SAT-based Abstraction Refinement for Real-time Systems

Stephanie Kemper[1] André Platzer[2,3]

[1]Centrum voor Wiskunde en Informatica, Software Engineering, Amsterdam, The Netherlands

[2]University of Oldenburg, Department of Computing Science, Germany

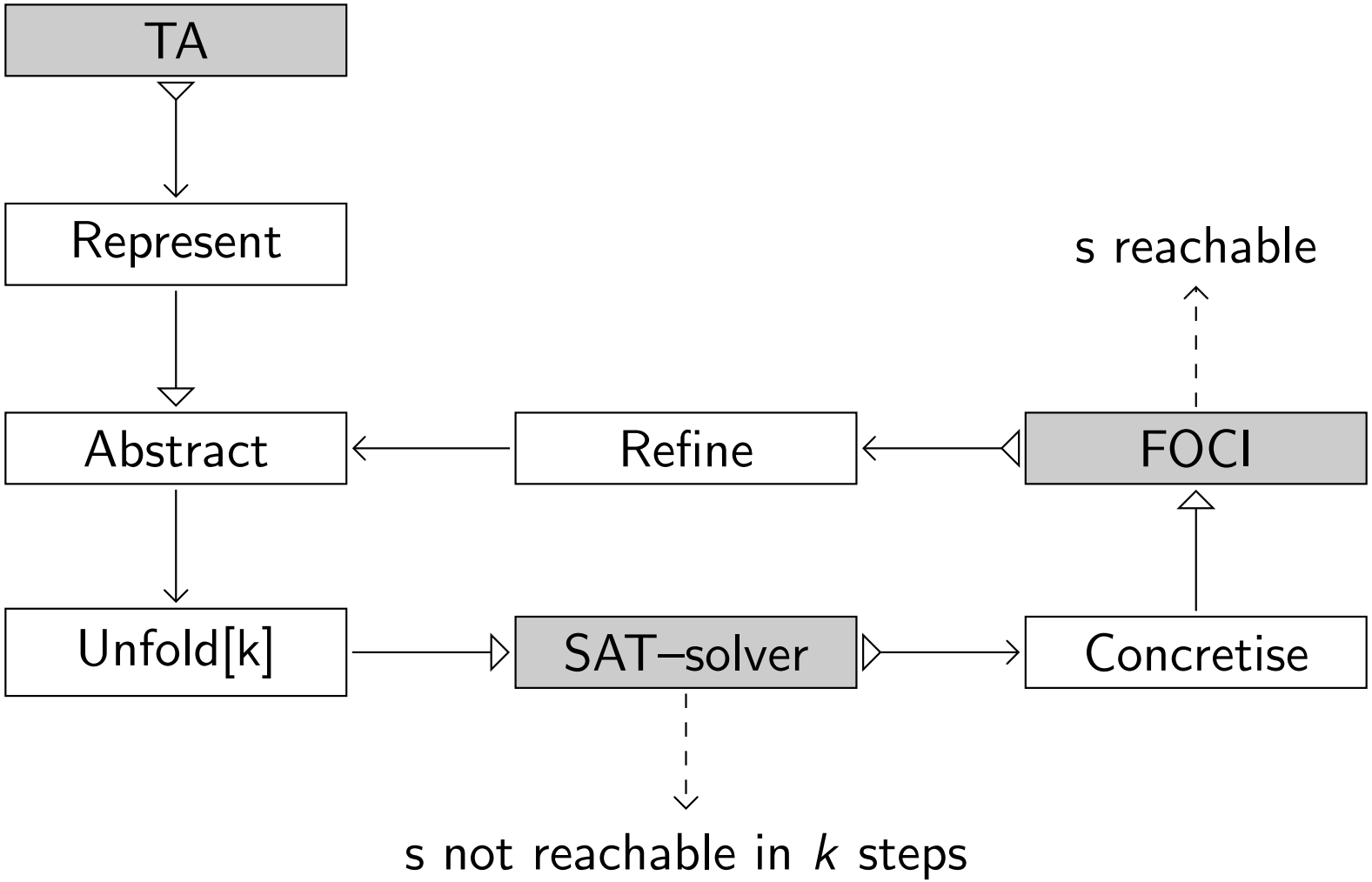
[3]Carnegie Mellon University, Pittsburgh, PA, USA

Third International Workshop on
Formal Aspects of Component Software (FACS'06)

