Safety Metric Temporal Logic is Fully Decidable

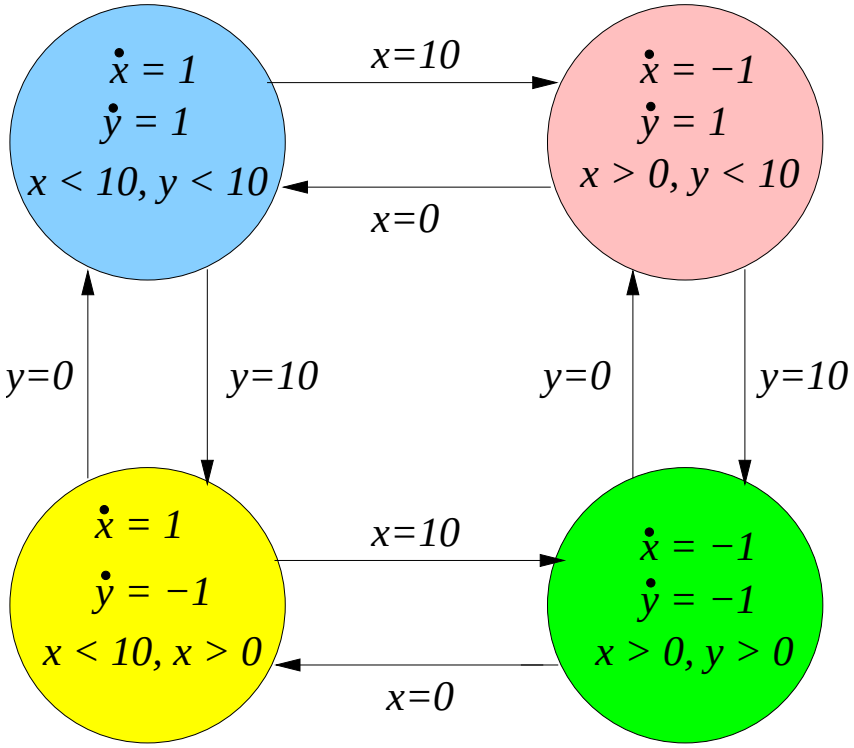
James Worrell
Oxford University Computing Laboratory

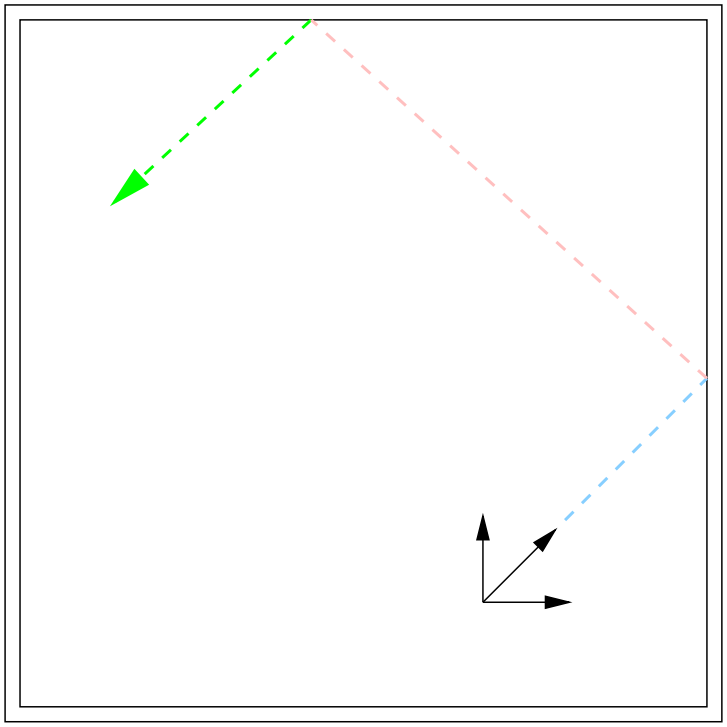
(joint work with Joël Ouaknine)

Concurrency, Nov 2005

Real-time Models

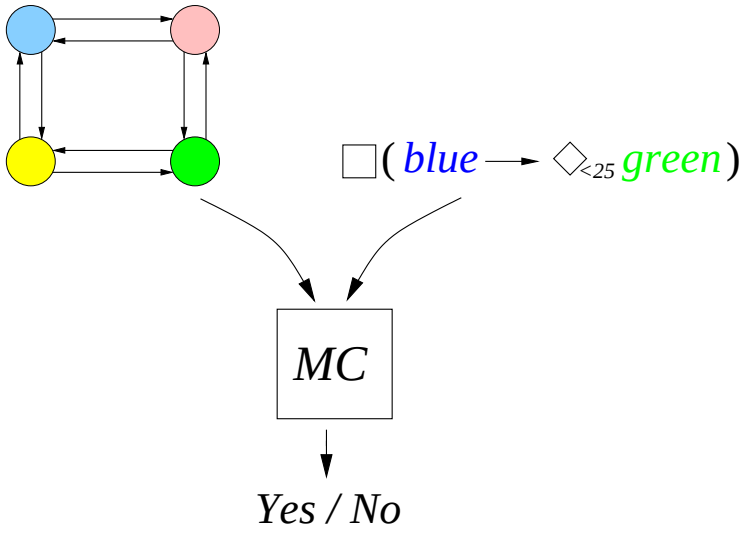
– Rectangular hybrid automata (Alur, Courcoubetis, Henzinger):





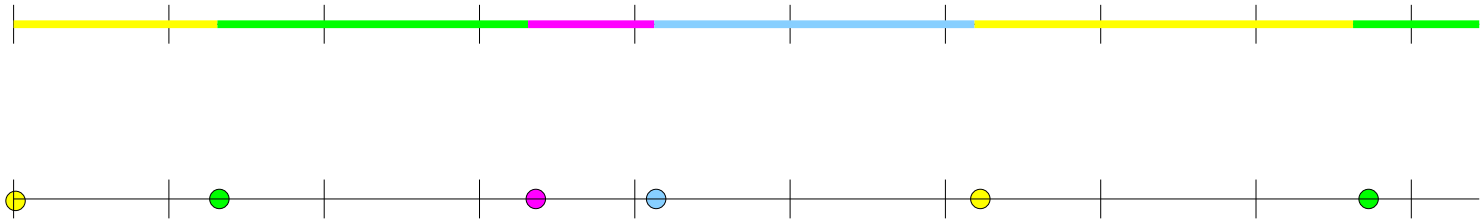
Model Checking

- Properties specified using **Metric Temporal Logic** (Pnueli, de Roever, Koymans)
  - Response: $\Box(blue \rightarrow \Diamond_{\leq 25}\ green)$
- Model checking algorithm:



Trajectories

A run of the billiard ball automaton yields a **trajectory**. We adopt a semantics of **timed words** where only events (corresponding to changes in state) are recorded.



Unlike in your **local pub**, this billiard ball is **deterministic**. A more realistic model could be obtained by writing, e.g., $\dot{x} \in [1-\varepsilon, 1+\varepsilon]$.

In general an automaton $\mathcal{A}$ yields a **language** of timed words $L(\mathcal{A})$. The **model checking problem** $\mathcal{A} \models \varphi$ asks whether every timed word $w \in L(\mathcal{A})$ satisfies φ.

Timed Words

- A **timed word** is an infinite sequence of **timed events**:

$$\langle (t_0, a_0), (t_1, a_1), (t_2, a_2), (t_3, a_3), \ldots \rangle$$

Timed Words

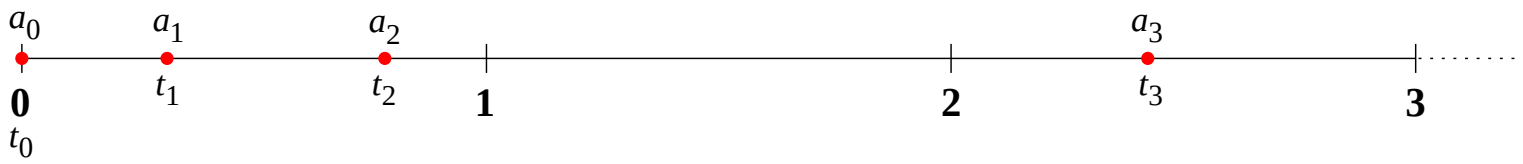
- A **timed word** is an infinite sequence of **timed events**:

$$\langle (t_0, a_0), (t_1, a_1), (t_2, a_2), (t_3, a_3), \ldots \rangle$$

Timed Words

- A **timed word** is an infinite sequence of **timed events**:

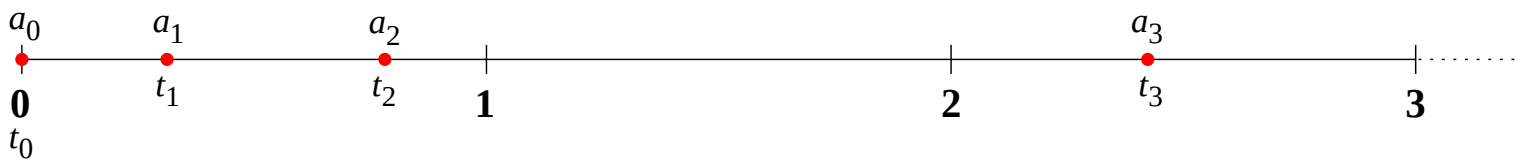
$$\langle (t_0, a_0), (t_1, a_1), (t_2, a_2), (t_3, a_3), \ldots \rangle$$



Timed Words

- A **timed word** is an infinite sequence of **timed events**:

$$\langle (t_0, a_0), (t_1, a_1), (t_2, a_2), (t_3, a_3), \ldots \rangle$$

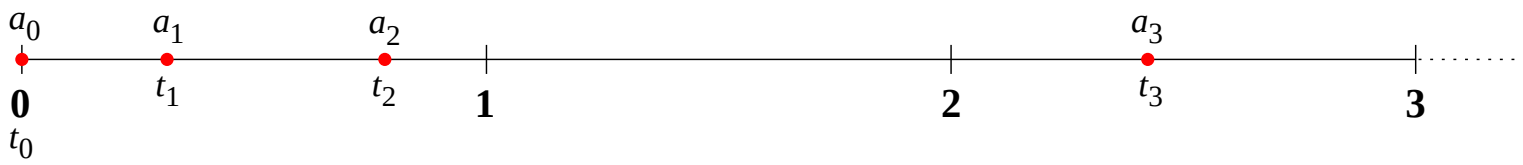


We only consider **non-Zeno** words.

Timed Words

- A **timed word** is an infinite sequence of **timed events**:

$$\langle (t_0, a_0), (t_1, a_1), (t_2, a_2), (t_3, a_3), \ldots \rangle$$



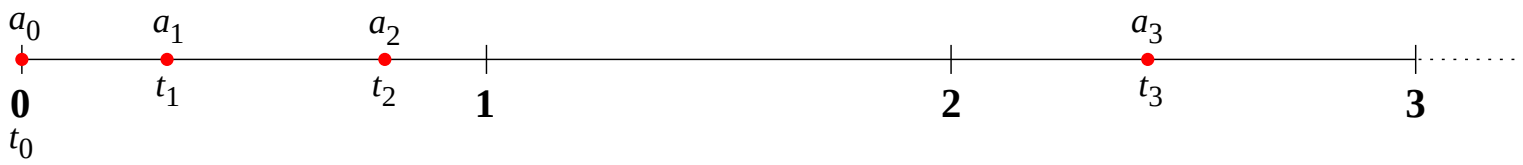
We only consider **non-Zeno** words.

– Only a finite number of events in each fixed time interval.

Timed Words

- A **timed word** is an infinite sequence of **timed events**:

$$\langle (t_0, a_0), (t_1, a_1), (t_2, a_2), (t_3, a_3), \ldots \rangle$$



We only consider **non-Zeno** words.

- Only a finite number of events in each fixed time interval.
- Systems are **finitely variable**.

Order vs Metric Specifications

- A standard (untimed) response property:
  - "Every request is eventually granted."

$$\Box(request \longrightarrow \Diamond grant)$$

Order vs Metric Specifications

- A standard (untimed) response property:
  - "Every request is eventually granted."

$$\Box(request \longrightarrow \Diamond grant)$$

- A *time-bounded* response property:
  - "Every request is eventually granted *within 5 time units*."

$$\Box(request \longrightarrow \Diamond_{[0,5]} grant)$$

Metric Temporal Logic

- **Metric Temporal Logic (MTL).**
  - MTL = Linear temporal logic + constrained modalities
- $\varphi ::= a \mid \top \mid \varphi \wedge \varphi \mid \neg\varphi \mid \Box_I \varphi \mid \Diamond_I \varphi \mid \varphi \, \mathcal{U}_I \, \varphi \mid ...$
  - $\Box_I \varphi$: All events occurring in the time interval I satisfy φ.
  - $\Diamond_I \varphi$: There exists an event in the time interval I that satisfies φ.

Point-based Semantics for MTL

- The relationship $w \models \varphi$ is defined inductively as in LTL.

Point-based Semantics for MTL

- The relationship $w \models \varphi$ is defined inductively as in LTL.
  - For example, let φ be:

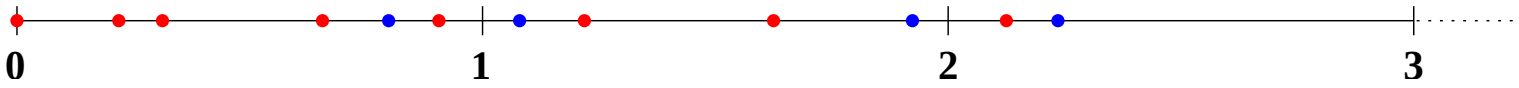
$$\Box(request \longrightarrow \Diamond_{[0,1]} grant)$$

Point-based Semantics for MTL

- The relationship $w \models \varphi$ is defined inductively as in LTL.
  - For example, let φ be:

$$\Box(\mathit{request} \longrightarrow \Diamond_{[0,1]} \mathit{grant})$$

  - Let w be:

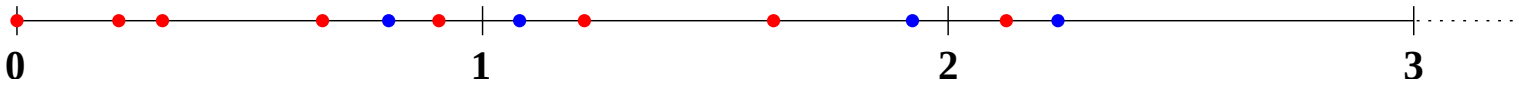


Point-based Semantics for MTL

- The relationship $w \models \varphi$ is defined inductively as in LTL.
  - For example, let φ be:

$$\Box(request \longrightarrow \Diamond_{[0,1]} grant)$$

  - Let w be:



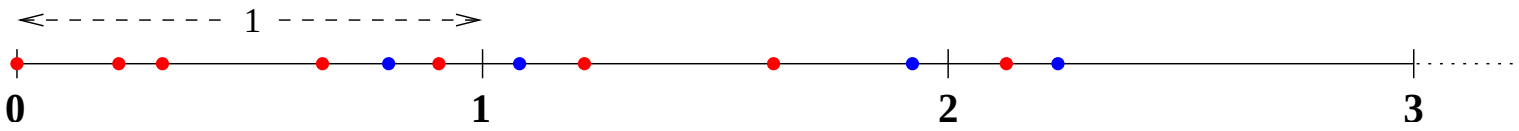
Does $w \models \varphi$?

