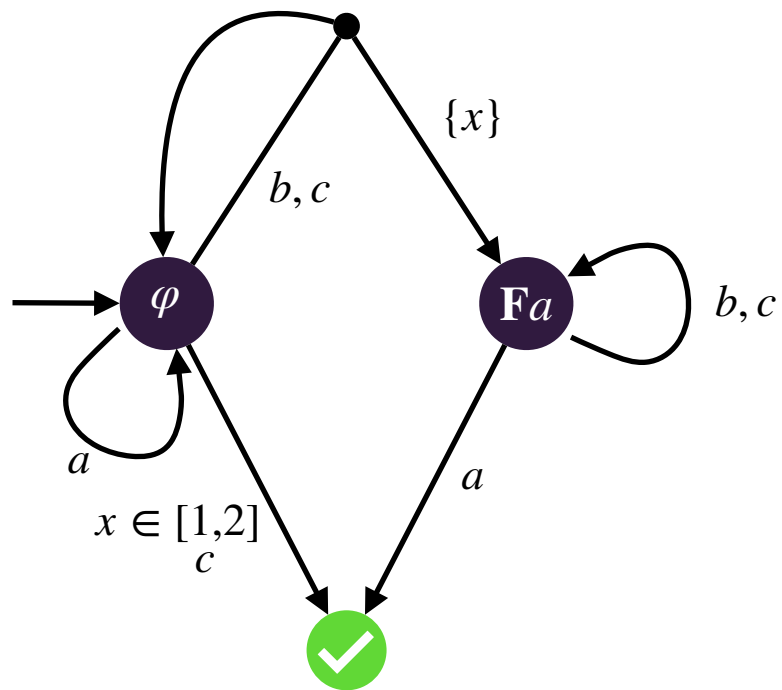
$$\Sigma = \{a, b, c\}$$



MTL to 1ATA

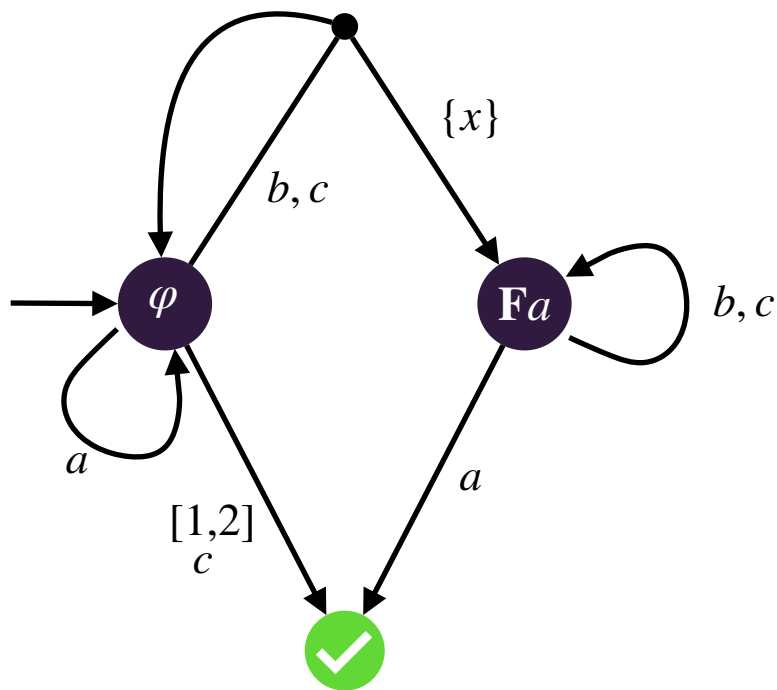
$$\varphi = (\mathbf{F}a)\mathbf{U}_{[1,2]}c$$

$$\Sigma = \{a, b, c\}$$



Deactivation operator

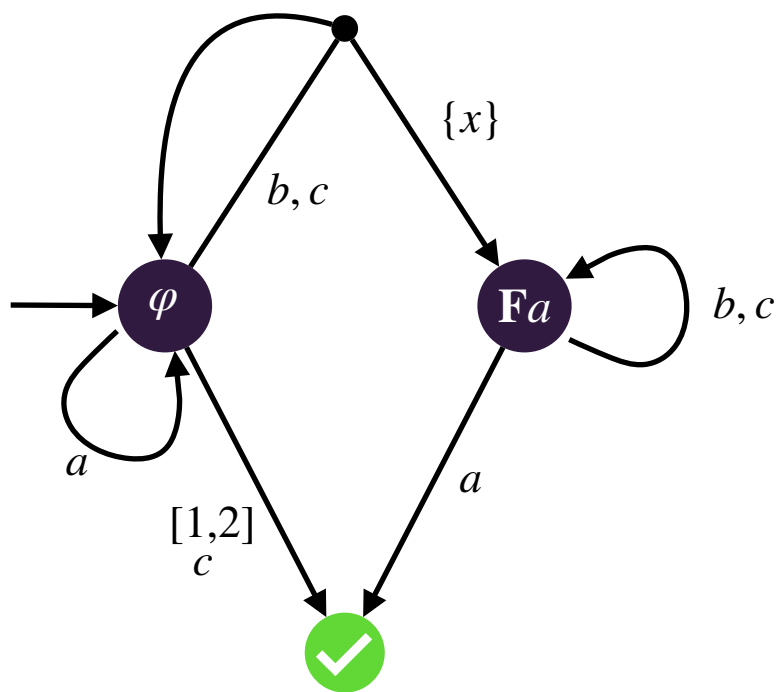
$$w = (0.5, b)(0.3, b)(1, c)(1, a)$$



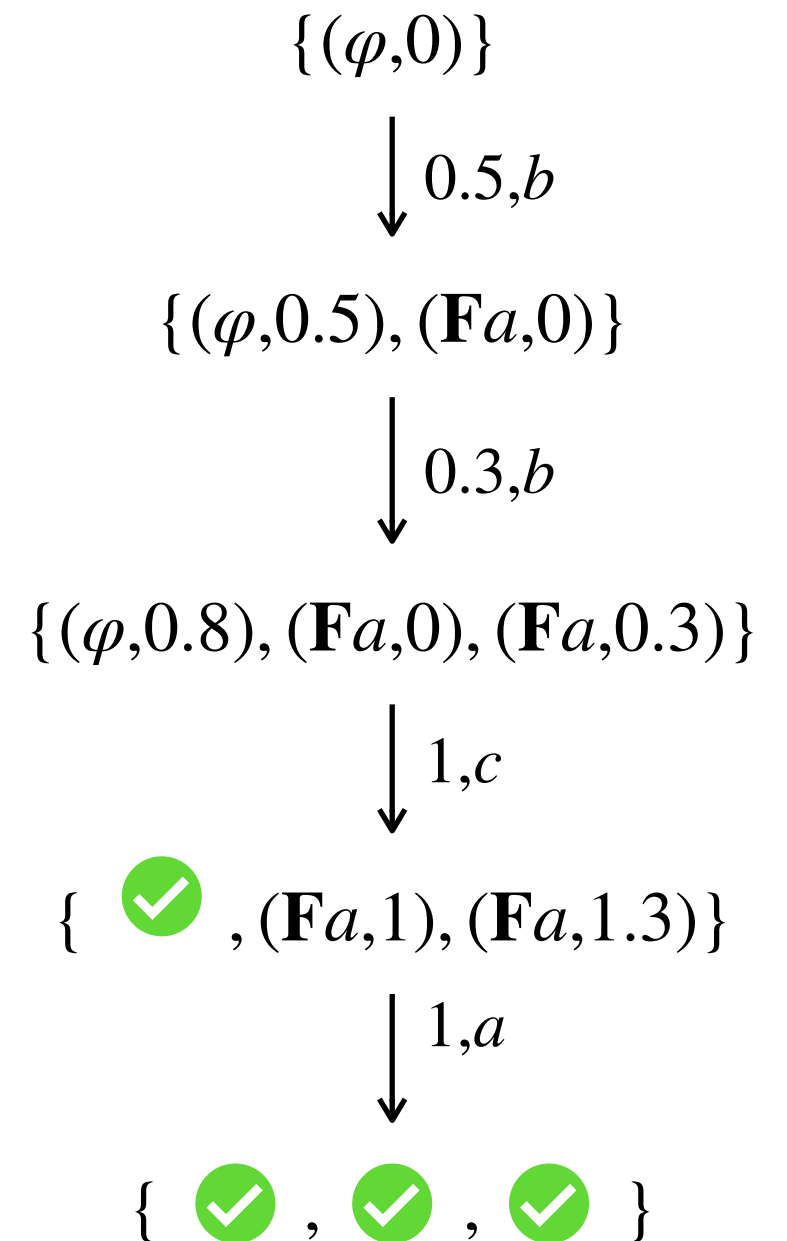
$$\varphi = (\mathbf{F}a)\mathbf{U}_{[1,2]}c$$