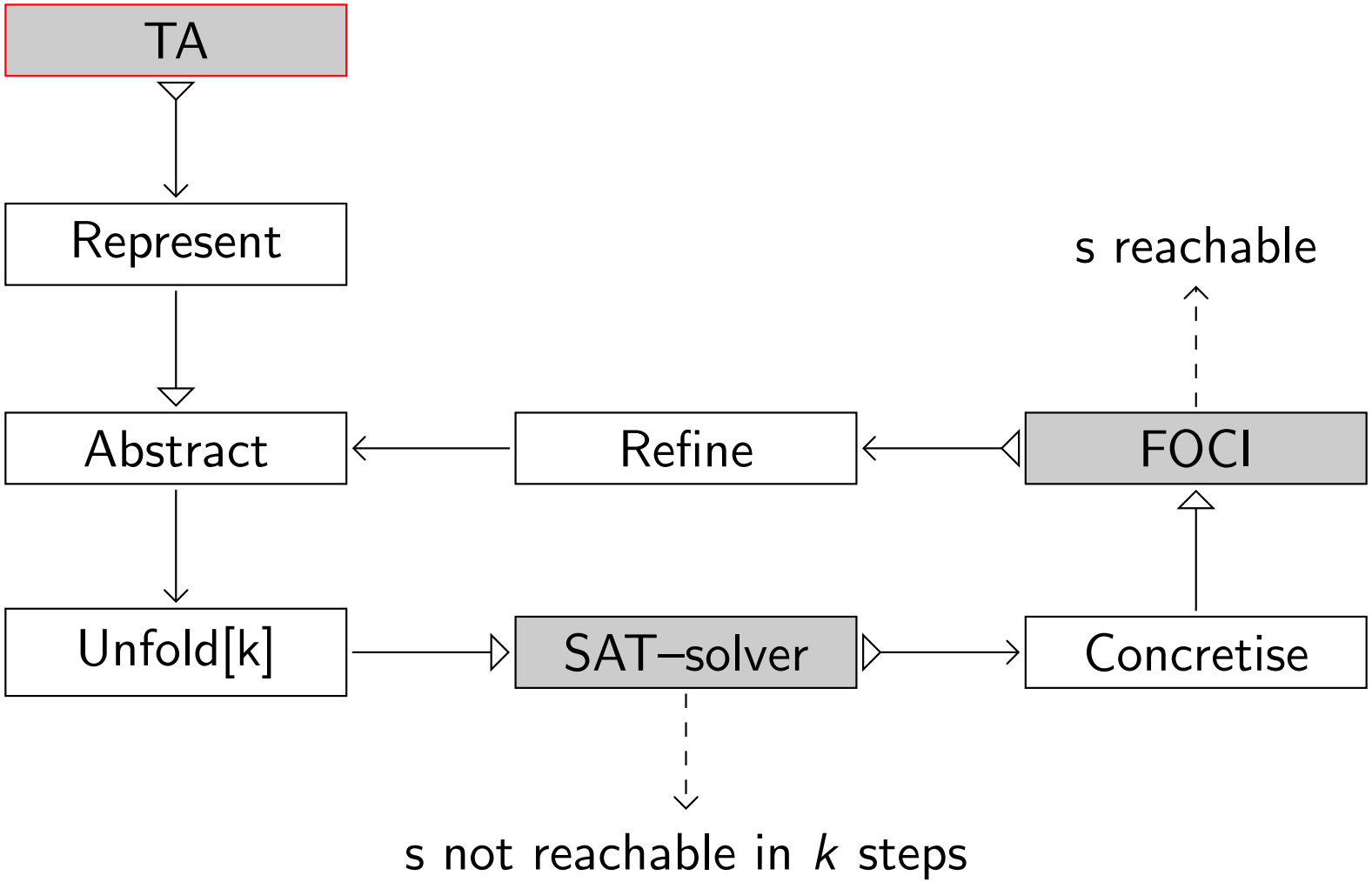
Motivation

- Failures in embedded systems: disastrous
- Safety critical systems must work correctly
- Single components, and their composition
- Responses in time
- Timed Systems: difficult to check (state explosion)
- Abstraction Refinement to cope with

Overview

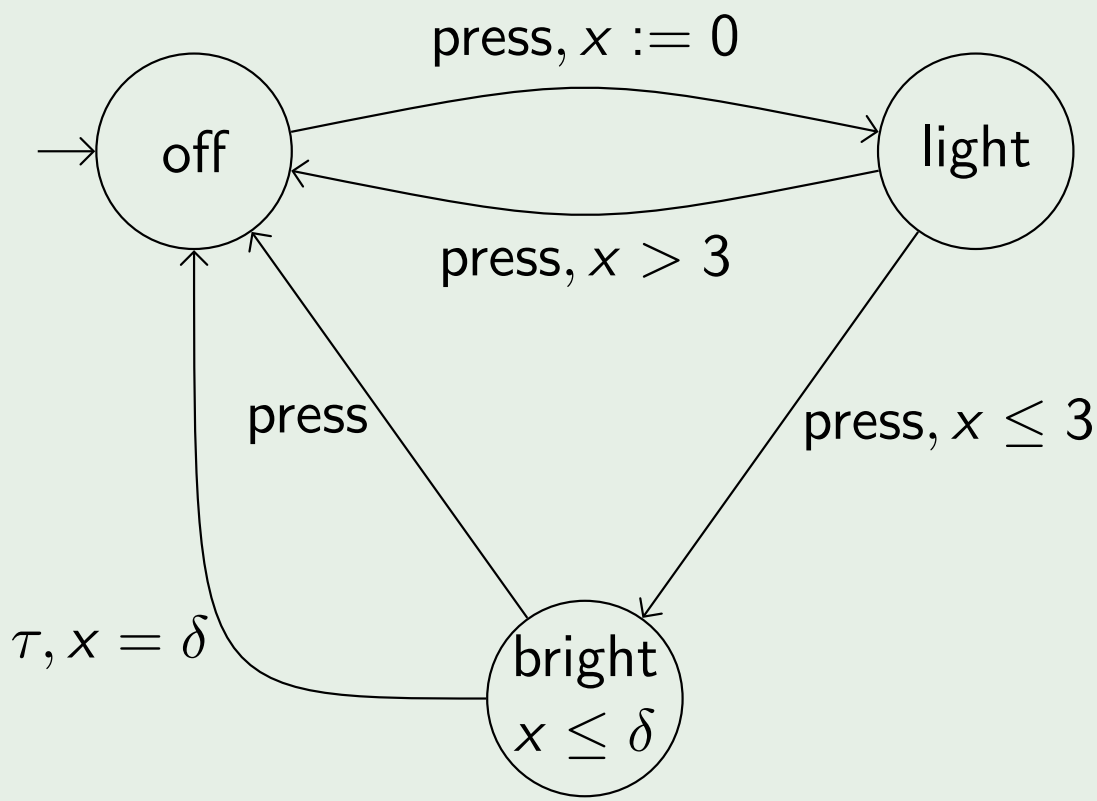

TA
Represent
Abstract
Unfold[k]
Refine
SAT–solver
FOCI
Concretise
s reachable
s not reachable in k steps