Point-based Semantics for MTL

- The relationship $w \models \varphi$ is defined inductively as in LTL.
  - For example, let φ be:

$$\Box(\mathit{request} \longrightarrow \Diamond_{[0,1]} \mathit{grant})$$

  - Let w be:



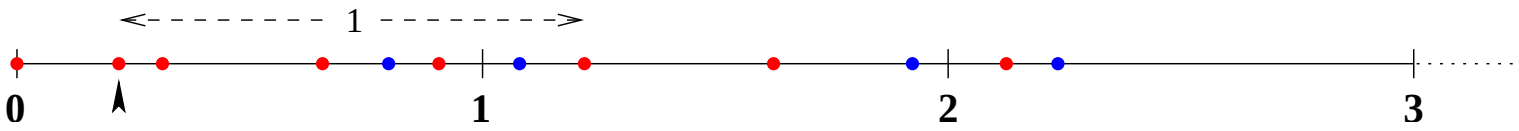
Does $w \models \varphi$?

Point-based Semantics for MTL

- The relationship $w \models \varphi$ is defined inductively as in LTL.
  - For example, let φ be:

$$\Box(request \longrightarrow \Diamond_{[0,1]} grant)$$

  - Let w be:



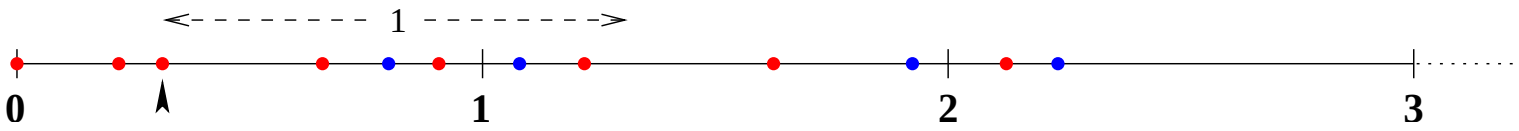
Does $w \models \varphi$?

Point-based Semantics for MTL

- The relationship $w \models \varphi$ is defined inductively as in LTL.
  - For example, let φ be:

$$\Box(request \longrightarrow \Diamond_{[0,1]} grant)$$

  - Let w be:



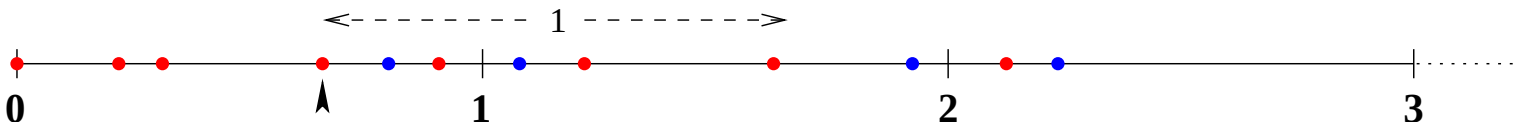
Does $w \models \varphi$?

Point-based Semantics for MTL

- The relationship $w \models \varphi$ is defined inductively as in LTL.
  - For example, let φ be:

$$\Box(request \longrightarrow \Diamond_{[0,1]} grant)$$

  - Let w be:



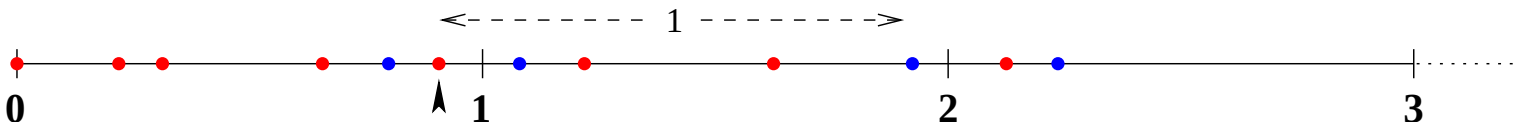
Does $w \models \varphi$?

Point-based Semantics for MTL

- The relationship $w \models \varphi$ is defined inductively as in LTL.
  - For example, let φ be:

$$\Box(\mathit{request} \longrightarrow \Diamond_{[0,1]} \mathit{grant})$$

  - Let w be:



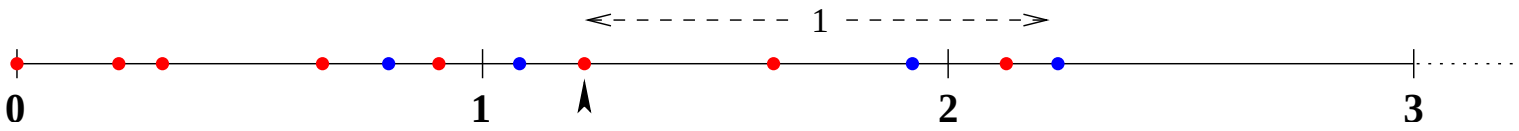
Does $w \models \varphi$?

Point-based Semantics for MTL

- The relationship $w \models \varphi$ is defined inductively as in LTL.
  - For example, let φ be:

$$\Box(\mathit{request} \longrightarrow \Diamond_{[0,1]} \mathit{grant})$$

  - Let w be:



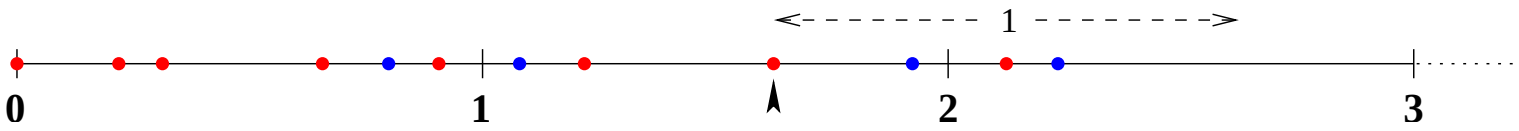
Does $w \models \varphi$?

Point-based Semantics for MTL

- The relationship $w \models \varphi$ is defined inductively as in LTL.
  - For example, let φ be:

$$\Box(request \longrightarrow \Diamond_{[0,1]} grant)$$

  - Let w be:



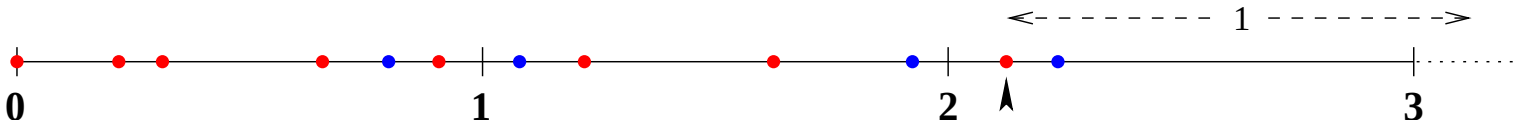
Does $w \models \varphi$?

Point-based Semantics for MTL

- The relationship $w \models \varphi$ is defined inductively as in LTL.
  - For example, let φ be:

$$\Box(request \longrightarrow \Diamond_{[0,1]} grant)$$

  - Let w be:



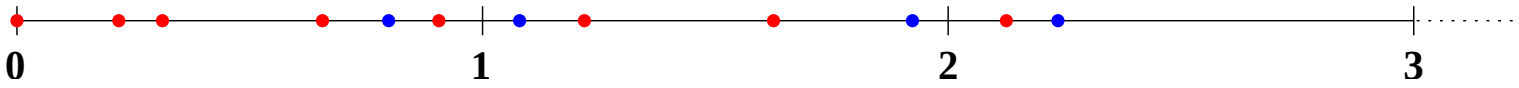
Does $w \models \varphi$?

Point-based Semantics for MTL

- The relationship $w \models \varphi$ is defined inductively as in LTL.
  - For example, let φ be:

$$\Box(\mathit{request} \longrightarrow \Diamond_{[0,1]} \mathit{grant})$$

  - Let w be:



Indeed, $w \models \varphi$.

More Examples

Alphabet $\Sigma = \{req_i, aq_i, rel_i : i = X, Y\}$ representing events of two processes X and Y that request, acquire, and release a lock.

- $\Box(req_X \rightarrow \Diamond_{[0,2]}(aq_X \wedge \Diamond_{=1} rel_X))$: whenever X requests the lock, it acquires the lock within 2 seconds and releases it exactly one second later.
- $\Box(aq_X \rightarrow rel_X \, \widetilde{\mathcal{U}}_{[0,3]} \, \neg aq_Y)$: Y cannot acquire the lock less than 3 seconds after X acquires the lock, unless X first releases it.

MTL is Undecidable

Thm. MTL is undecidable (OW05).

Proof. By reduction from **recurrent-state problem** for faulty Turing machines (channel machines with insertion errors). The reduction involves using the **liveness** formula $\Box\Diamond a$ (a occurs infinitely often) to capture recurrence.

In the rest of this talk we identify a **fully decidable** fragment of MTL.

Safety MTL

- **Safety MTL** includes those MTL formulas in which, in negation normal form, all eventualities $\Diamond_I$ or $\mathcal{U}_I$ are bounded.
  - $\Box(request \longrightarrow \Diamond_{[0,5]} grant)$ is a Safety MTL formula.
  - $\Box(request \longrightarrow \Diamond grant)$ is not a Safety MTL formula.
- If φ is Safety MTL formula and w is a non-Zeno timed word such that $w \not\models \varphi$, then w has a **finite bad prefix** $w_0 w_1 \ldots w_n$ none of whose extensions satisfies φ.

Full Decidability

- **Satisfiability:** Given φ, does there exist a timed word w such that $w \models \varphi$?

Full Decidability

- **Satisfiability:** Given φ, does there exist a timed word w such that $w \models \varphi$?
- **Model checking:** Given φ and a timed automaton A, does $w \models \varphi$ for all $w \in L(A)$?

Full Decidability

- **Satisfiability:** Given φ, does there exist a timed word w such that $w \models \varphi$?
- **Model checking:** Given φ and a timed automaton A, does $w \models \varphi$ for all $w \in L(A)$?
- **Refinement:** Given φ_1 and φ_2, does $w \models \varphi_1$ imply $w \models \varphi_2$ for every timed word w ?

Automata Representation

Given φ, construct an equivalent **timed alternating automaton** $\mathcal{A}_\varphi$ with a *single* clock variable:

Automata Representation

Given φ, construct an equivalent **timed alternating automaton** $\mathcal{A}_\varphi$ with a *single* clock variable:

- States of $\mathcal{A}_\varphi$ are the modal subformulas of φ.

Automata Representation

Given φ, construct an equivalent **timed alternating automaton** $\mathcal{A}_\varphi$ with a *single* clock variable:

- States of $\mathcal{A}_\varphi$ are the modal subformulas of φ.
- Every state is accepting.

Automata Representation

Given φ, construct an equivalent **timed alternating automaton** $\mathcal{A}_\varphi$ with a *single* clock variable:

- States of $\mathcal{A}_\varphi$ are the modal subformulas of φ.
- Every state is accepting.
- A single clock is needed to enforce any single timing constraint.

Automata Representation

Given φ, construct an equivalent **timed alternating automaton** $\mathcal{A}_\varphi$ with a *single* clock variable:

- States of $\mathcal{A}_\varphi$ are the modal subformulas of φ.
- Every state is accepting.
- A single clock is needed to enforce any single timing constraint.
- The conjunctive branching of $\mathcal{A}_\varphi$ spawns a new computation thread (with a new fresh single clock) for every new timing requirement.

Automata Representation

Given φ, construct an equivalent **timed alternating automaton** $\mathcal{A}_\varphi$ with a *single* clock variable:

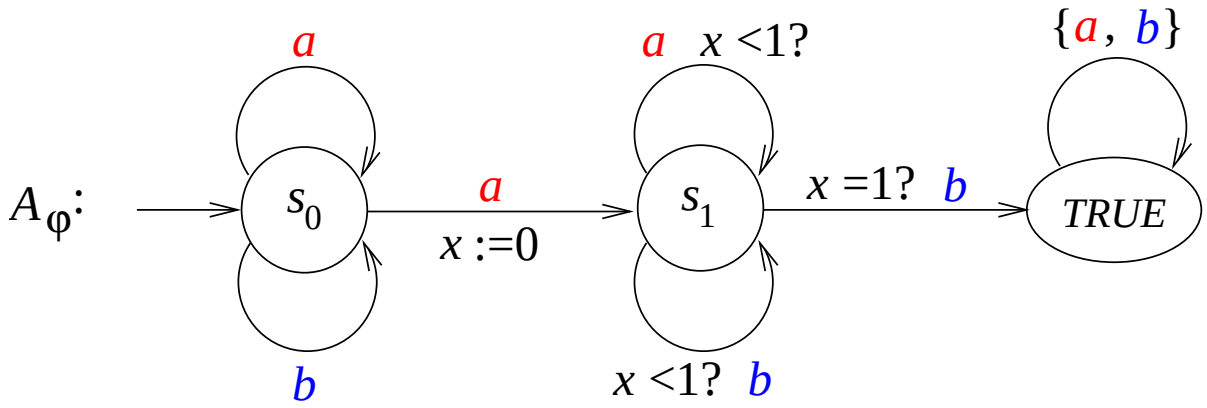
- States of $\mathcal{A}_\varphi$ are the modal subformulas of φ.
- Every state is accepting.
- A single clock is needed to enforce any single timing constraint.
- The conjunctive branching of $\mathcal{A}_\varphi$ spawns a new computation thread (with a new fresh single clock) for every new timing requirement.
- The language of $\mathcal{A}_\varphi$ consists of all non-Zeno words that satisfy φ.