Deactivation operator

$$w = (0.5, b)(0.3, b)(1, c)(1, a)$$

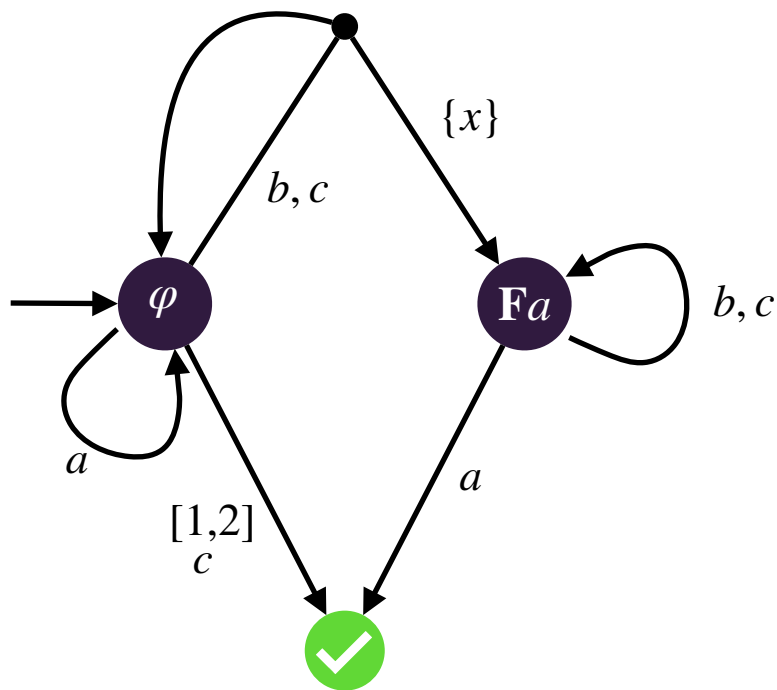


$$\varphi = (\mathbf{F}a)U_{[1,2]}c$$

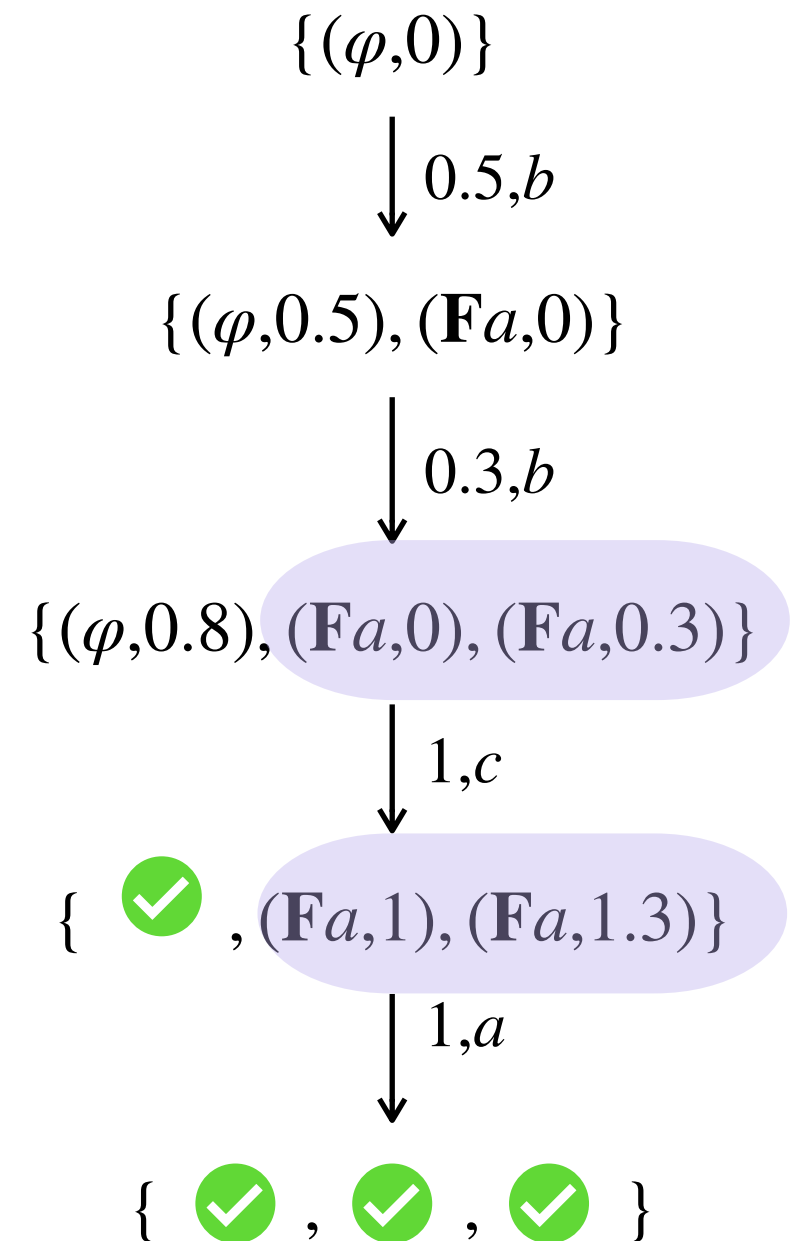


Deactivation operator

$$w = (0.5, b)(0.3, b)(1, c)(1, a)$$

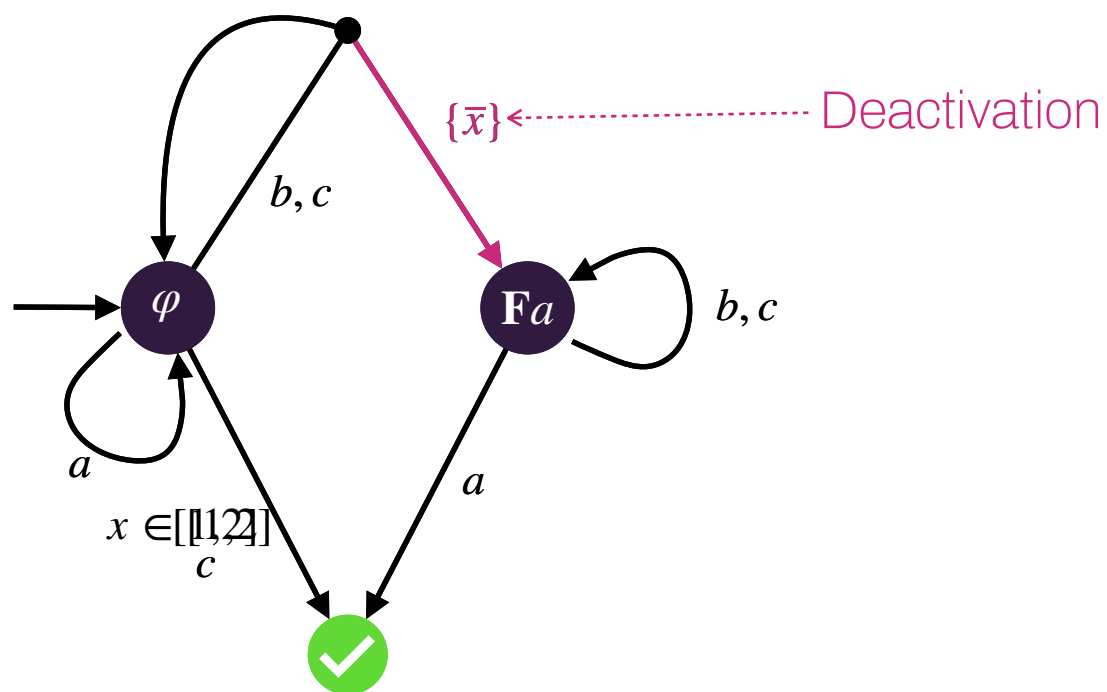


$$\varphi = (\mathbf{F}a)U_{[1,2]}c$$

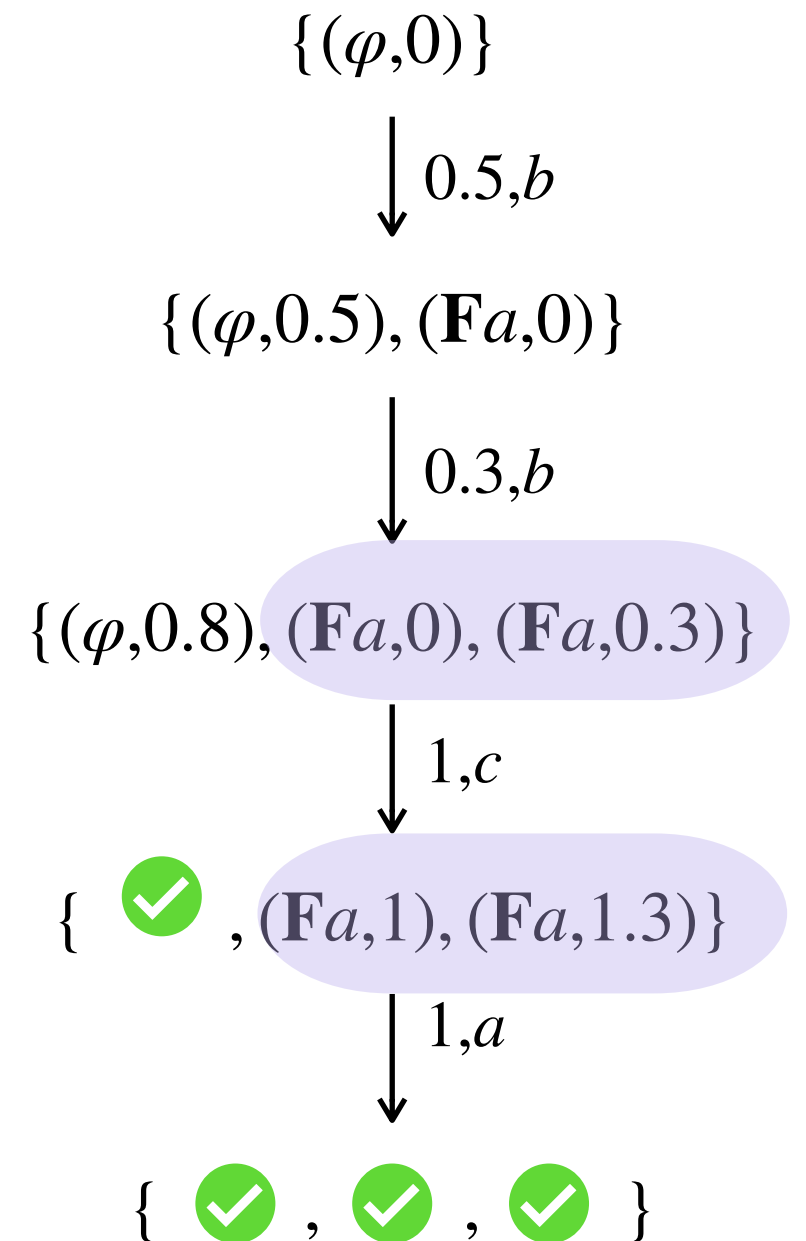


Deactivation operator

$$w = (0.5, b)(0.3, b)(1, c)(1, a)$$