Timed Automata



Timed Automata

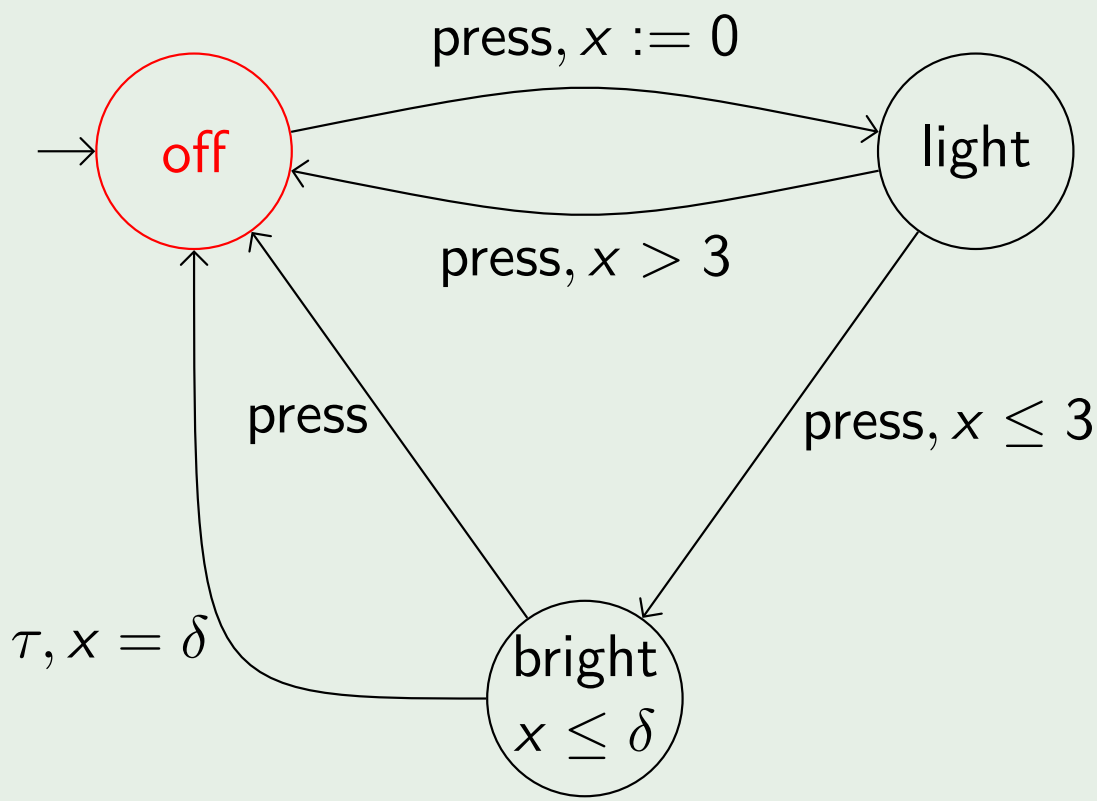
Example (Intelligent Light Controller)



Semantics = All possible traces

Timed Automata

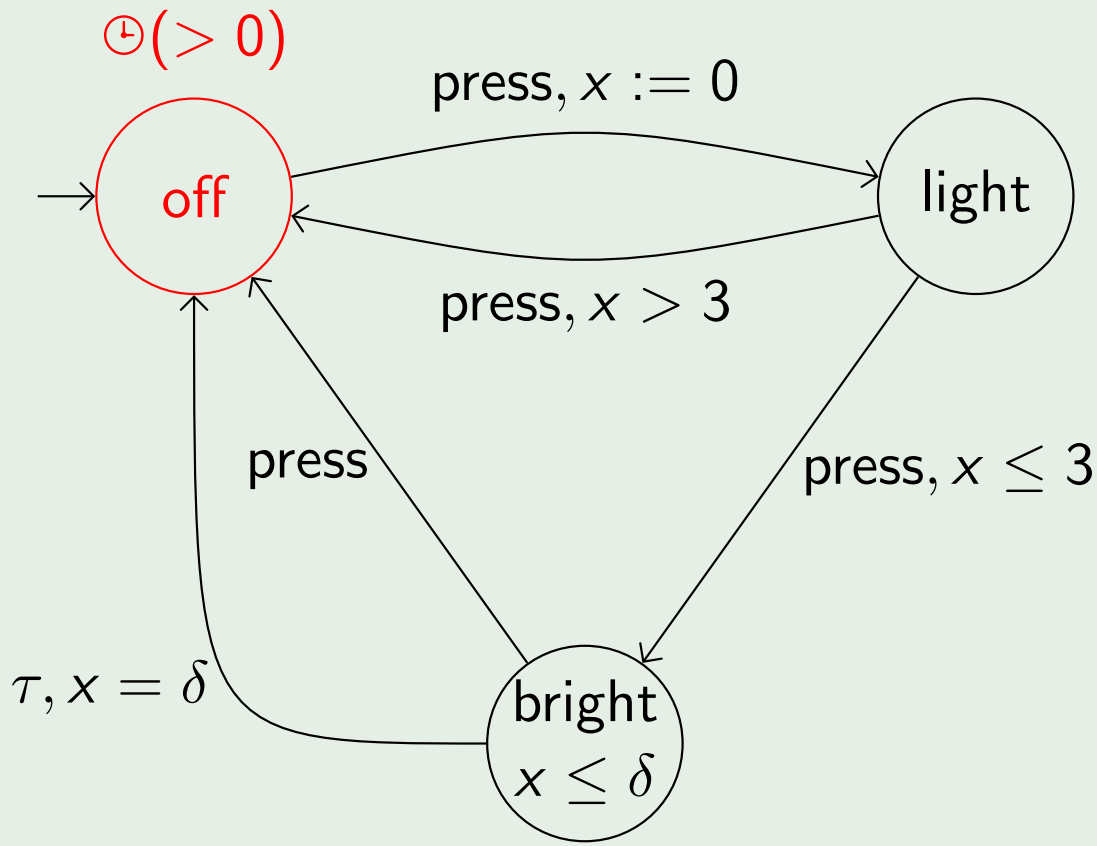
Example (Intelligent Light Controller)