Example: Punctual Response

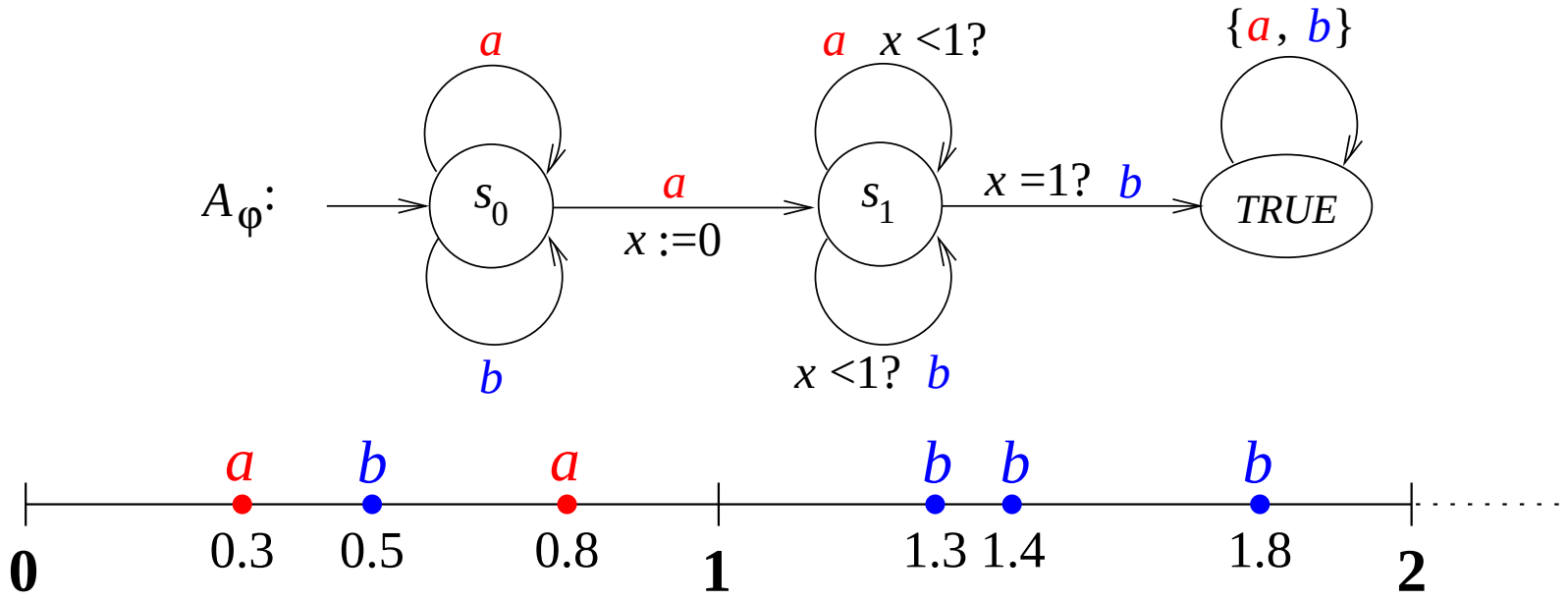
Punctual response: let φ be the formula $\Box(a \longrightarrow \Diamond_{=1} b)$.

- $s_0 : \Box(a \longrightarrow \Diamond_{=1} b)$. Initial state
- $s_1 : \Diamond_{=1} b$.
- All states are accepting.

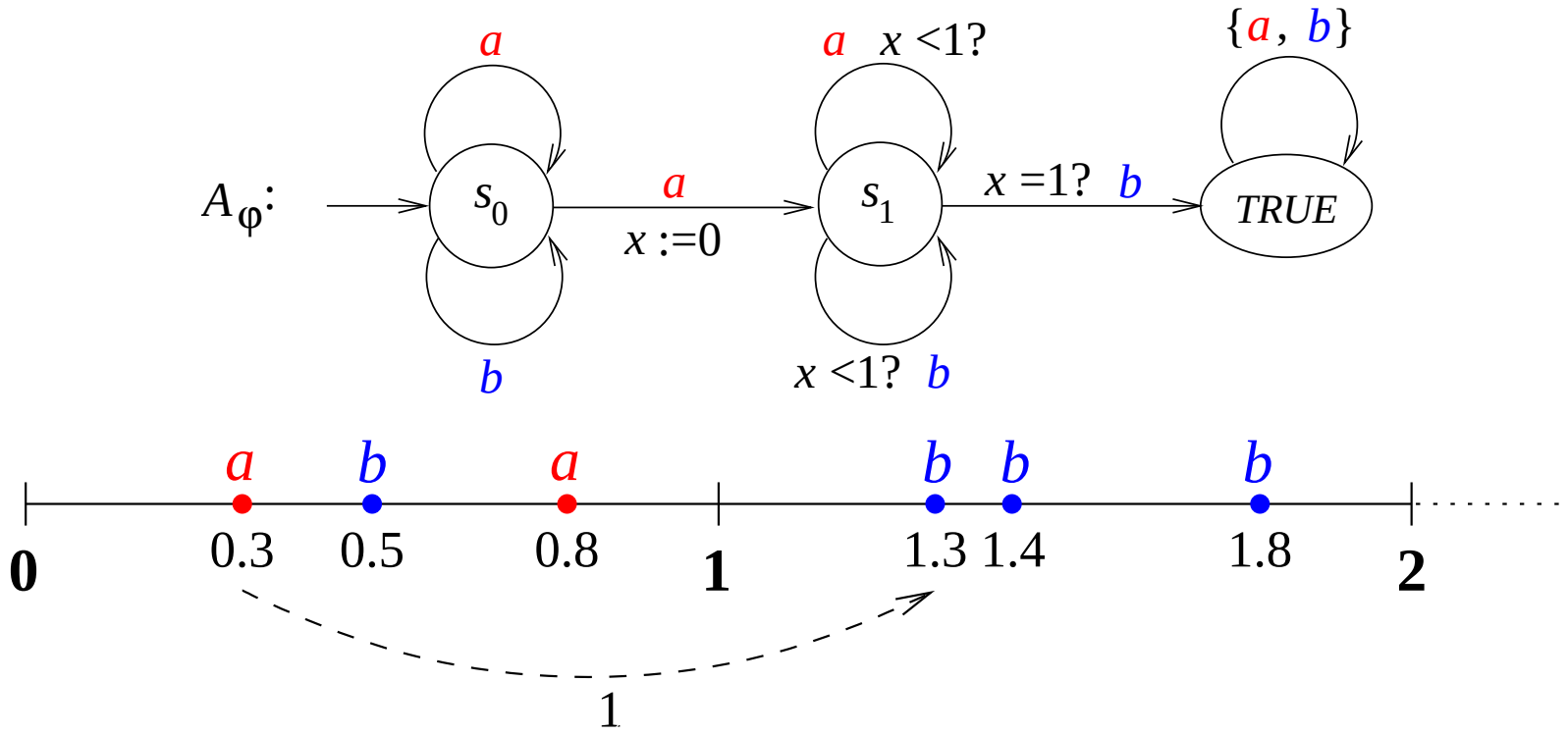
φ: $\Box(a \longrightarrow \Diamond_{=1} b)$



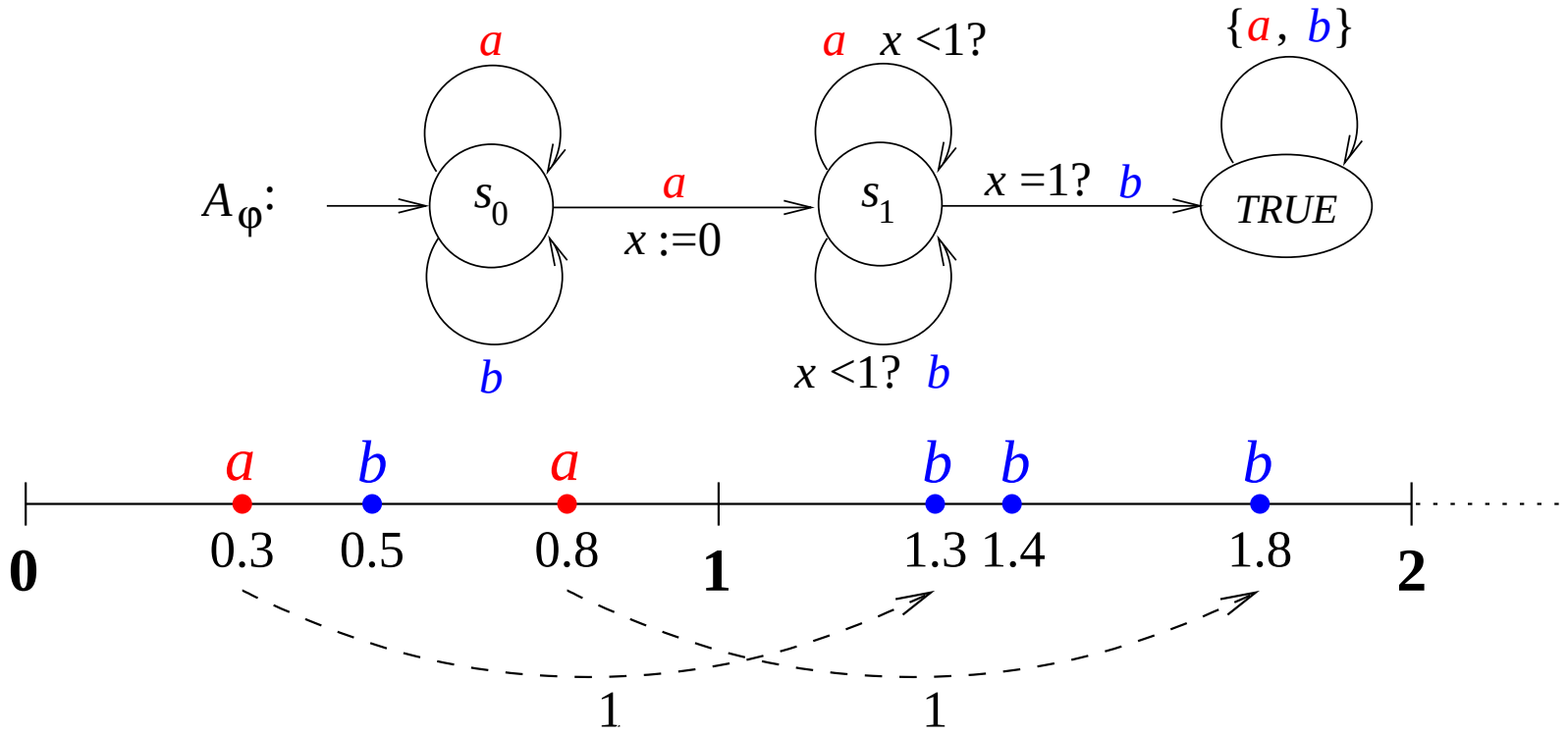
φ: $\Box(a \longrightarrow \Diamond_{=1} b)$



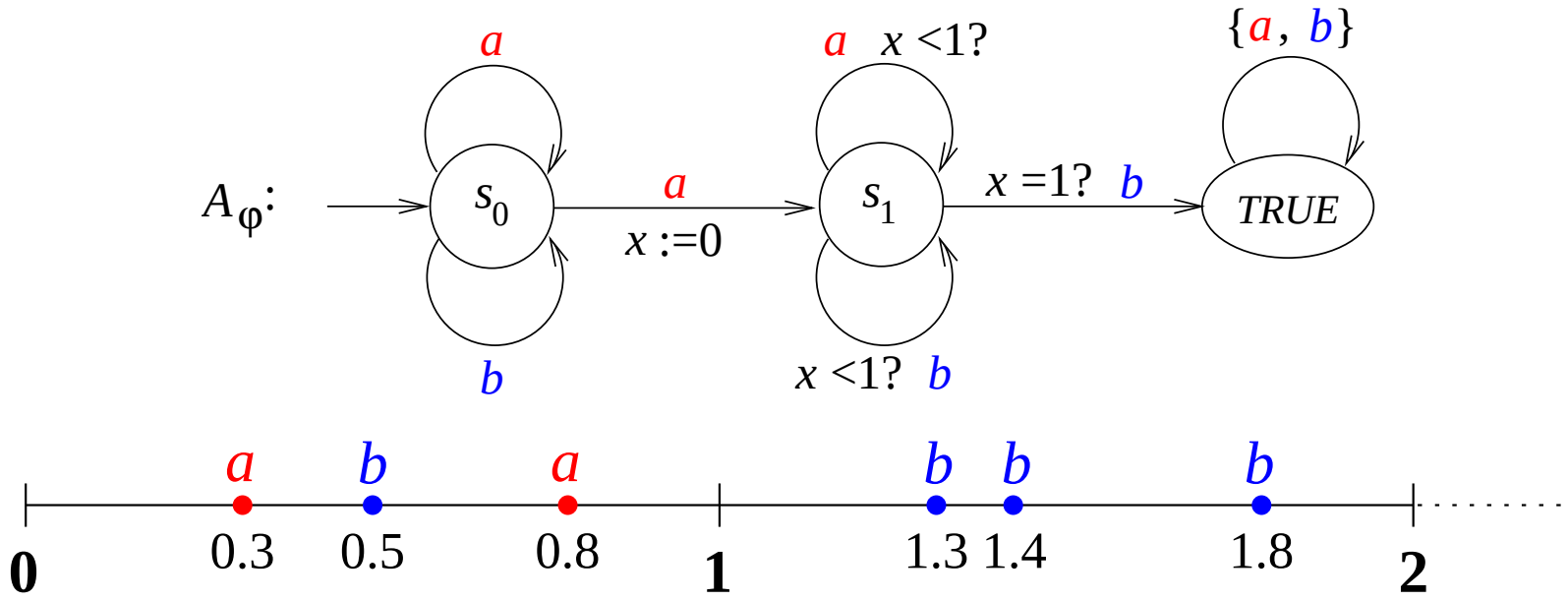
φ: $\Box(a \longrightarrow \Diamond_{=1} b)$

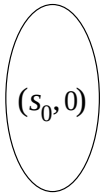


φ: $\Box(a \longrightarrow \Diamond_{=1} b)$

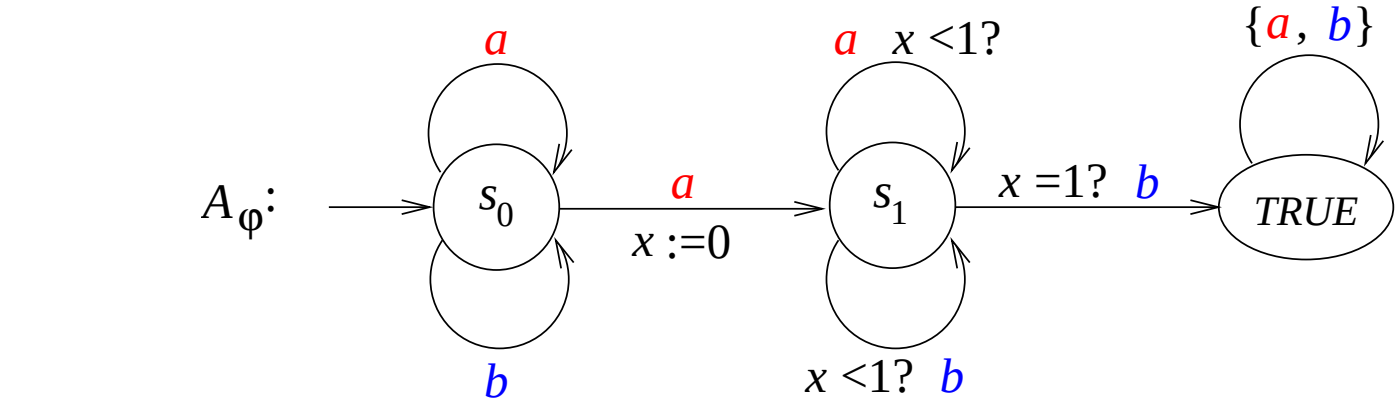


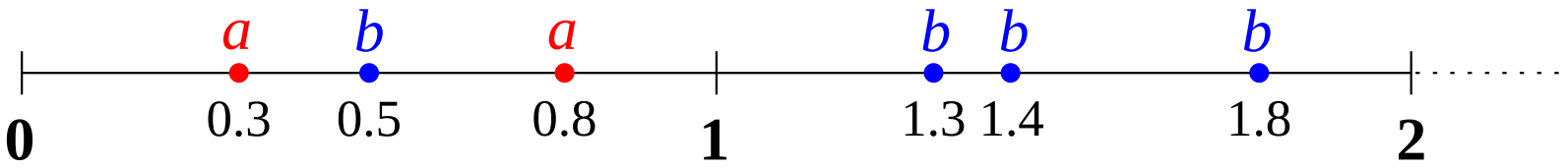
φ: $\Box(a \longrightarrow \Diamond_{=1} b)$