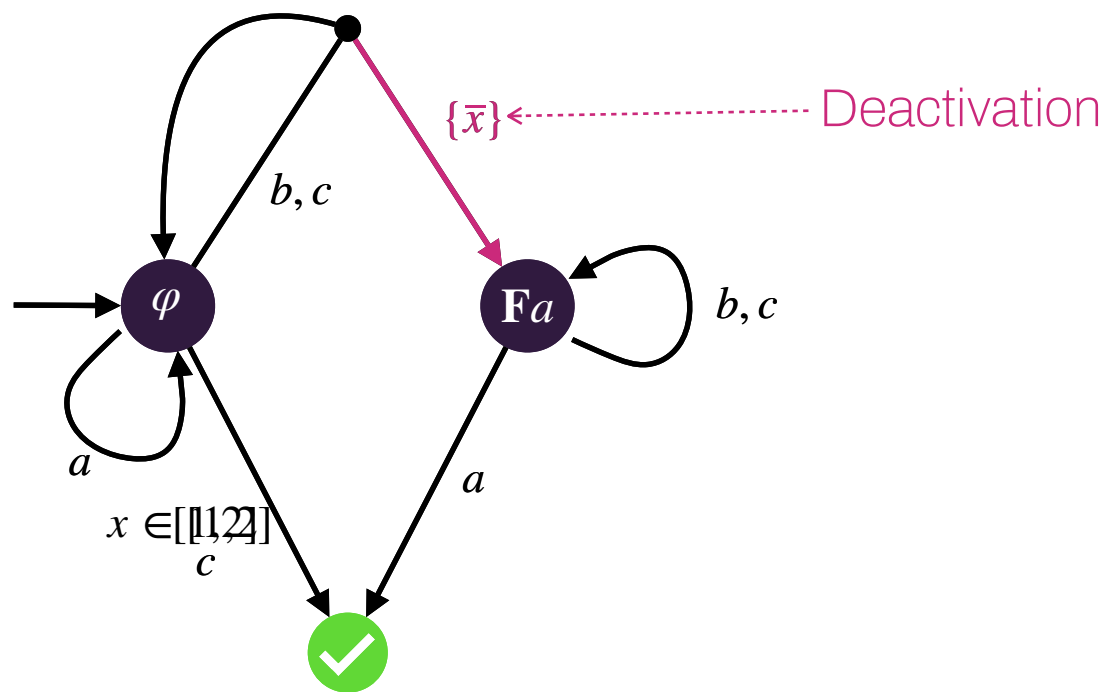


$$\varphi = (\mathbf{F}a)U_{[1,2]}c$$

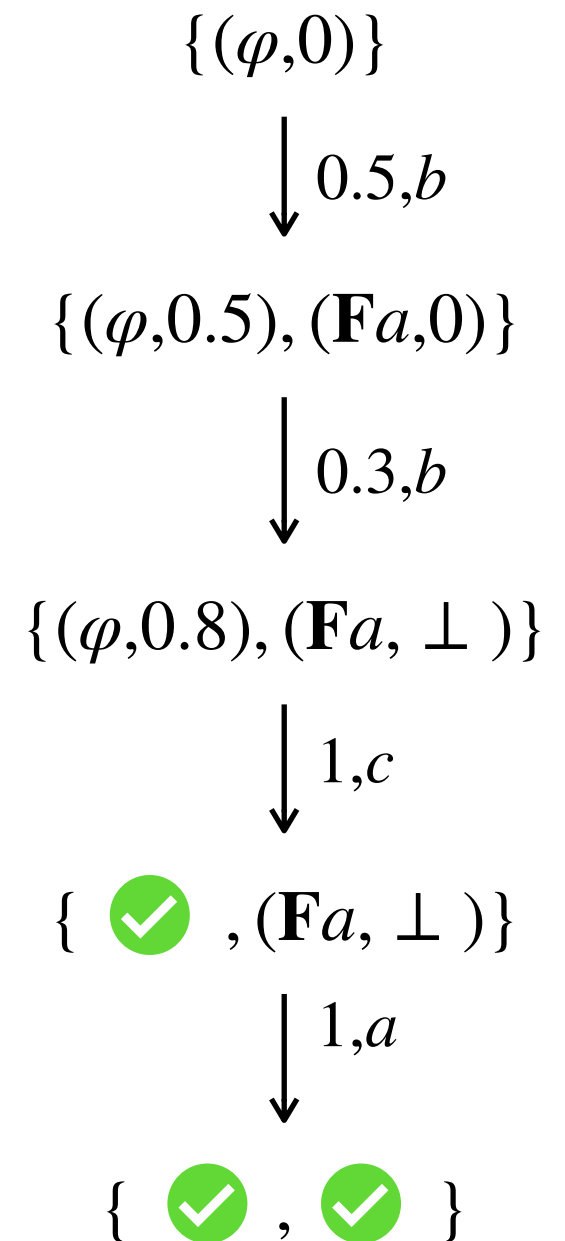


Deactivation operator

$$w = (0.5, b)(0.3, b)(1, c)(1, a)$$



$$\varphi = (\mathbf{F}a)U_{[1,2]}c$$



Zones

Zones

- ▶ Standard notion in TA verification

Zones

- ▶ Standard notion in TA verification
- ▶ Set of constraints that define (symbolically) the set of clock values that can be reached