Semantics = All possible traces

Timed Automata

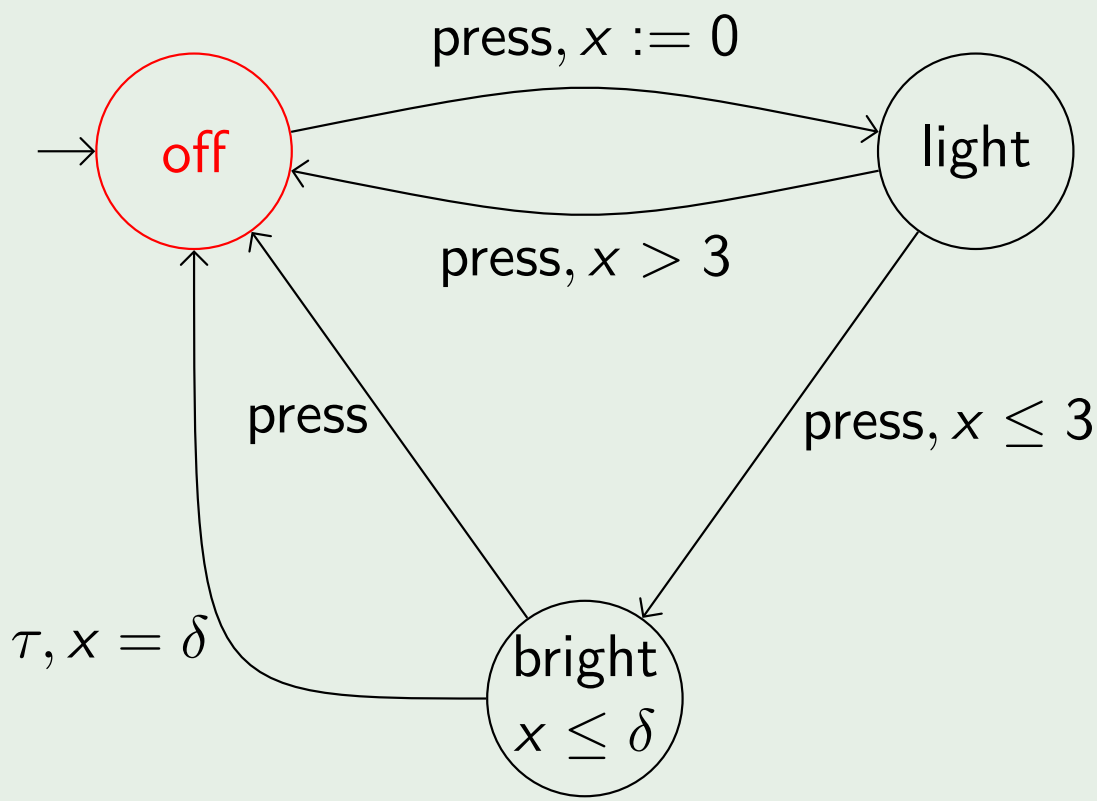
Example (Intelligent Light Controller)



Semantics = All possible traces

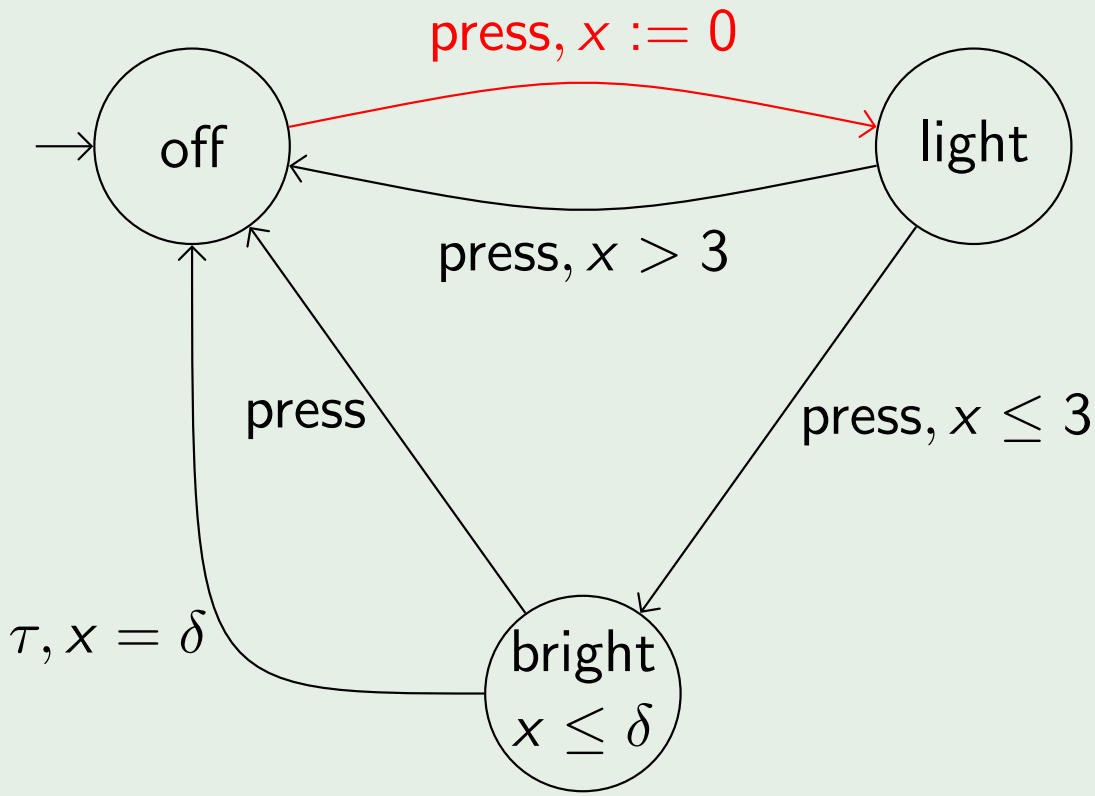
Timed Automata

Example (Intelligent Light Controller)