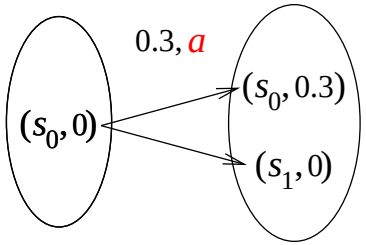




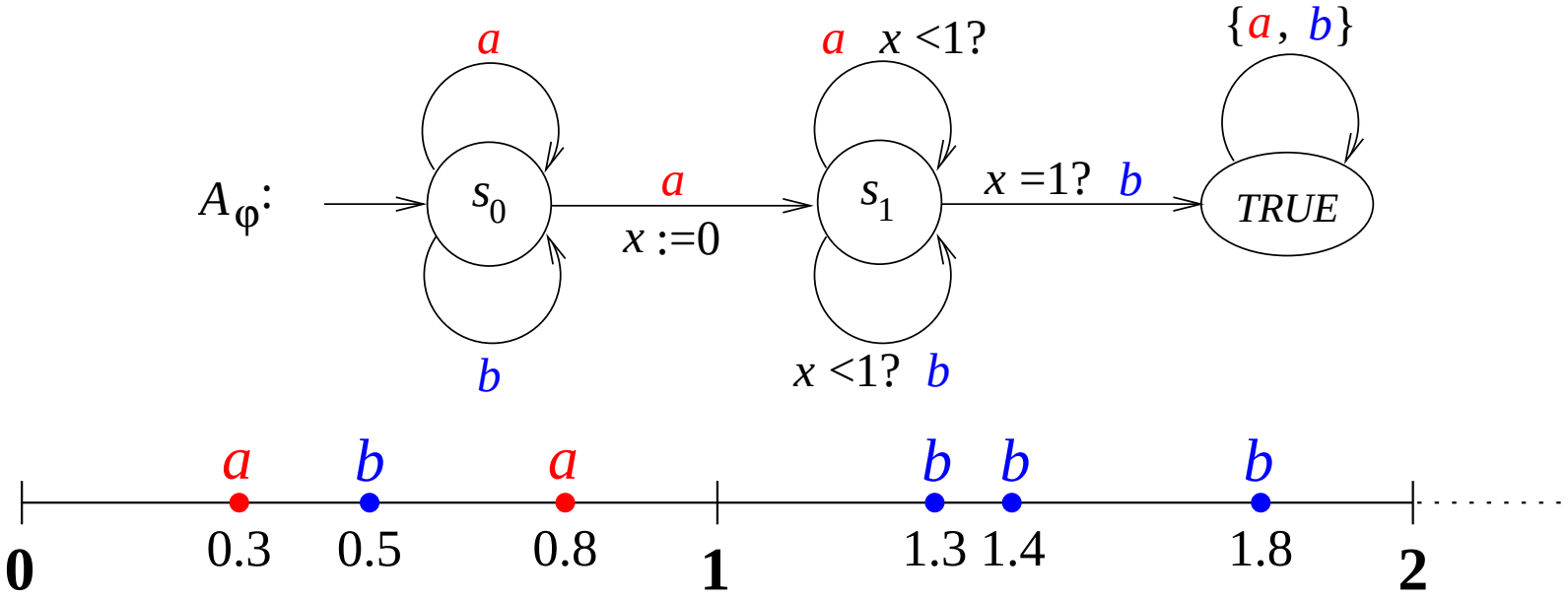
φ: $\Box(a \longrightarrow \Diamond_{=1} b)$

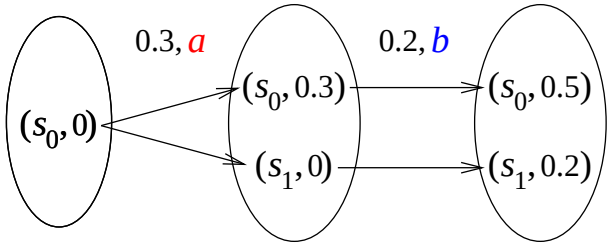




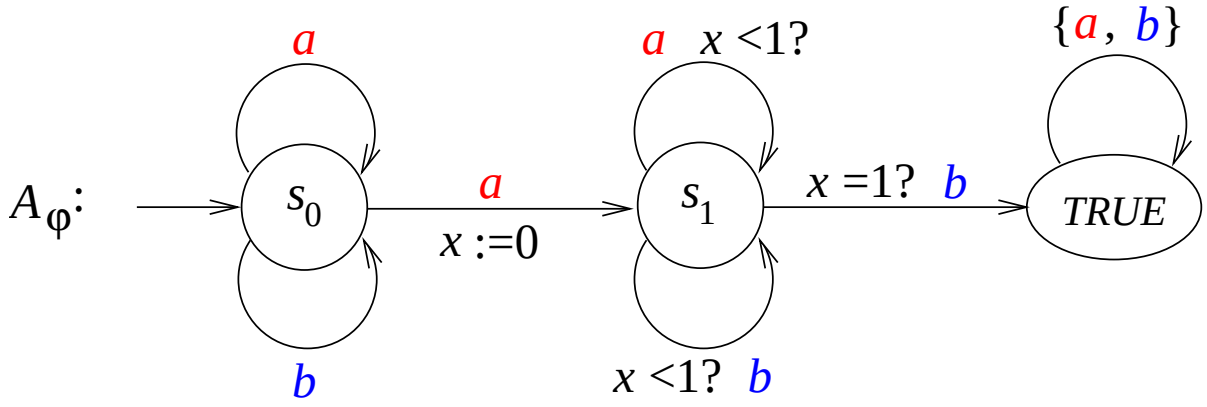


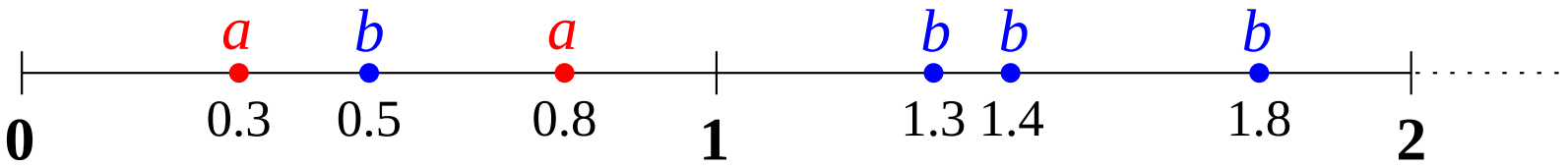
φ: $\Box(a \longrightarrow \Diamond_{=1} b)$

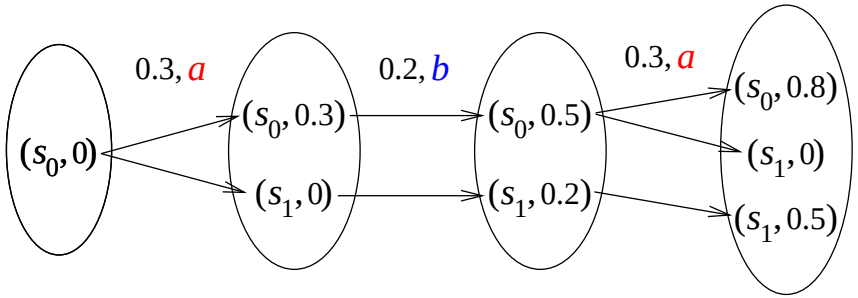




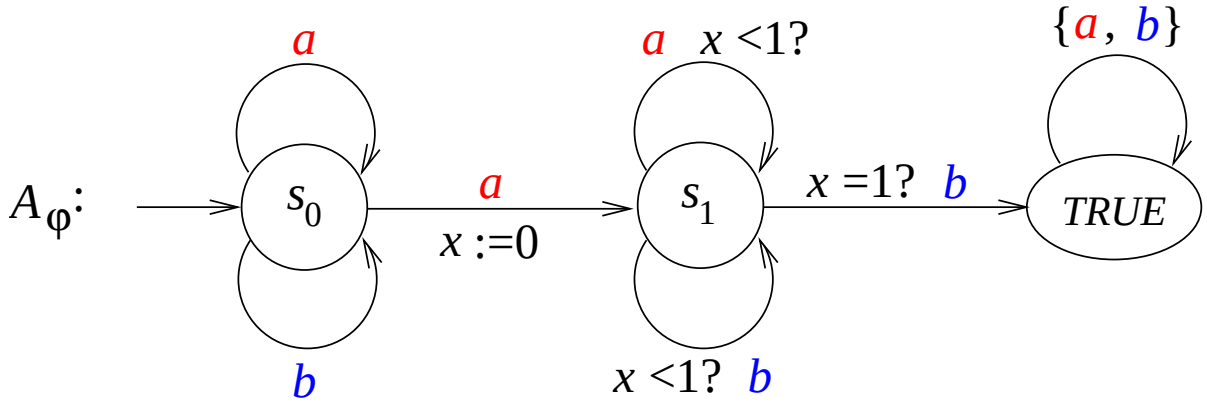
φ: $\Box(a \longrightarrow \Diamond_{=1} b)$

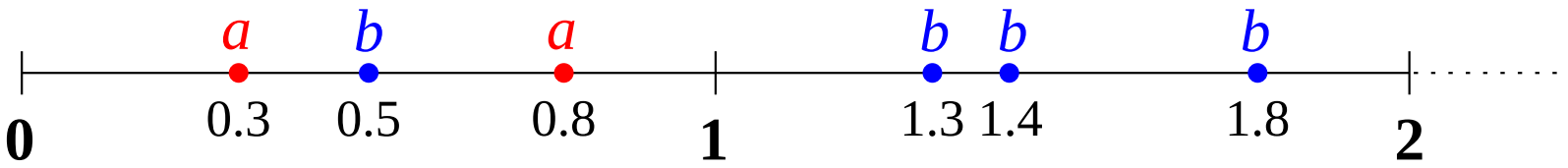


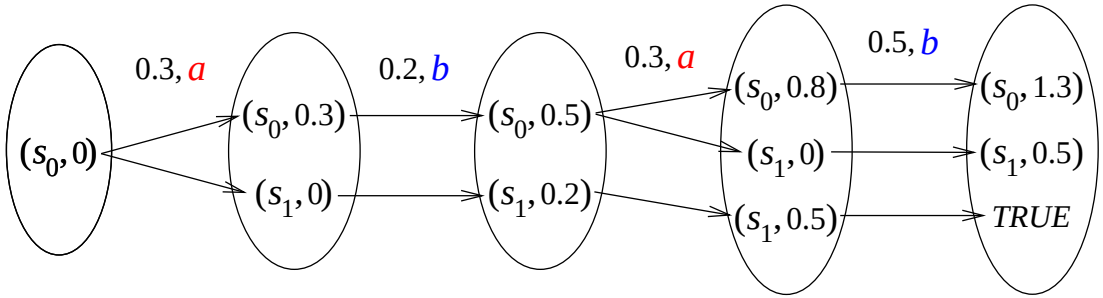




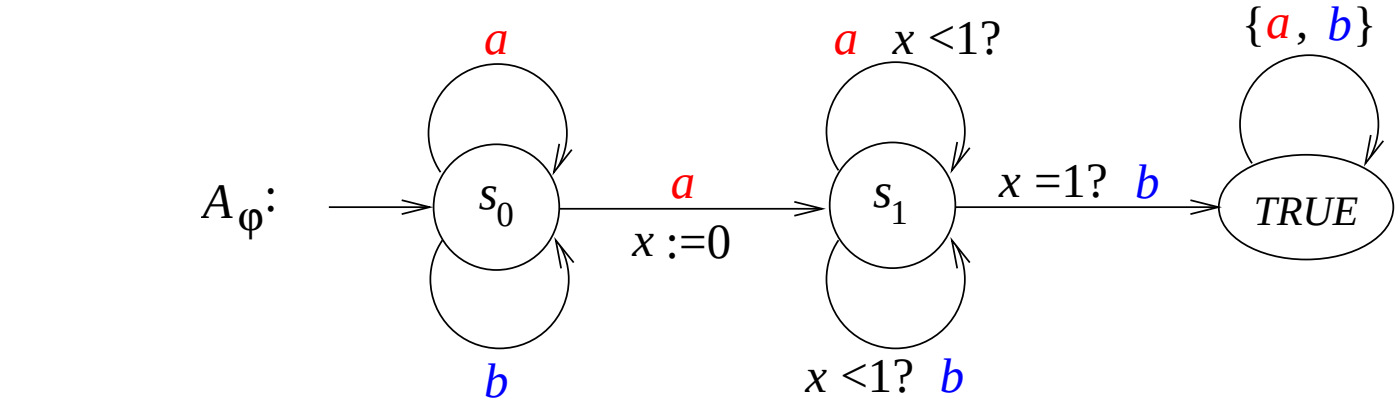
φ: $\Box(a \longrightarrow \Diamond_{=1} b)$