Zones

- ▶ Standard notion in TA verification
- ▶ Set of constraints that define (symbolically) the set of clock values that can be reached
- ▶ They are all of the form:

Zones

- ▶ Standard notion in TA verification
- ▶ Set of constraints that define (symbolically) the set of clock values that can be reached
- ▶ They are all of the form:

$$x \leq c \quad \text{or} \quad x - y \leq c$$

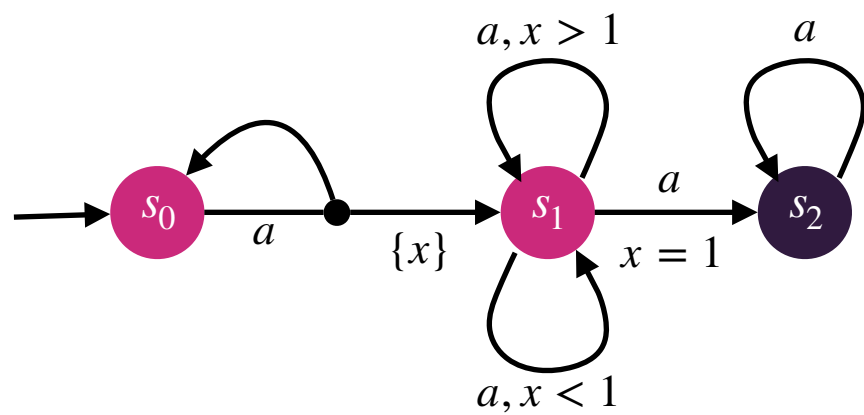
Zones

- ▶ Standard notion in TA verification
- ▶ Set of constraints that define (symbolically) the set of clock values that can be reached
- ▶ They are all of the form:

$$x \leq c \quad \text{or} \quad x - y \leq c$$

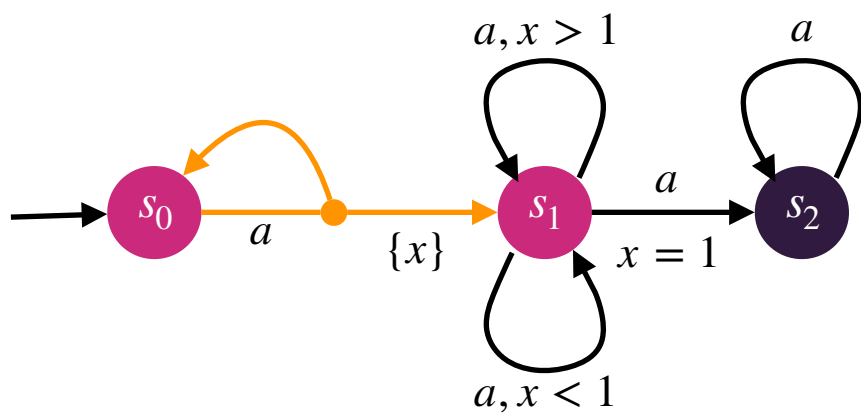
See our survey

Zone graph computation



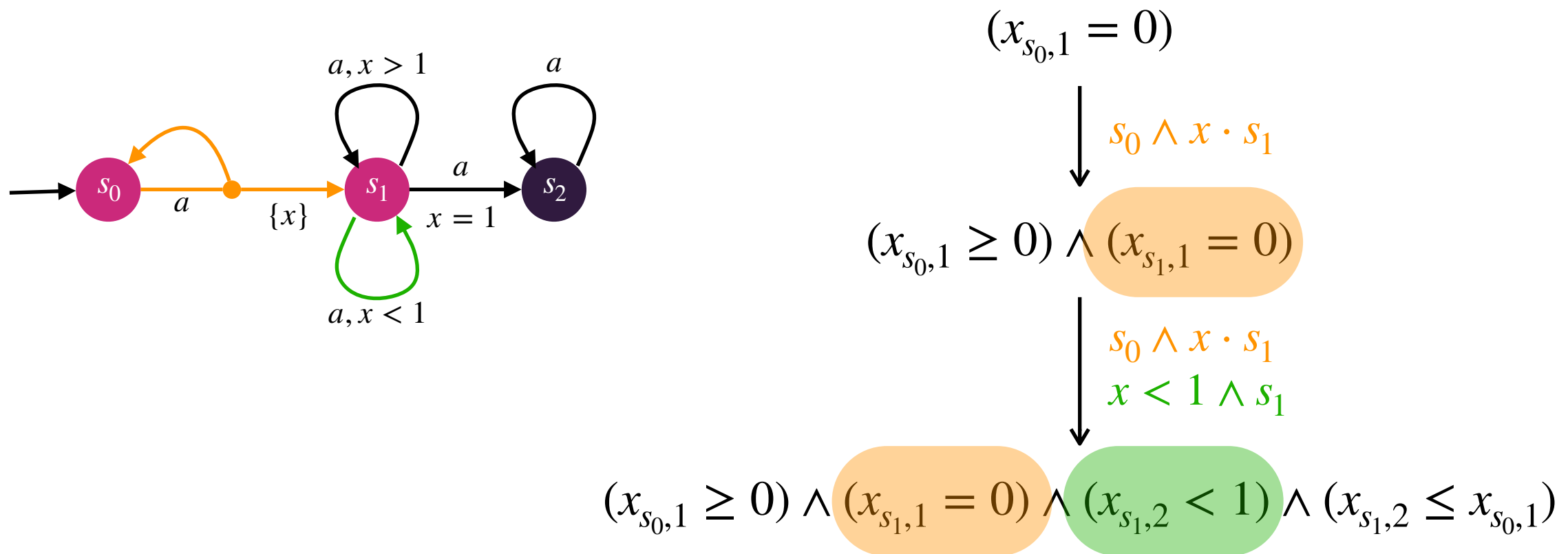
$$(x_{s_0,1} = 0)$$

Zone graph computation

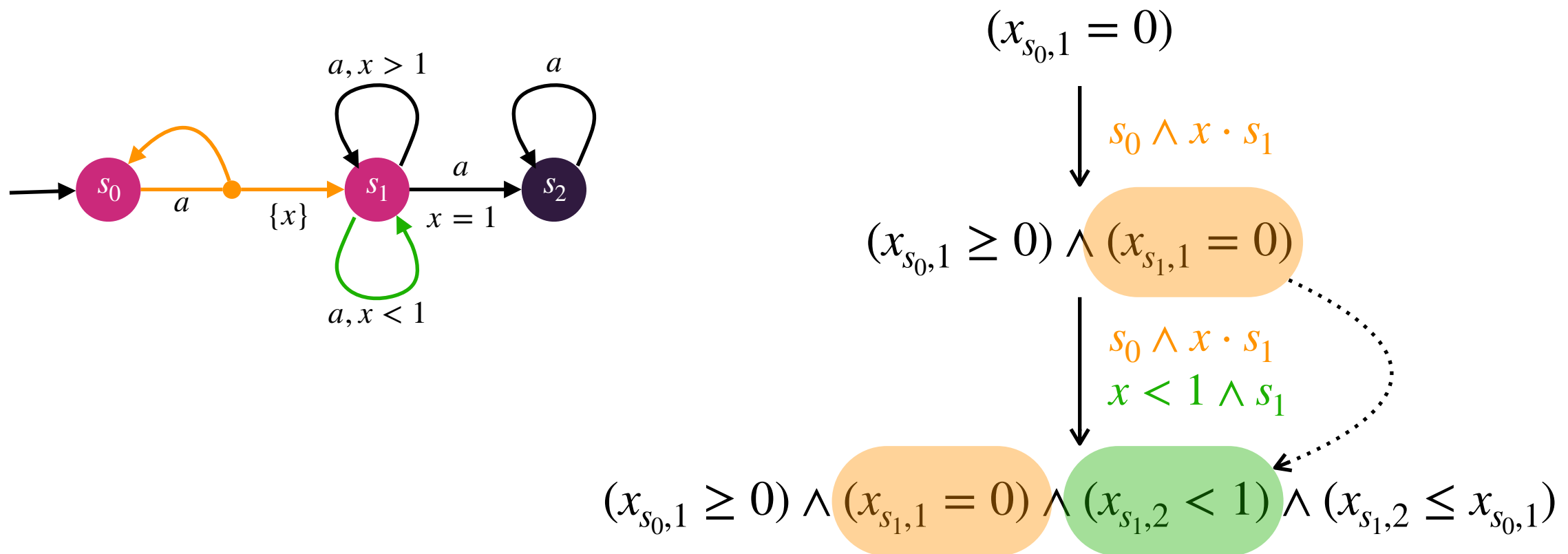


$$\begin{array}{c}
 (x_{s_0,1} = 0) \\
 \downarrow s_0 \wedge x \cdot s_1 \\
 (x_{s_0,1} \geq 0) \wedge (x_{s_1,1} = 0)
 \end{array}$$