Semantics = All possible traces

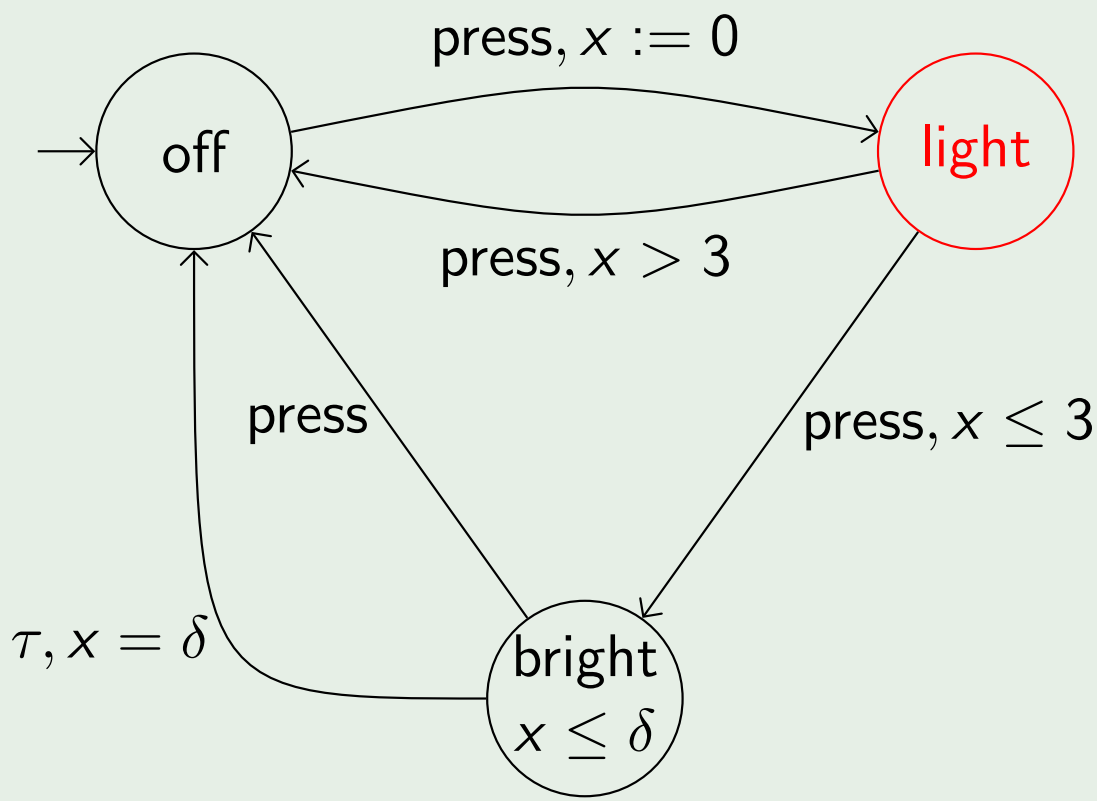
Example (Intelligent Light Controller)



Semantics = All possible traces

Timed Automata

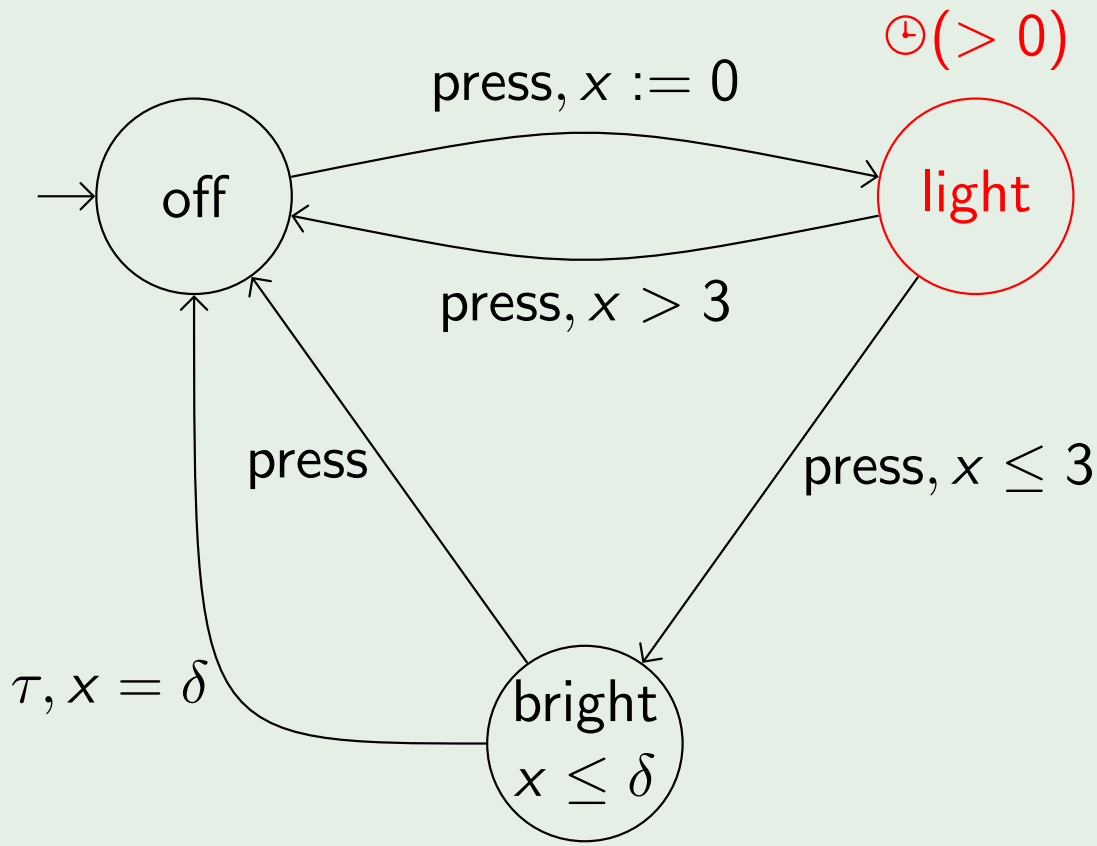
Example (Intelligent Light Controller)