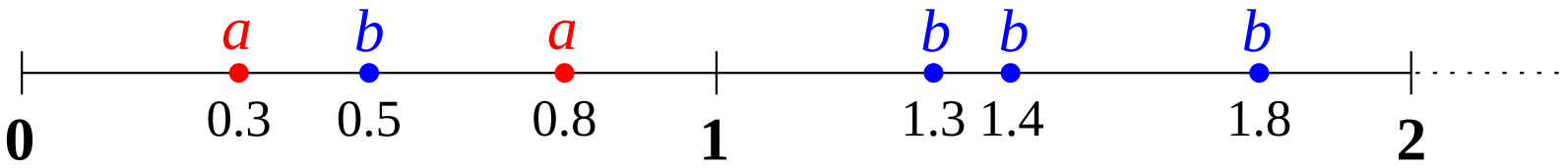


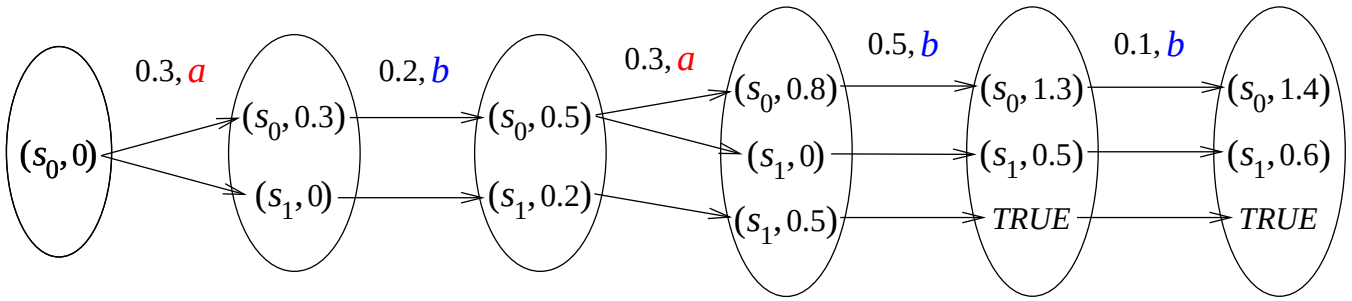




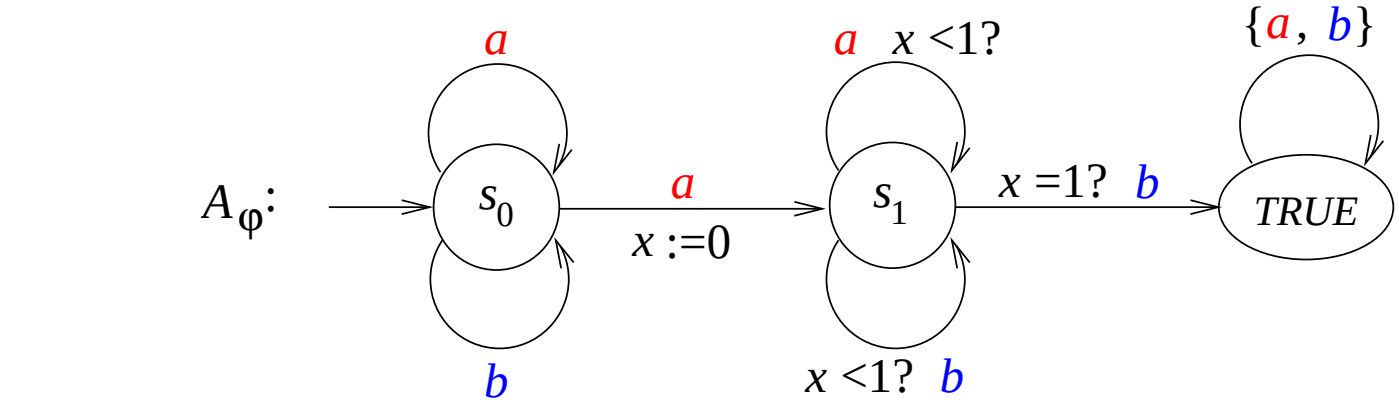
φ: $\Box(a \longrightarrow \Diamond_{=1} b)$

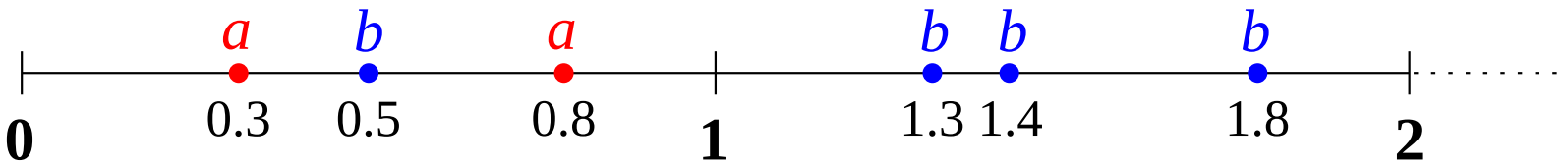


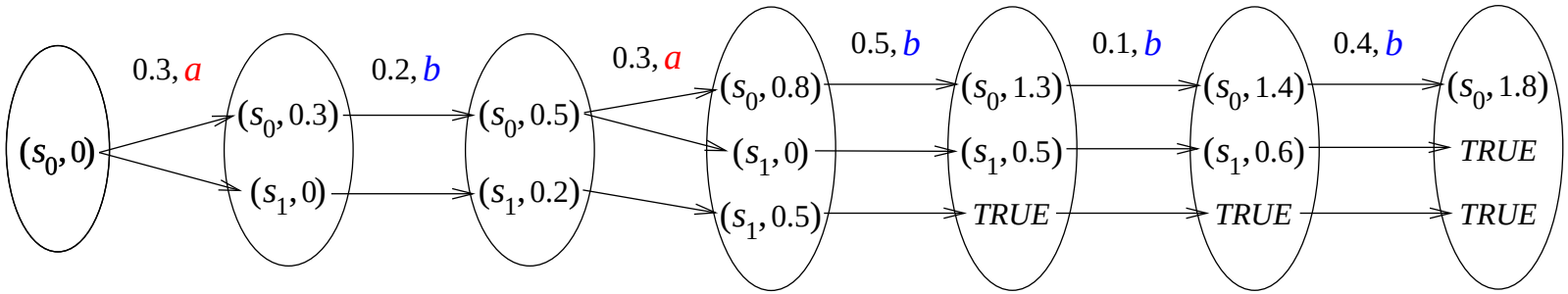




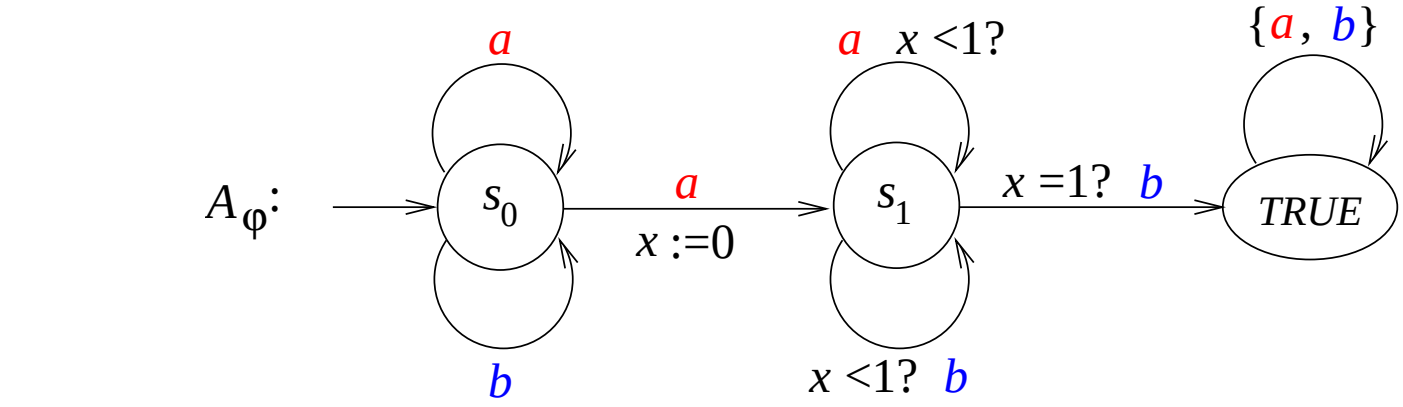
φ: $\Box(a \longrightarrow \Diamond_{=1} b)$

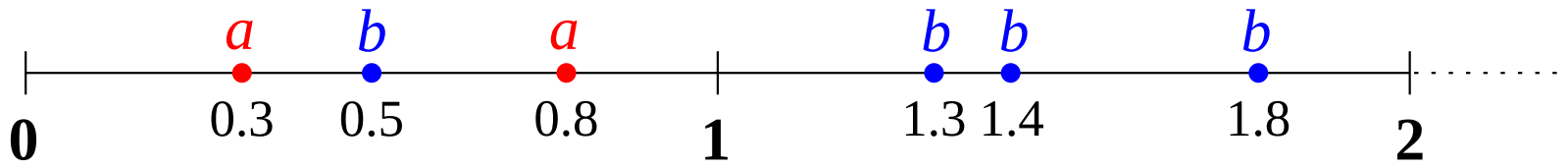


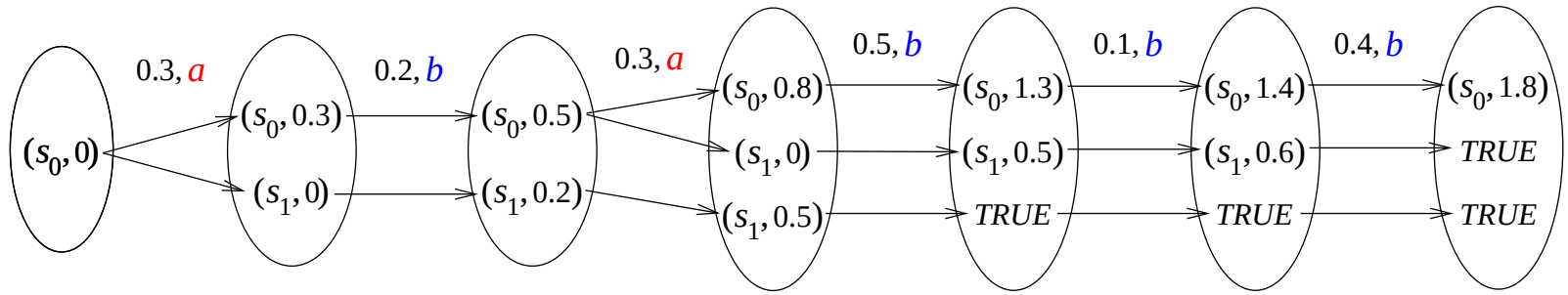




φ: $\Box(a \longrightarrow \Diamond_{=1} b)$







- Construct an infinite non-Zeno run to verify that $w \models \varphi$.

Well-quasi-orders

- Exploring the state space of $\mathcal{A}_\varphi$ is non-trivial since it has infinitely many configurations and does not admit a finite quotient.

Well-quasi-orders

- Exploring the state space of $\mathcal{A}_\varphi$ is non-trivial since it has infinitely many configurations and does not admit a finite quotient.
- In lieu of finiteness we observe the existence of a **well-quasi-order (wqo)** on the configurations of $\mathcal{A}_\varphi$.

Well-quasi-orders

- Exploring the state space of $\mathcal{A}_\varphi$ is non-trivial since it has infinitely many configurations and does not admit a finite quotient.
- In lieu of finiteness we observe the existence of a **well-quasi-order (wqo)** on the configurations of $\mathcal{A}_\varphi$.
  - A quasi-order $(Q, \preccurlyeq)$ is a well-quasi-order if for every infinite sequence $q_0, q_1, q_2, \ldots$ in Q there exist indices $i < j$ such that $q_i \preccurlyeq q_j$.

Well-quasi-orders

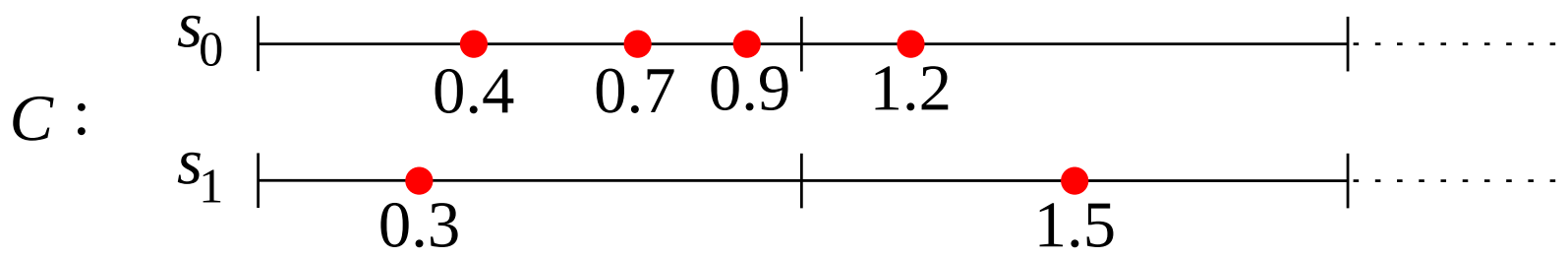
- Exploring the state space of $\mathcal{A}_\varphi$ is non-trivial since it has infinitely many configurations and does not admit a finite quotient.
- In lieu of finiteness we observe the existence of a **well-quasi-order (wqo)** on the configurations of $\mathcal{A}_\varphi$.
  - A quasi-order $(Q, \preccurlyeq)$ is a well-quasi-order if for every infinite sequence $q_0, q_1, q_2, \ldots$ in Q there exist indices $i < j$ such that $q_i \preccurlyeq q_j$.
  - The subword order (e.g., $HIGMAN \preccurlyeq HIGHMOUNTAIN$) is a wqo.

Well-quasi-orders

- Exploring the state space of $\mathcal{A}_\varphi$ is non-trivial since it has infinitely many configurations and does not admit a finite quotient.
- In lieu of finiteness we observe the existence of a **well-quasi-order (wqo)** on the configurations of $\mathcal{A}_\varphi$.
  - A quasi-order $(Q, \preccurlyeq)$ is a well-quasi-order if for every infinite sequence $q_0, q_1, q_2, \ldots$ in Q there exist indices $i < j$ such that $q_i \preccurlyeq q_j$.
  - The subword order (e.g., $HIGMAN \preccurlyeq HIGHMOUNTAIN$) is a wqo.
- The wqo is used to prove termination of model checking and satisfiability algorithms.