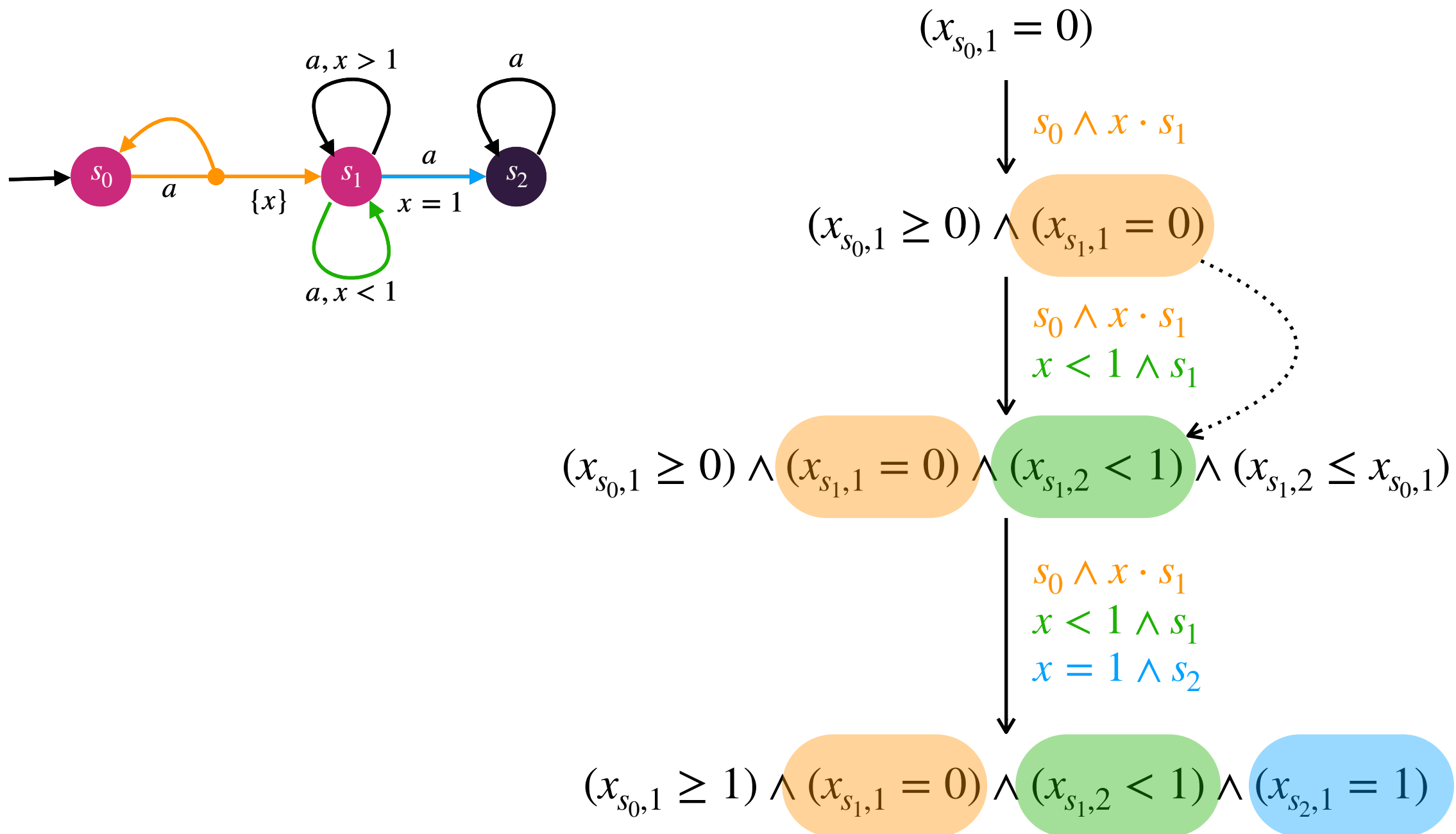
Zone graph computation



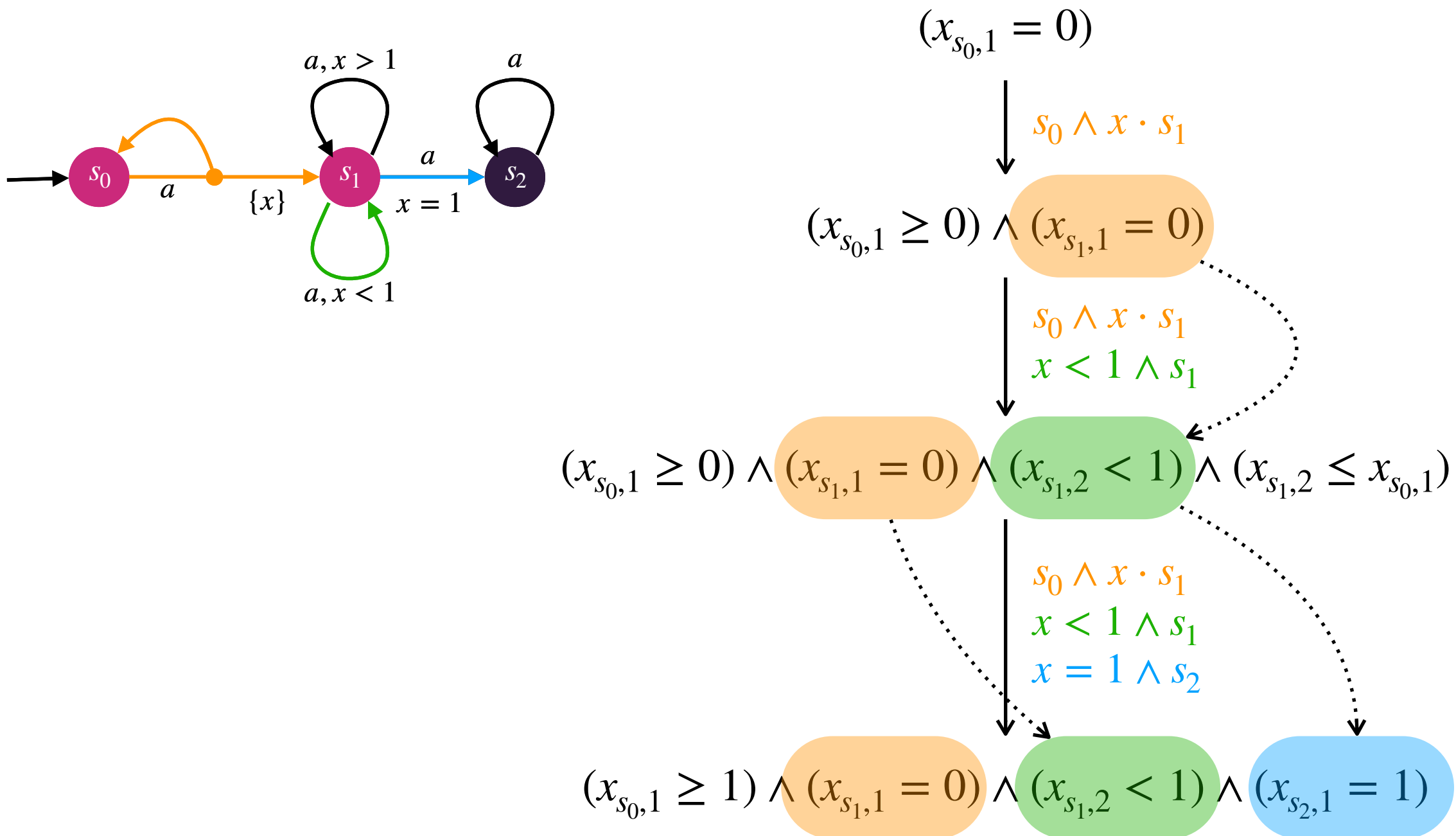
Zone graph computation



Zone graph computation



Zone graph computation



Algorithm

Explore the zone graph until:

- ▶ Some accepting zone is found: non-empty
- ▶ All reachable zones have a non-accepting state: empty

Algorithm

Explore the zone graph until:

- ▶ Some accepting zone is found: non-empty
- ▶ All reachable zones have a non-accepting state: empty

In general the zone graph could be infinite!

Algorithm

Explore the zone graph until:

- ▶ Some accepting zone is found: non-empty
- ▶ All reachable zones have a non-accepting state: empty

In general the zone graph could be infinite!

- ▶ We use an entailment relation $\sqsubseteq$ such that:
 - If $Z \sqsubseteq Z'$ and  is reachable from Z' , then it is also reachable from Z
 - **Pruning** using $\sqsubseteq$ ensures a finite graph

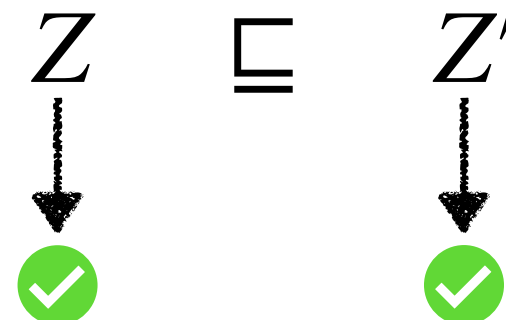
Algorithm

Explore the zone graph until:

- ▶ Some accepting zone is found: non-empty
- ▶ All reachable zones have a non-accepting state: empty

In general the zone graph could be infinite!

- ▶ We use an entailment relation $\sqsubseteq$ such that:
 - If $Z \sqsubseteq Z'$ and  is reachable from Z' , then it is also reachable from Z
 - **Pruning** using $\sqsubseteq$ ensures a finite graph



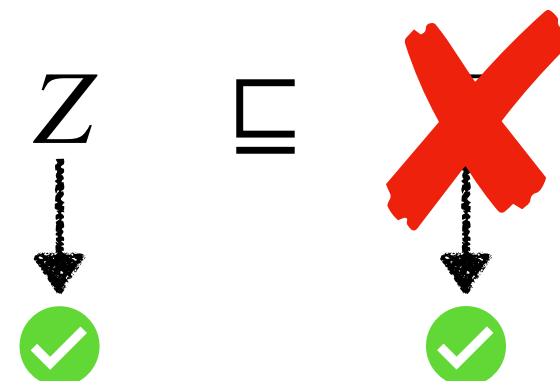
Algorithm

Explore the zone graph until:

- ▶ Some accepting zone is found: non-empty
- ▶ All reachable zones have a non-accepting state: empty

In general the zone graph could be infinite!

- ▶ We use an entailment relation $\sqsubseteq$ such that:
 - If $Z \sqsubseteq Z'$ and  is reachable from Z' , then it is also reachable from Z
 - **Pruning** using $\sqsubseteq$ ensures a finite graph



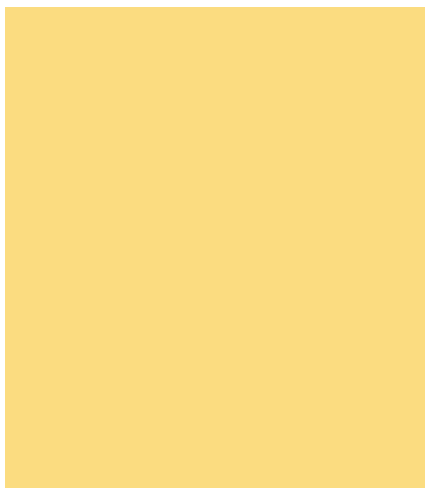
Entailment relation on zones

- ▶ $Z \sqsubseteq Z'$ if for each $\gamma' \in Z'$, there is $\gamma \in Z$ and $\gamma'' \subseteq \gamma'$ s.t. $\gamma'' \cong \gamma$

Entailment relation on zones

- ▶ $Z \sqsubseteq Z'$ if for each $\gamma' \in Z'$, there is $\gamma \in Z$ and $\gamma'' \subseteq \gamma'$ s.t. $\gamma'' \cong \gamma$

Z'