Semantics = All possible traces

Timed Automata

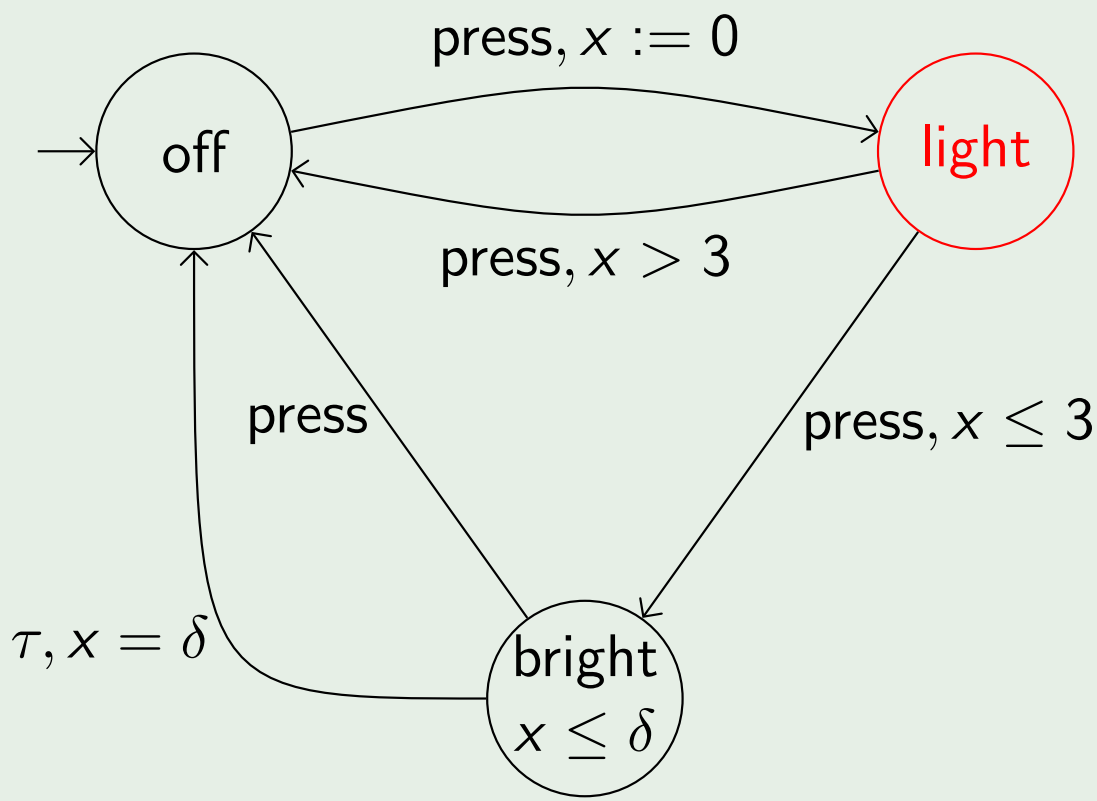
Example (Intelligent Light Controller)



Semantics = All possible traces

Timed Automata

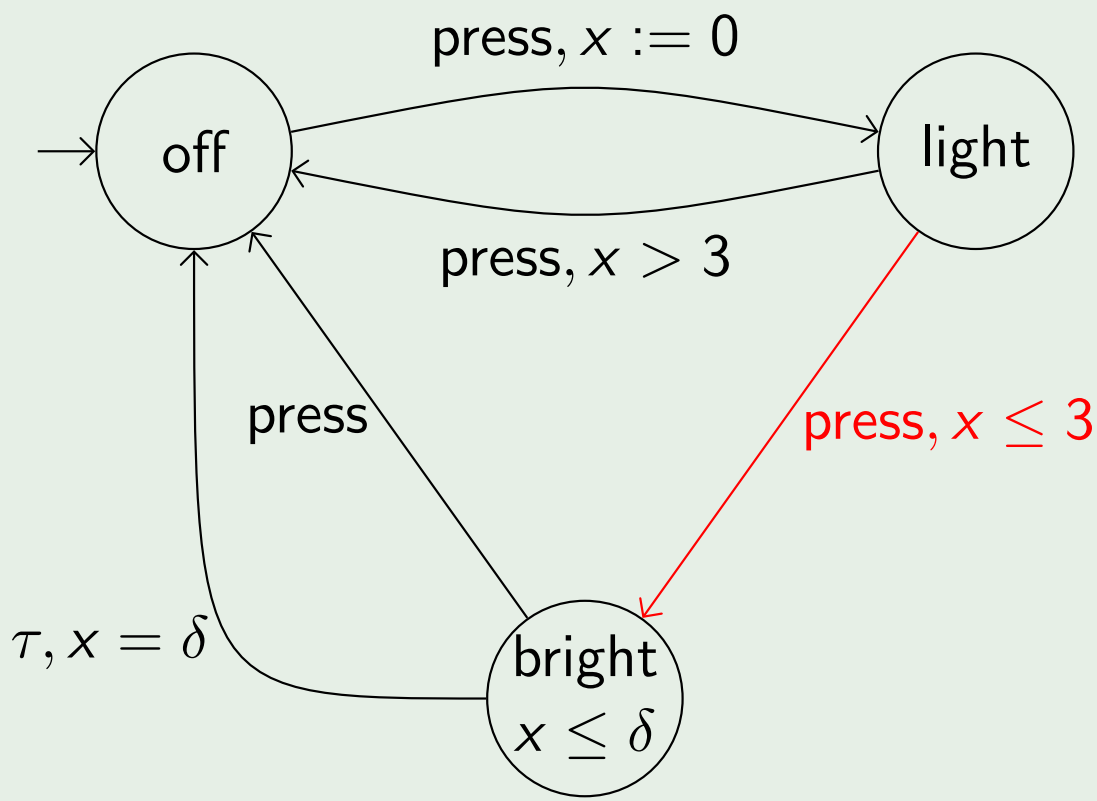
Example (Intelligent Light Controller)