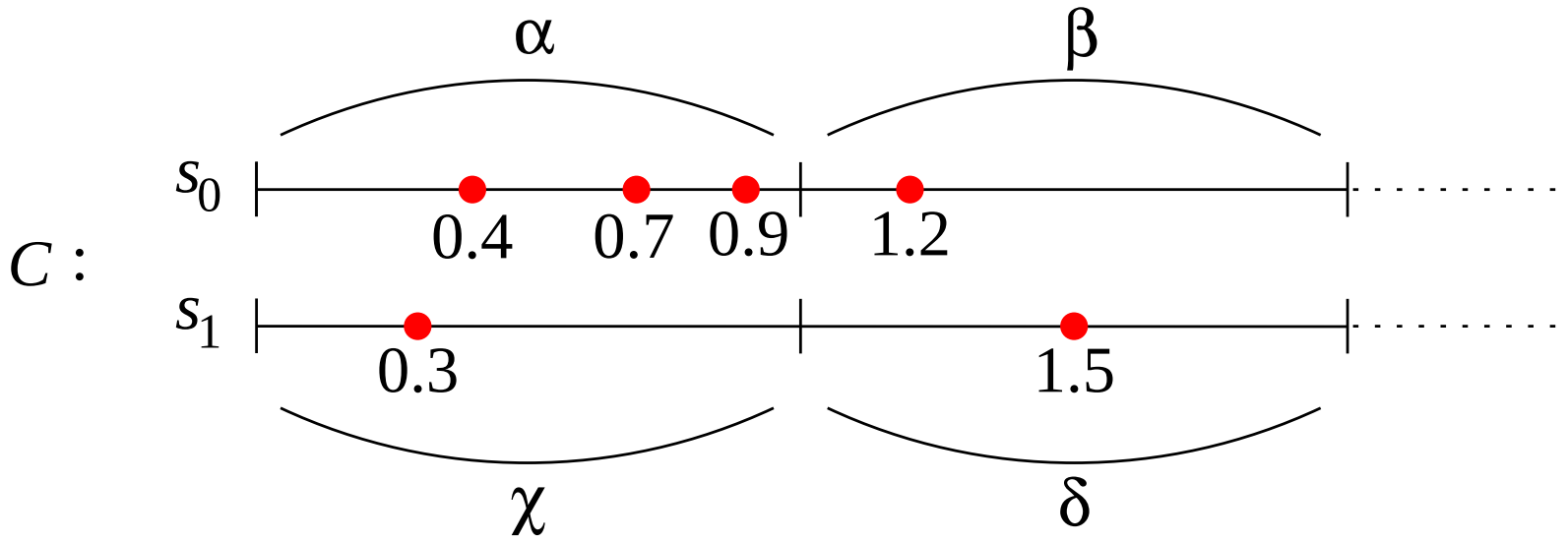
Encoding Configurations as Words: Example

Consider the configuration C:



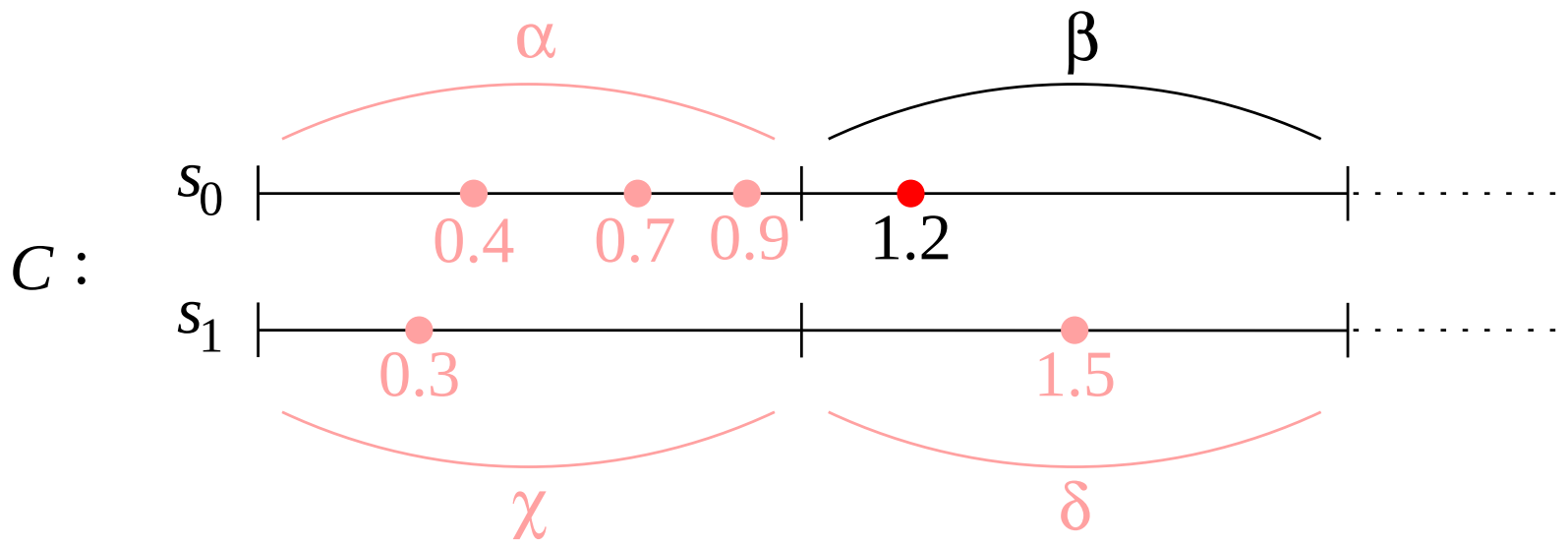
Encoding Configurations as Words: Example

Consider the configuration C:



Encoding Configurations as Words: Example

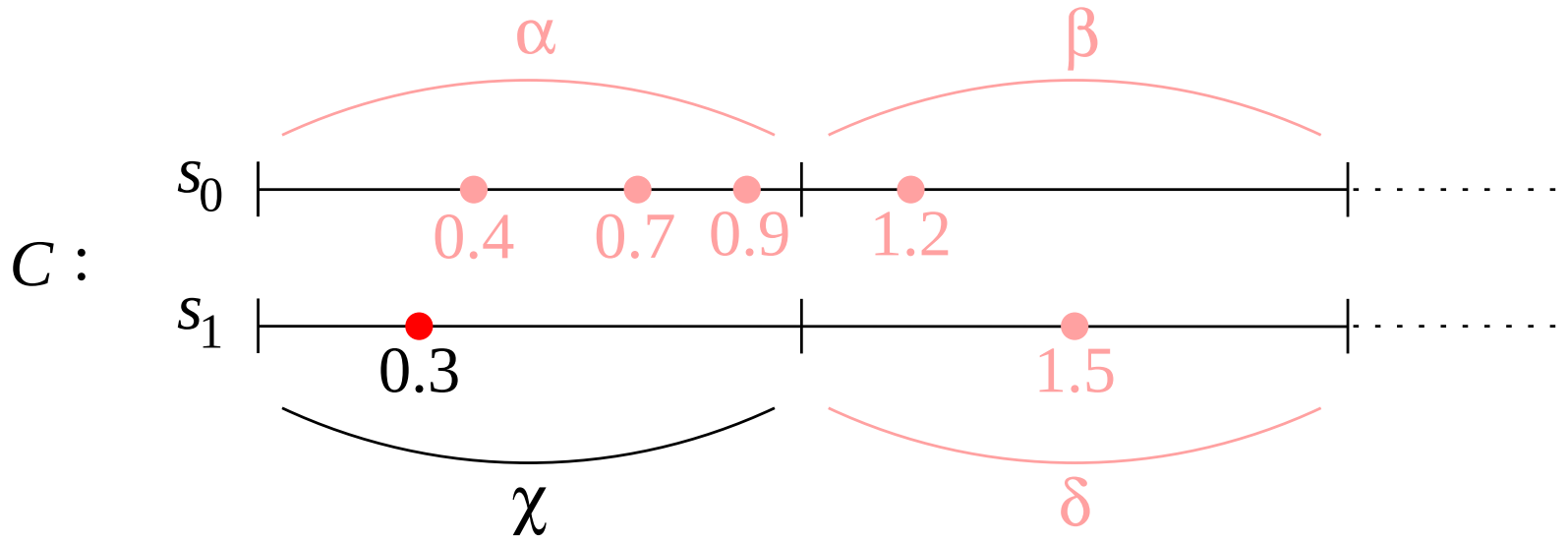
Consider the configuration C:



We encode C as $H(C) = \beta \ldots$

Encoding Configurations as Words: Example

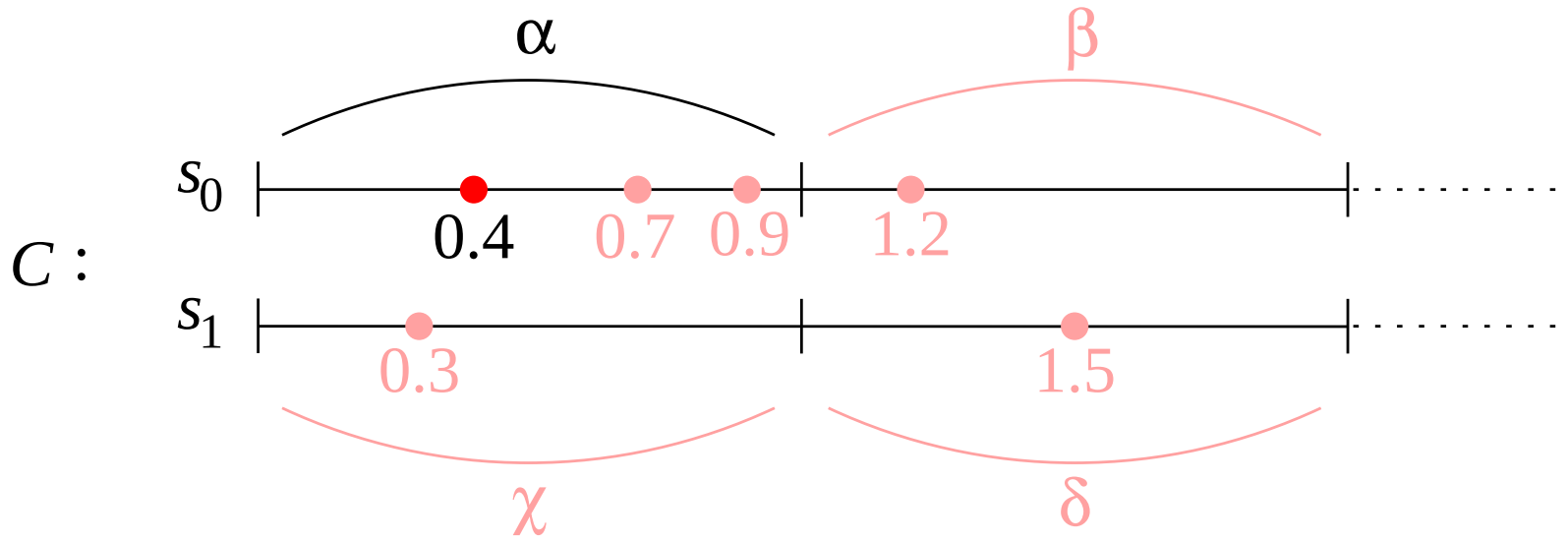
Consider the configuration C:



We encode C as $H(C) = \beta\chi\ldots$

Encoding Configurations as Words: Example

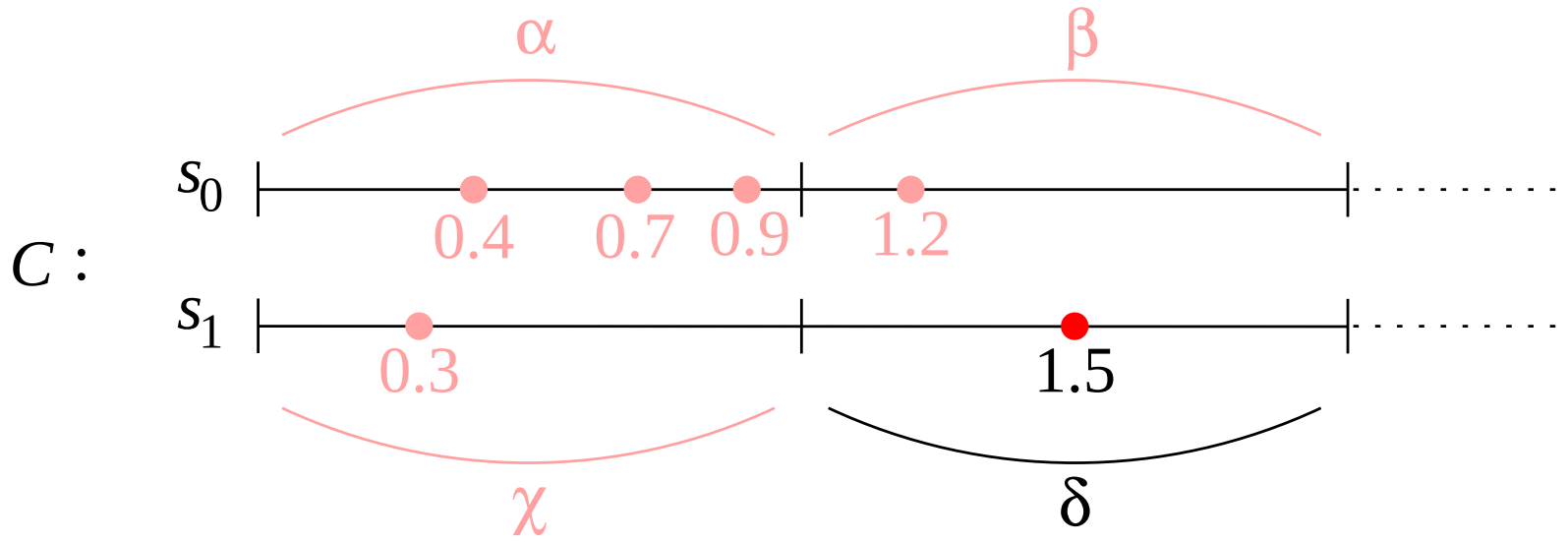
Consider the configuration C:



We encode C as $H(C) = \beta\chi\alpha\ldots$

Encoding Configurations as Words: Example

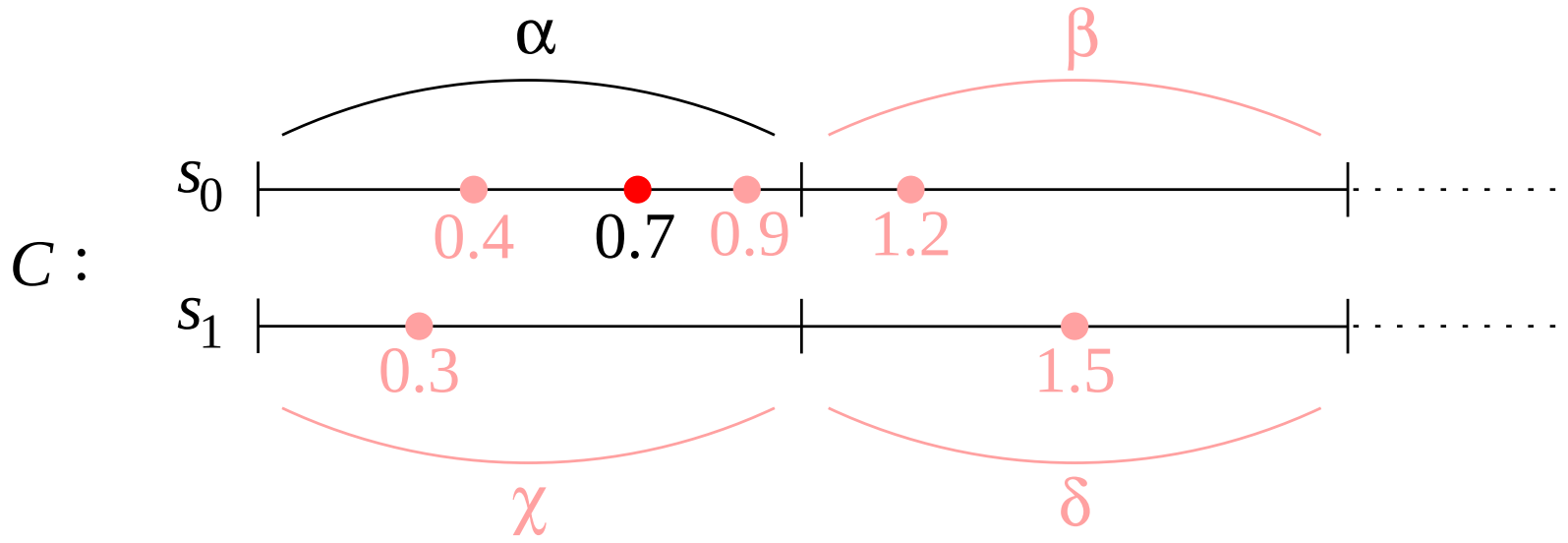
Consider the configuration C:



We encode C as $H(C) = \beta\chi\alpha\delta\ldots$

Encoding Configurations as Words: Example

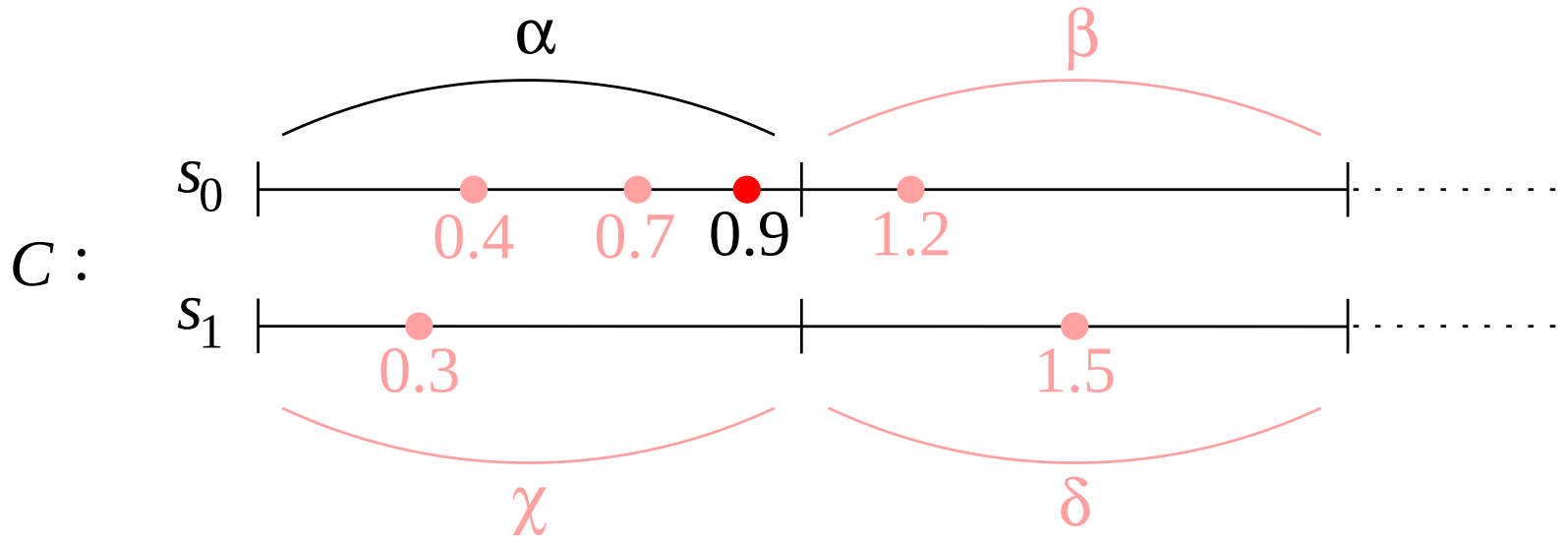
Consider the configuration C:



We encode C as $H(C) = \beta\chi\alpha\delta\alpha\ldots$

Encoding Configurations as Words: Example

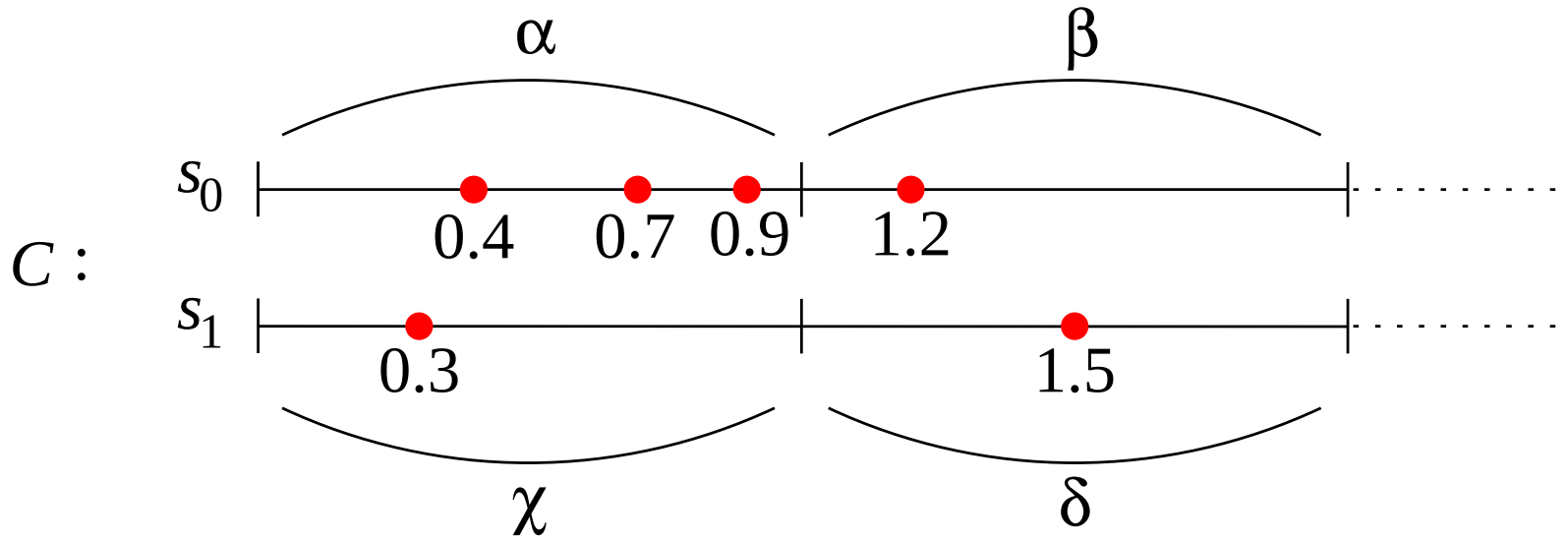
Consider the configuration C:



We encode C as $H(C) = \beta\chi\alpha\delta\alpha\alpha$

Encoding Configurations as Words: Example

Consider the configuration C:



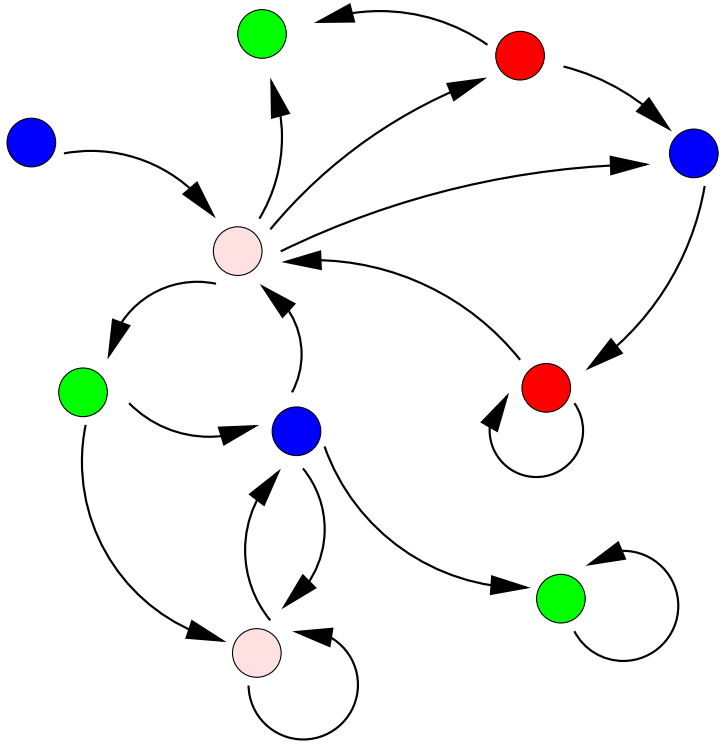
We encode C as $H(C) = \beta\chi\alpha\delta\alpha\alpha$.

The Simulation Lemma

Lemma. Suppose C and D are configurations of $\mathcal{A}_\varphi$ such that $H(C) \preccurlyeq H(D)$. If $D \longrightarrow D'$ then there exists C' such that $C \longrightarrow C'$ and $H(C') \preccurlyeq H(D')$.

Using Well-quasi-orders

MTL formula $\varphi \Longrightarrow$ Automaton $\mathcal{A}_\varphi \Longrightarrow$ Configuration graph $\mathcal{G}_\varphi$.



Higman's Lemma yields a **compatible** well-quasi-order on $\mathcal{G}_\varphi$.

Safety MTL is Fully Decidable

- The **model-checking problem** for Safety MTL is decidable.
  - The decision procedure uses a forward-reachability algorithm to search for a finite counterexample to '$SYSTEM \models \varphi$?'.

Safety MTL is Fully Decidable

- The **model-checking problem** for Safety MTL is decidable.
  - The decision procedure uses a forward-reachability algorithm to search for a finite counterexample to '$SYSTEM \models \varphi$?'.
- The **satisfiability problem** for Safety MTL is decidable.
  - The decision procedure uses a backward-reachability algorithm to search for an infinite timed word satisfying φ.

Safety MTL is Fully Decidable

- The **model-checking problem** for Safety MTL is decidable.
  - The decision procedure uses a forward-reachability algorithm to search for a finite counterexample to '$SYSTEM \models \varphi$?'.
- The **satisfiability problem** for Safety MTL is decidable.
  - The decision procedure uses a backward-reachability algorithm to search for an infinite timed word satisfying φ.
- The **refinement problem** for Safety MTL is decidable.
  - The decision procedure combines forward reachability and backward reachability.

Using Well-quasi-orders

Let $(Q, \preccurlyeq)$ be a wqo.

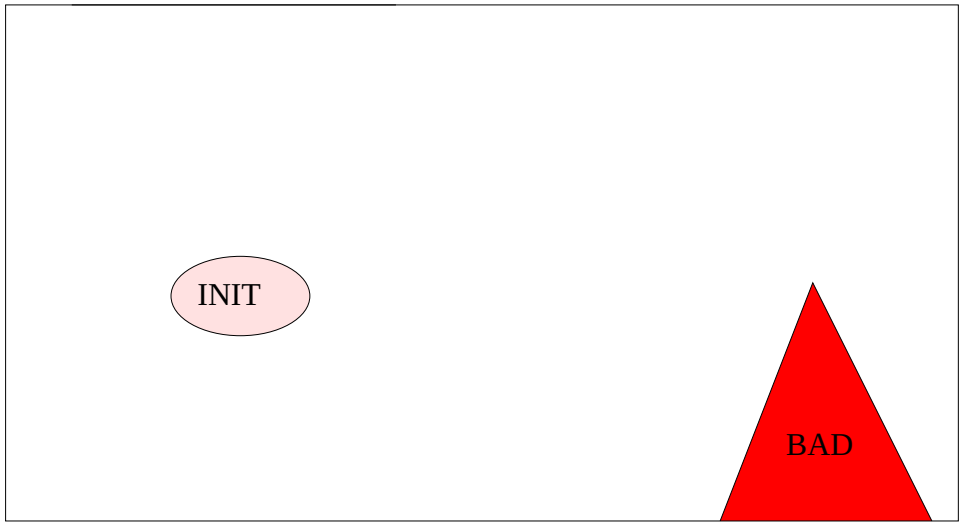
- **Defn.** A **basis** of an upper set $U \subseteq Q$ is a subset $U_b \subseteq U$ such that $U = {\uparrow} U_b$.
- **Defn.** A **cobasis** of a lower set $L \subseteq Q$ is a basis of $Q \setminus L$.
- **Prop.** Every upper set has a finite basis and every lower set has a finite cobasis.
- **Prop.** Every increasing sequence of upper sets $U_0 \subseteq U_1 \subseteq U_2 \subseteq \cdots$ eventually converges.
- **Prop.** Every decreasing sequence of lower sets $L_0 \supseteq L_1 \supseteq L_2 \supseteq \cdots$ eventually converges.

Model Checking

Is a bad configuration reachable in $SYSTEM \,||\, \mathcal{A}_{\neg\varphi}$?

Safe overestimation of reachable configurations:

$$U_0 = \uparrow \{init\}, \qquad U_{n+1} = U_n \cup \uparrow Succ(U_n).$$



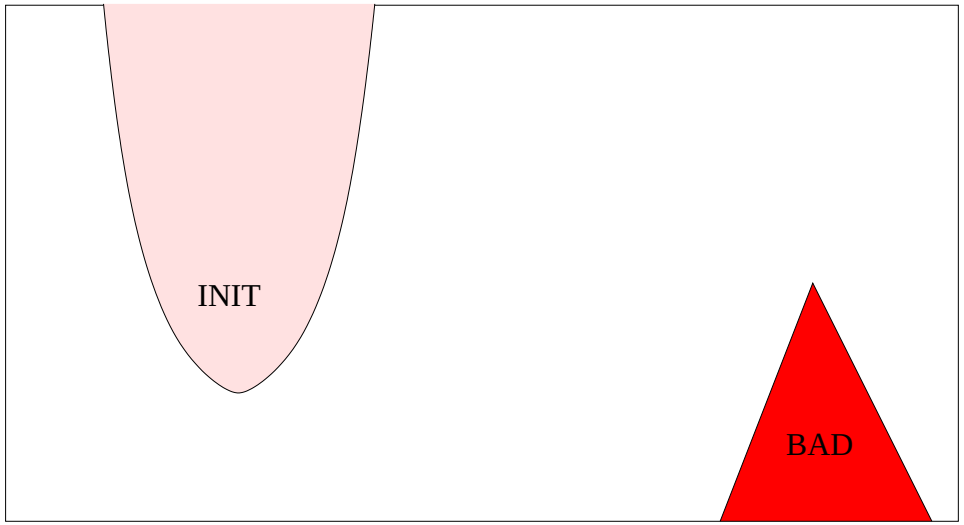
$$U_0 \subseteq U_1 \subseteq U_2 \subseteq \cdots \subseteq U_n = U_{n+1} \qquad BAD \cap U_n \neq \emptyset \text{ ?}$$

Model Checking

Is a bad configuration reachable in $SYSTEM \,||\, \mathcal{A}_{\neg\varphi}$?