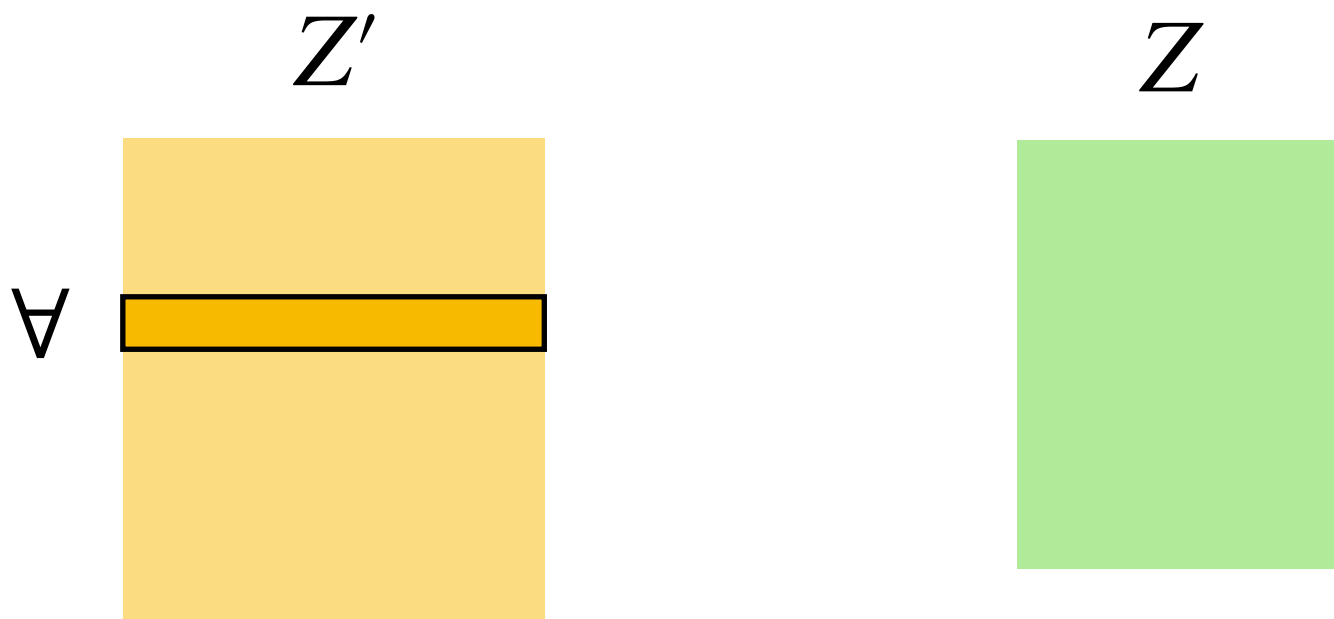


Z



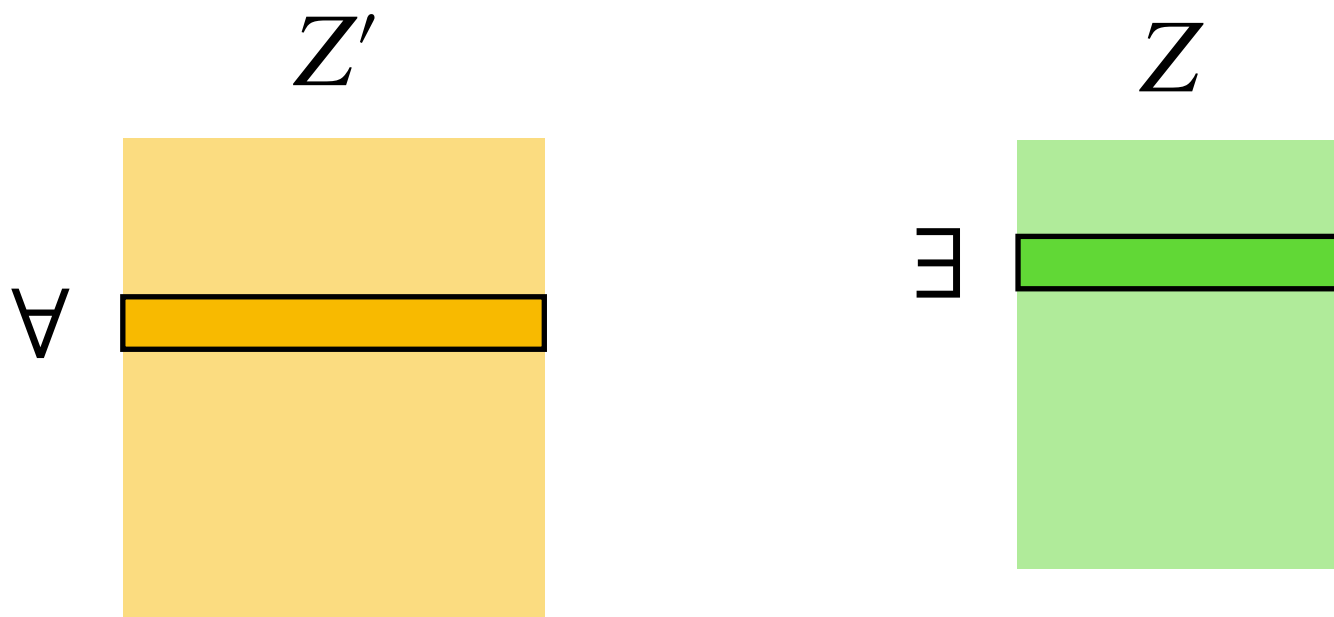
Entailment relation on zones

- ▶ $Z \sqsubseteq Z'$ if for each $\gamma' \in Z'$, there is $\gamma \in Z$ and $\gamma'' \subseteq \gamma'$ s.t. $\gamma'' \cong \gamma$



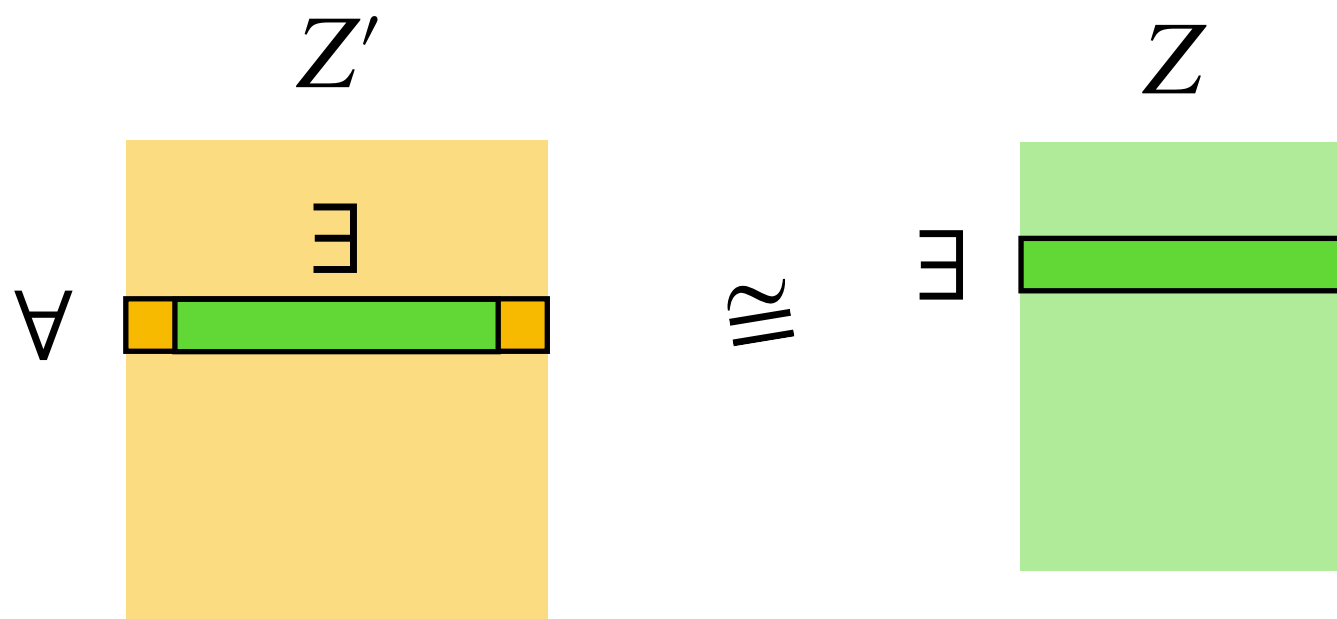
Entailment relation on zones

- ▶ $Z \sqsubseteq Z'$ if for each $\gamma' \in Z'$, there is $\gamma \in Z$ and $\gamma'' \subseteq \gamma'$ s.t. $\gamma'' \cong \gamma$



Entailment relation on zones

- ▶ $Z \sqsubseteq Z'$ if for each $\gamma' \in Z'$, there is $\gamma \in Z$ and $\gamma'' \subseteq \gamma'$ s.t. $\gamma'' \cong \gamma$