Semantics = All possible traces

Timed Automata

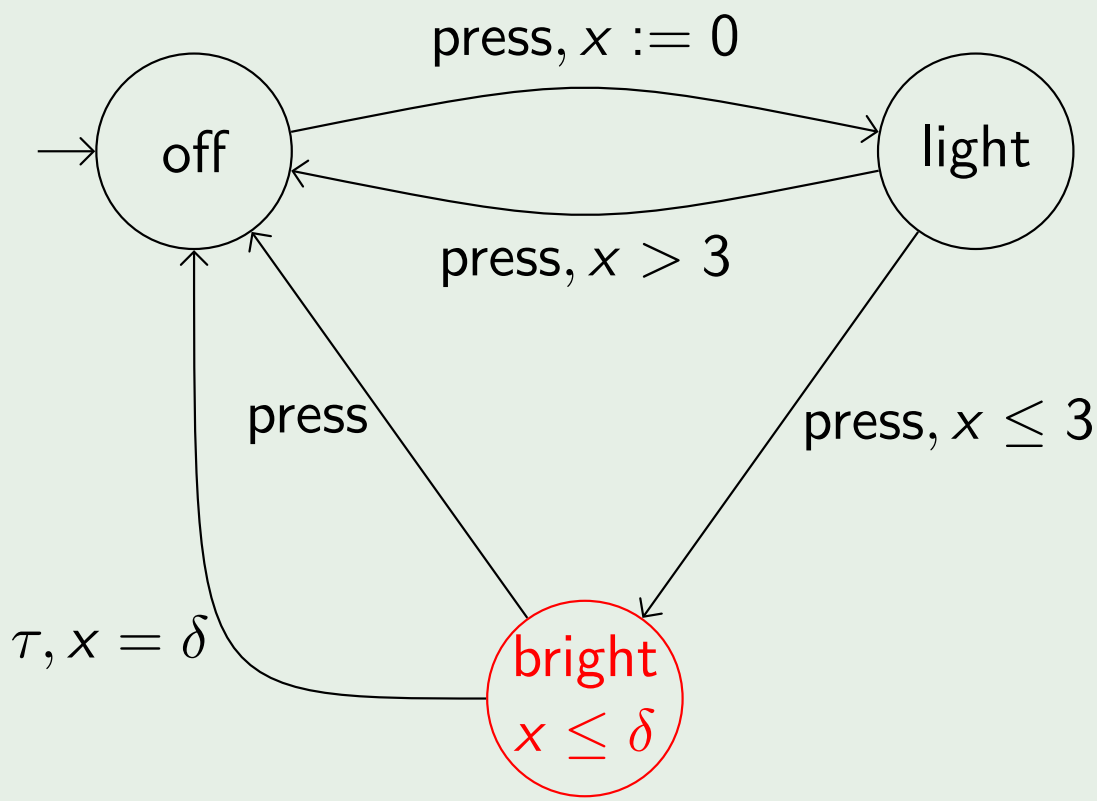
Example (Intelligent Light Controller)



Semantics = All possible traces

Timed Automata

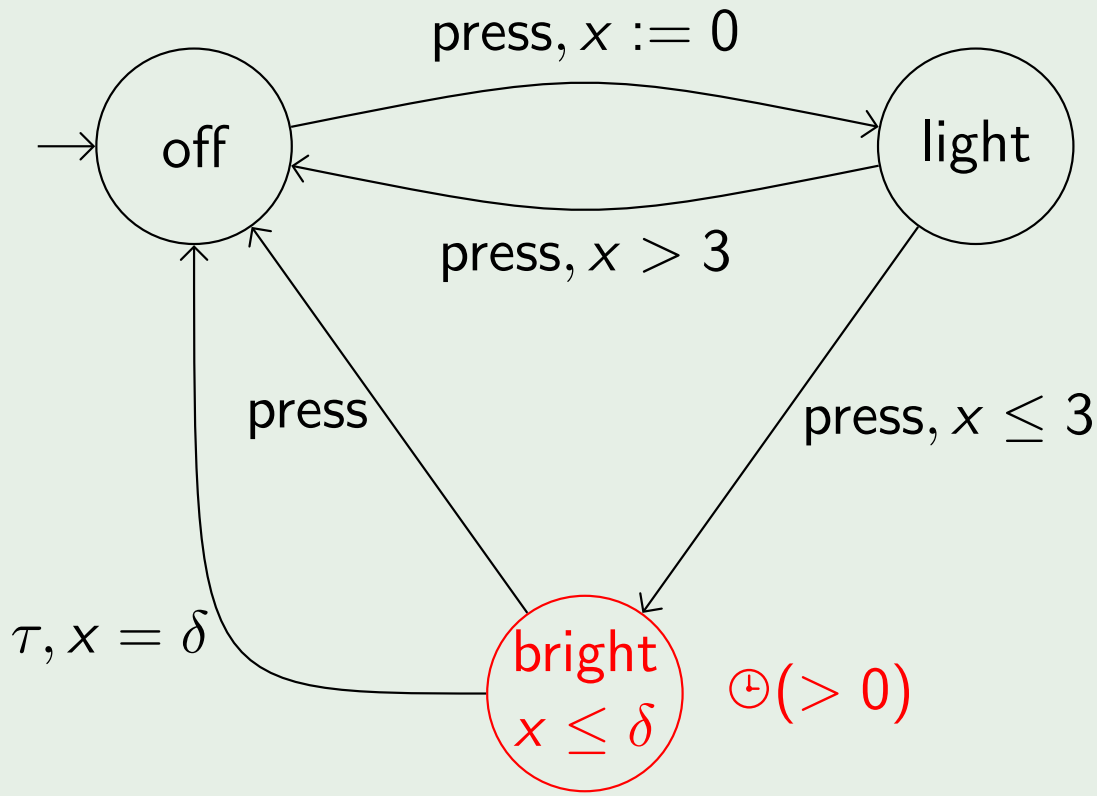
Example (Intelligent Light Controller)



Semantics = All possible traces

Timed Automata

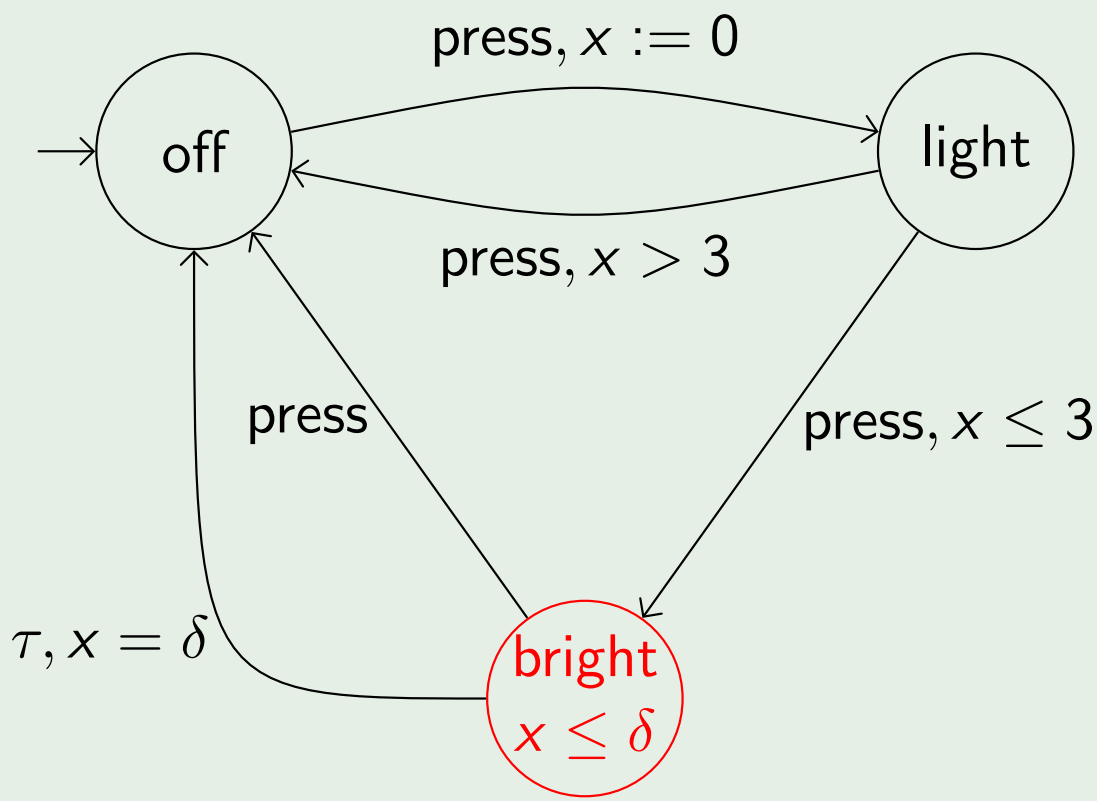
Example (Intelligent Light Controller)