Safe overestimation of reachable configurations:

$$U_0 = \uparrow \{init\}, \qquad U_{n+1} = U_n \cup \uparrow Succ(U_n).$$



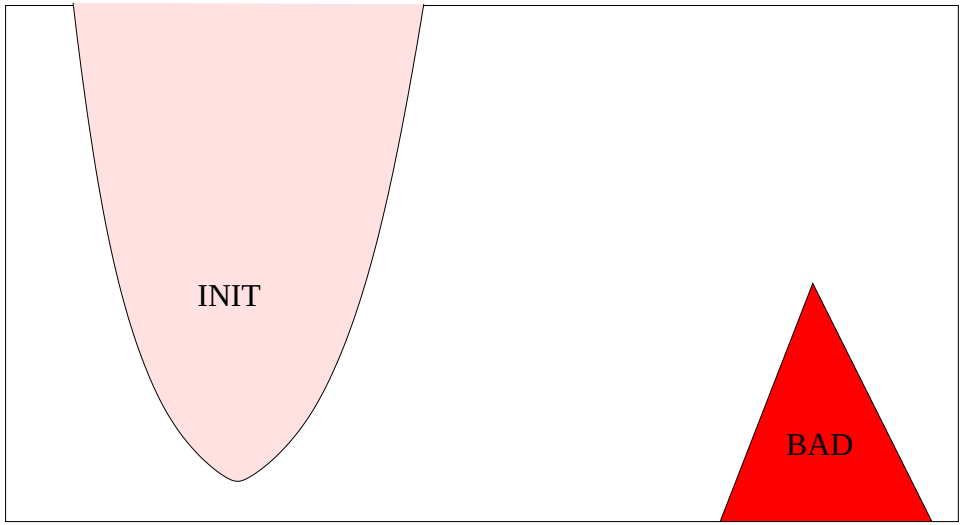
$U_0 \subseteq U_1 \subseteq U_2 \subseteq \cdots \subseteq U_n = U_{n+1}$ $\qquad$ $BAD \cap U_n \neq \emptyset$?

Model Checking

Is a bad configuration reachable in $SYSTEM \,||\, \mathcal{A}_{\neg\varphi}$?

Safe overestimation of reachable configurations:

$$U_0 = \uparrow \{init\}, \qquad U_{n+1} = U_n \cup \uparrow Succ(U_n).$$



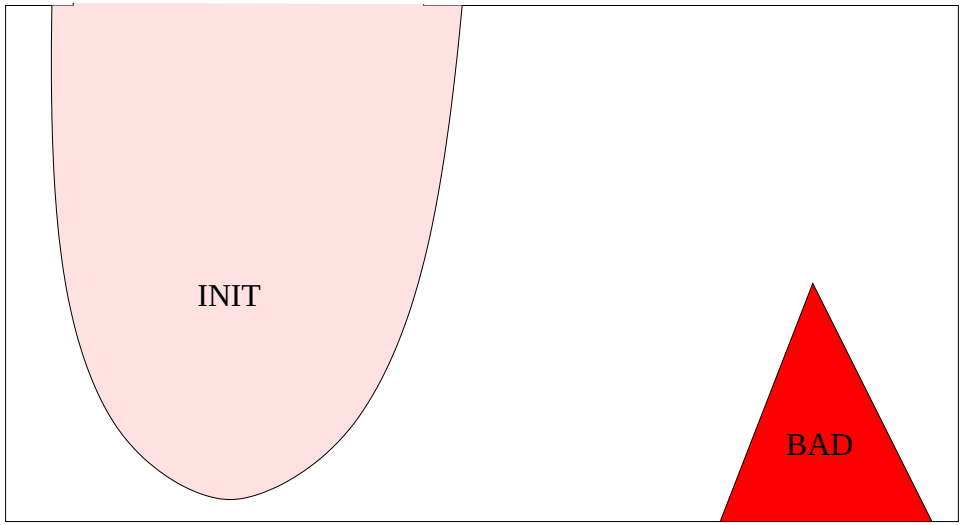
$$U_0 \subseteq U_1 \subseteq U_2 \subseteq \cdots \subseteq U_n = U_{n+1} \qquad BAD \cap U_n \neq \emptyset \,?$$

Model Checking

Is a bad configuration reachable in $SYSTEM \,||\, \mathcal{A}_{\neg\varphi}$?

Safe overestimation of reachable configurations:

$$U_0 = \uparrow \{init\}, \qquad U_{n+1} = U_n \cup \uparrow Succ(U_n).$$



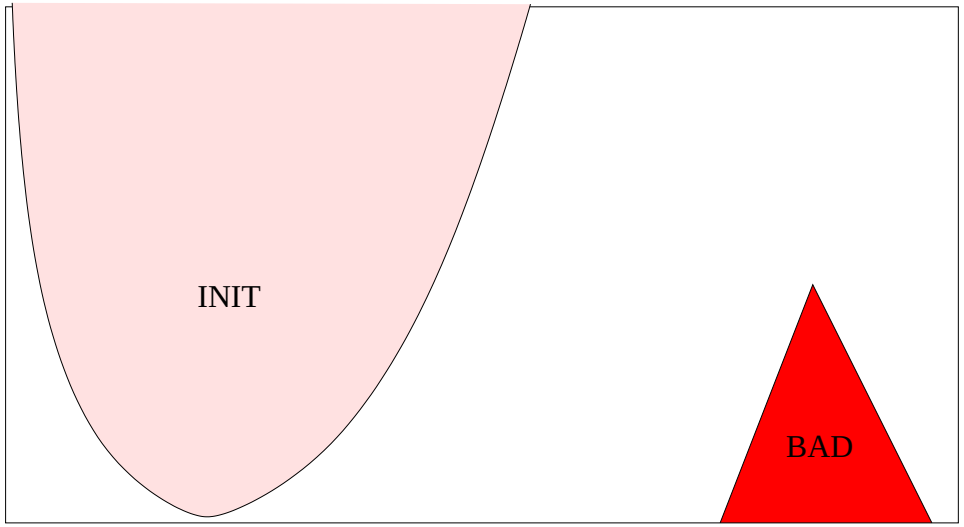
$$U_0 \subseteq U_1 \subseteq U_2 \subseteq \cdots \subseteq U_n = U_{n+1} \qquad BAD \cap U_n \neq \emptyset \,?$$

Model Checking

Is a bad configuration reachable in $SYSTEM \,||\, \mathcal{A}_{\neg\varphi}$?

Safe overestimation of reachable configurations:

$$U_0 = \uparrow \{init\}, \qquad U_{n+1} = U_n \cup \uparrow Succ(U_n).$$



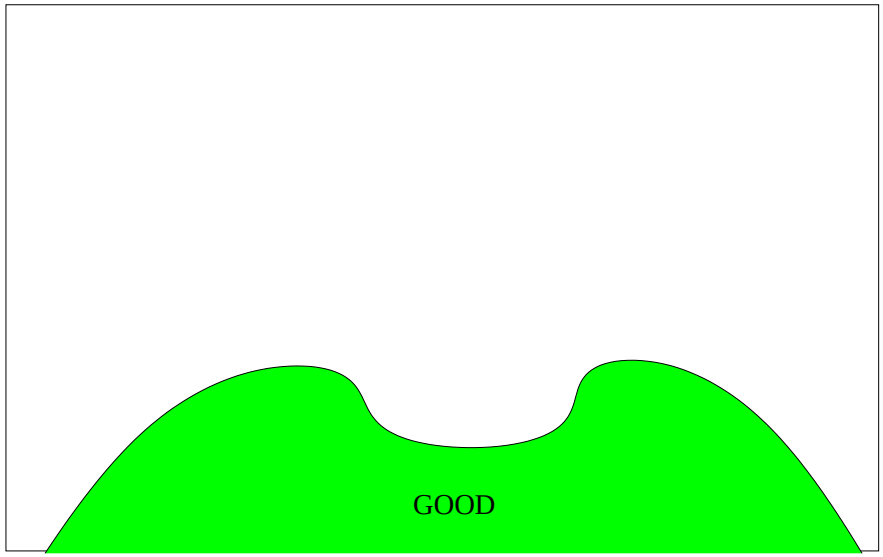
$$U_0 \subseteq U_1 \subseteq U_2 \subseteq \cdots \subseteq U_n = U_{n+1} \qquad BAD \cap U_n \neq \emptyset \text{ ?}$$

Satisfiability

Does $\mathcal{A}_\varphi$ have an infinite non-Zeno computation ?

The target set of 'good' configurations is a lower set.

$$L_0 = Q, \qquad L_{n+1} = Q \setminus \uparrow (Q \setminus Pred(L_n)).$$



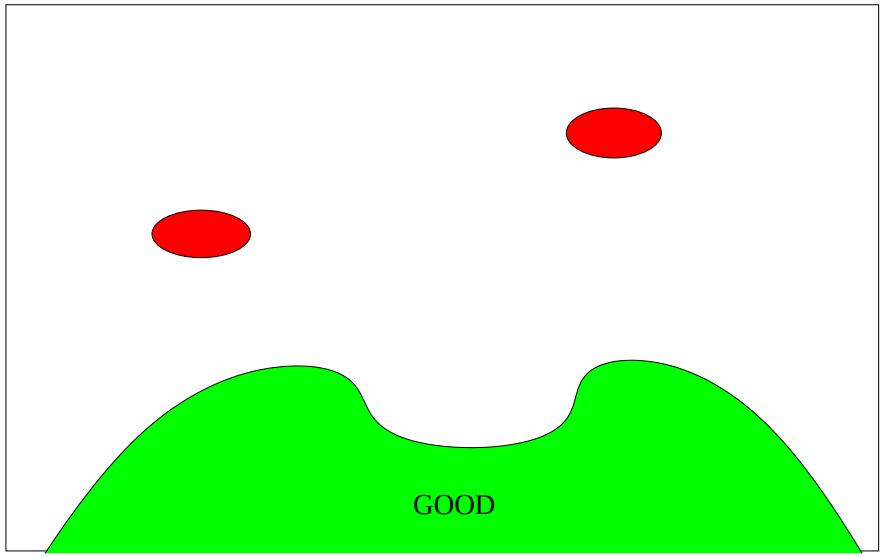
$$L_0 \supseteq L_1 \supseteq L_2 \supseteq L_3 \supseteq \cdots \supseteq L_n = L_{n+1} = GOOD.$$

Satisfiability

Does $\mathcal{A}_\varphi$ have an infinite non-Zeno computation ?

The target set of 'good' configurations is a lower set.

$$L_0 = Q, \qquad L_{n+1} = Q \setminus \uparrow (Q \setminus Pred(L_n)).$$



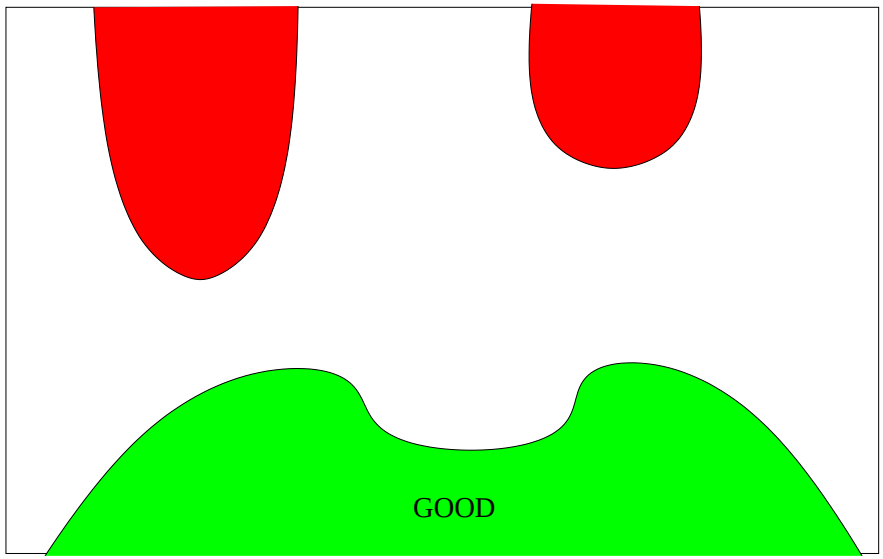
$$L_0 \supseteq L_1 \supseteq L_2 \supseteq L_3 \supseteq \cdots \supseteq L_n = L_{n+1} = GOOD.$$

Satisfiability

Does $\mathcal{A}_\varphi$ have an infinite non-Zeno computation ?

The target set of 'good' configurations is a lower set.

$$L_0 = Q, \qquad L_{n+1} = Q \setminus \uparrow (Q \setminus Pred(L_n)).$$



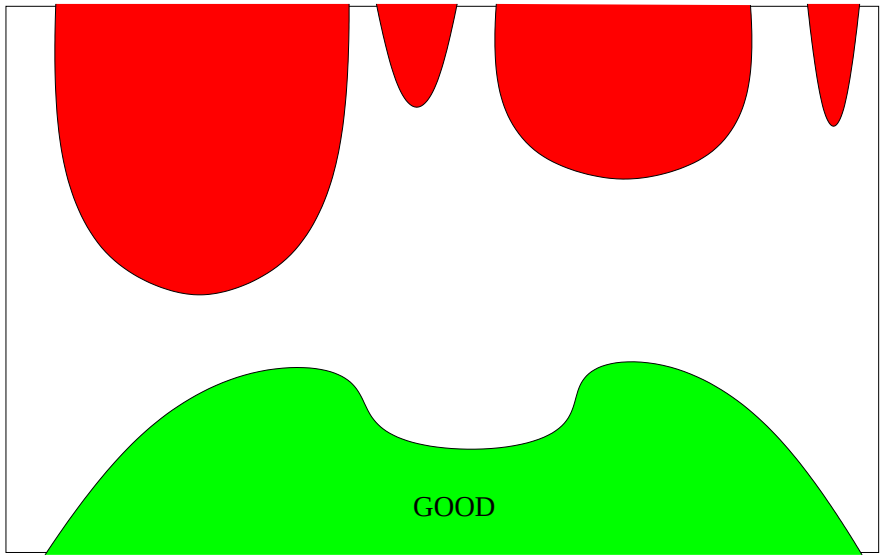
$$L_0 \supseteq L_1 \supseteq L_2 \supseteq L_3 \supseteq \cdots \supseteq L_n = L_{n+1} = GOOD.$$

Satisfiability

Does $\mathcal{A}_\varphi$ have an infinite non-Zeno computation ?

The target set of 'good' configurations is a lower set.

$$L_0 = Q, \qquad L_{n+1} = Q \setminus \uparrow (Q \setminus Pred(L_n)).$$



$$L_0 \supseteq L_1 \supseteq L_2 \supseteq L_3 \supseteq \cdots \supseteq L_n = L_{n+1} = GOOD.$$

Satisfiability

Does $\mathcal{A}_\varphi$ have an infinite non-Zeno computation ?

The target set of 'good' configurations is a lower set.

$$L_0 = Q, \qquad L_{n+1} = Q \setminus \uparrow (Q \setminus Pred(L_n)).$$

$$L_0 \supseteq L_1 \supseteq L_2 \supseteq L_3 \supseteq \cdots \supseteq L_n = L_{n+1} = GOOD.$$

Summary

- We considered specifying **metric** properties of real-time systems under a **non-Zeno** semantics.
- **Thm.** Metric Temporal Logic is undecidable.
- **Thm.** Safety Metric Temporal Logic is fully decidable.
  - Require that all eventualities be time bounded.
  - Liveness imposed by time-bounded eventualities is subsumed by the liveness of time.
  - Translate Safety MTL formulas into automata in which every location is accepting.
  - Check whether automata have non-Zeno runs.