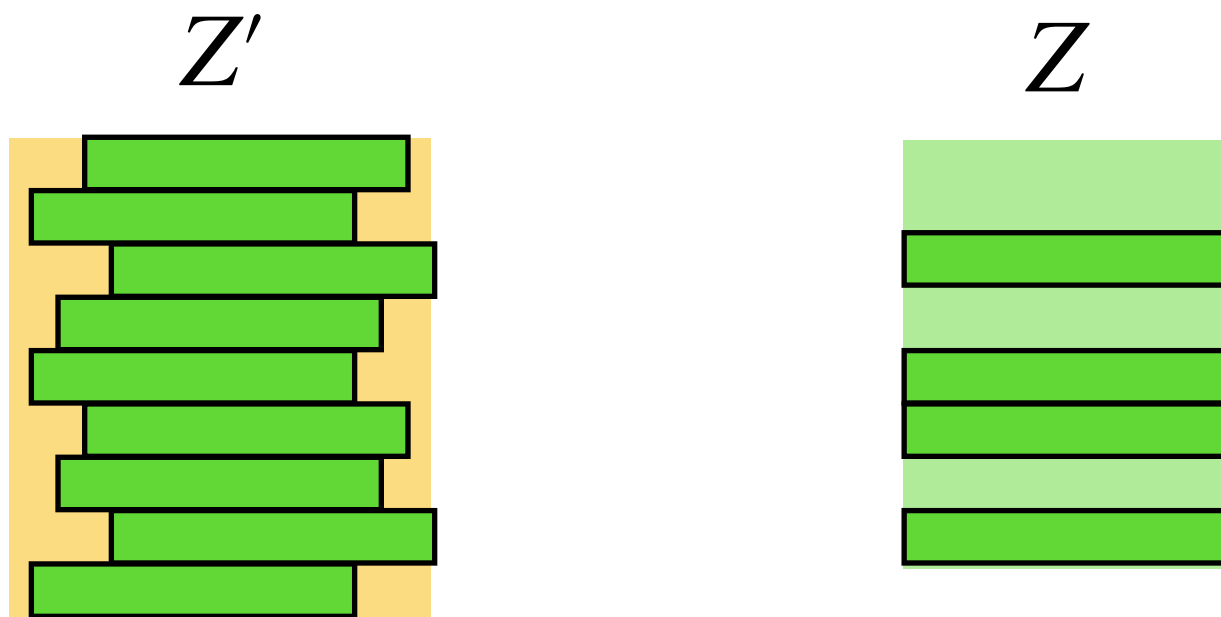


Entailment relation on zones

- ▶ $Z \sqsubseteq Z'$ if for each $\gamma' \in Z'$, there is $\gamma \in Z$ and $\gamma'' \subseteq \gamma'$ s.t. $\gamma'' \cong \gamma$

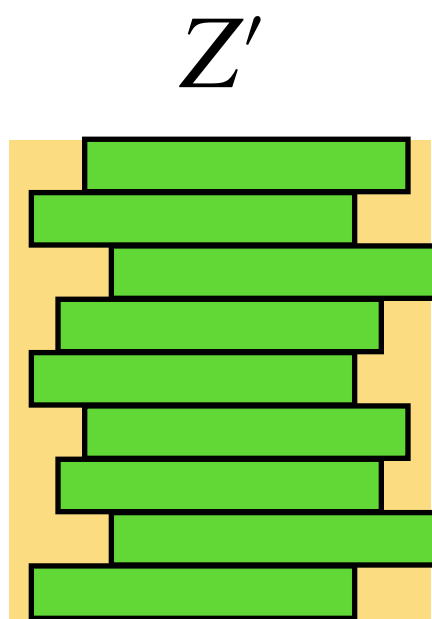
Entailment relation on zones

- ▶ $Z \sqsubseteq Z'$ if for each $\gamma' \in Z'$, there is $\gamma \in Z$ and $\gamma'' \subseteq \gamma'$ s.t. $\gamma'' \cong \gamma$



Entailment relation on zones

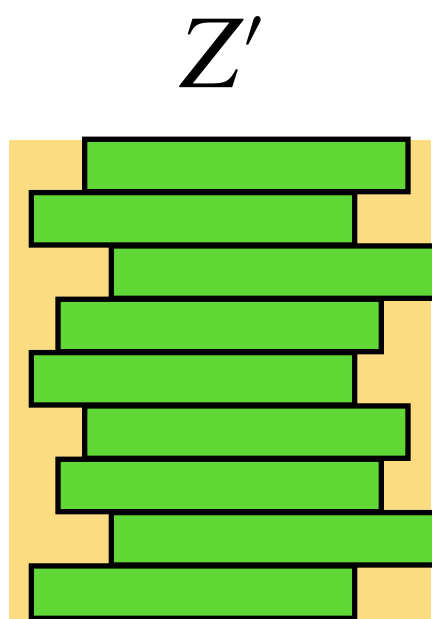
- ▶ $Z \sqsubseteq Z'$ if for each $\gamma' \in Z'$, there is $\gamma \in Z$ and $\gamma'' \subseteq \gamma'$ s.t. $\gamma'' \cong \gamma$



- ▶ Different size zones
- ▶ Multiple witness subsets

Entailment relation on zones

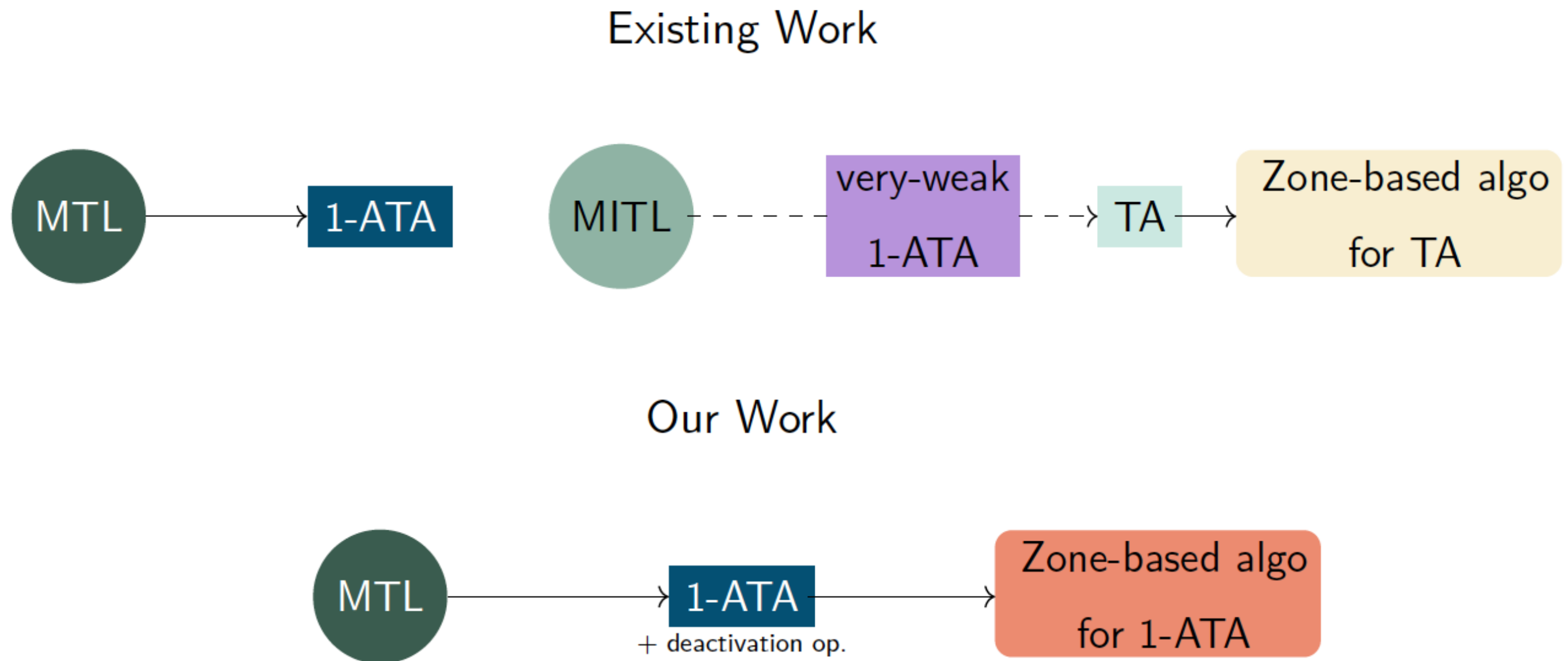
- ▶ $Z \sqsubseteq Z'$ if for each $\gamma' \in Z'$, there is $\gamma \in Z$ and $\gamma'' \subseteq \gamma'$ s.t. $\gamma'' \cong \gamma$



- ▶ Different size zones
- ▶ Multiple witness subsets

- ▶ Exponential algorithm
- ▶ NP-hardness result for the $\not\sqsubseteq$ check

Conclusion



Perspectives — Ongoing

Perspectives — Ongoing



- ▶ Use of duality clock vs timer to improve verification of MITL
 - Define obligations and reduction rules
 - Define regions
 - Can handle full MITL

Perspectives — Ongoing



- ▶ Use of duality clock vs timer to improve verification of MITL
 - Define obligations and reduction rules
 - Define regions
 - Can handle full MITL
- ▶ Has just been accepted at FSTTCS'26