Semantics = All possible traces

Timed Automata

Example (Intelligent Light Controller)

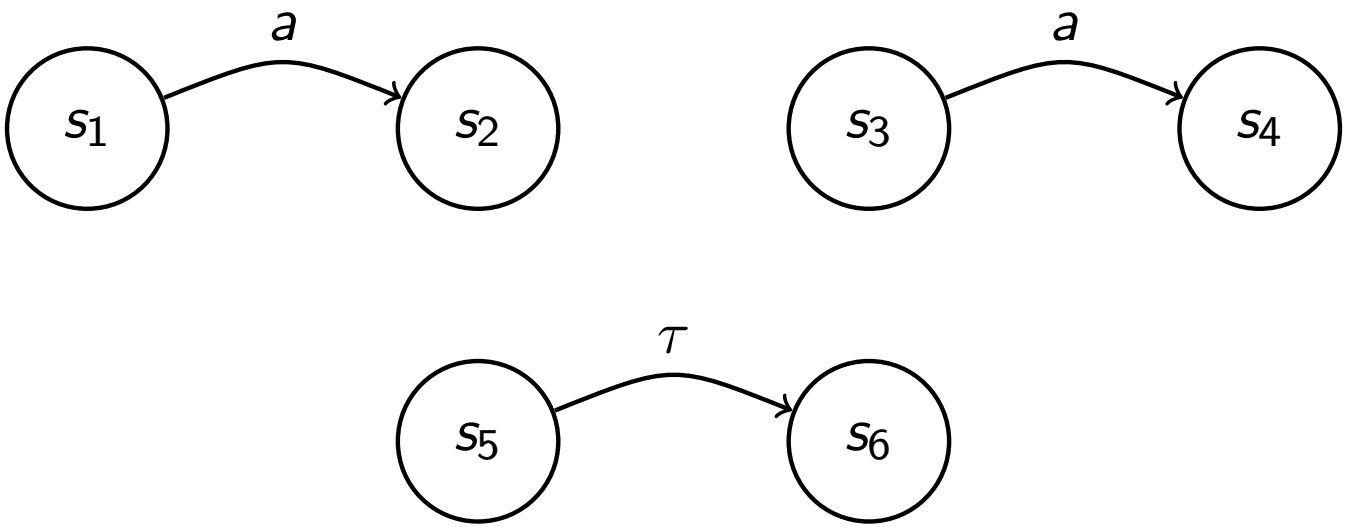


Semantics = All possible traces

details

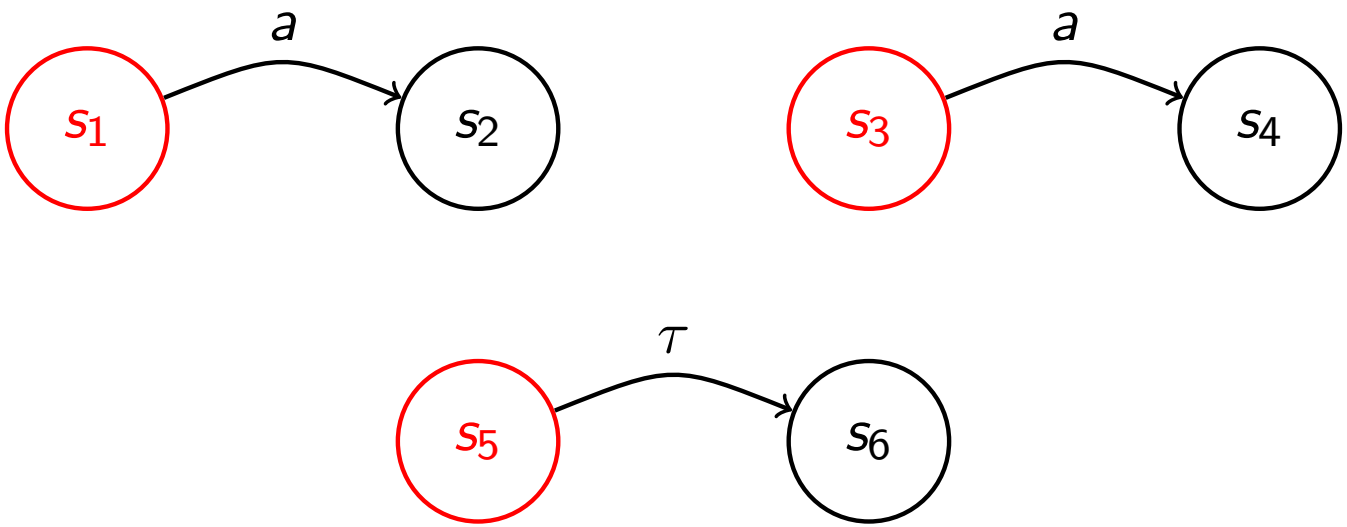
Timed Automata: Systems

- Synchronisation step (or zero delay)
- Internal step (or zero delay)
- Delay step



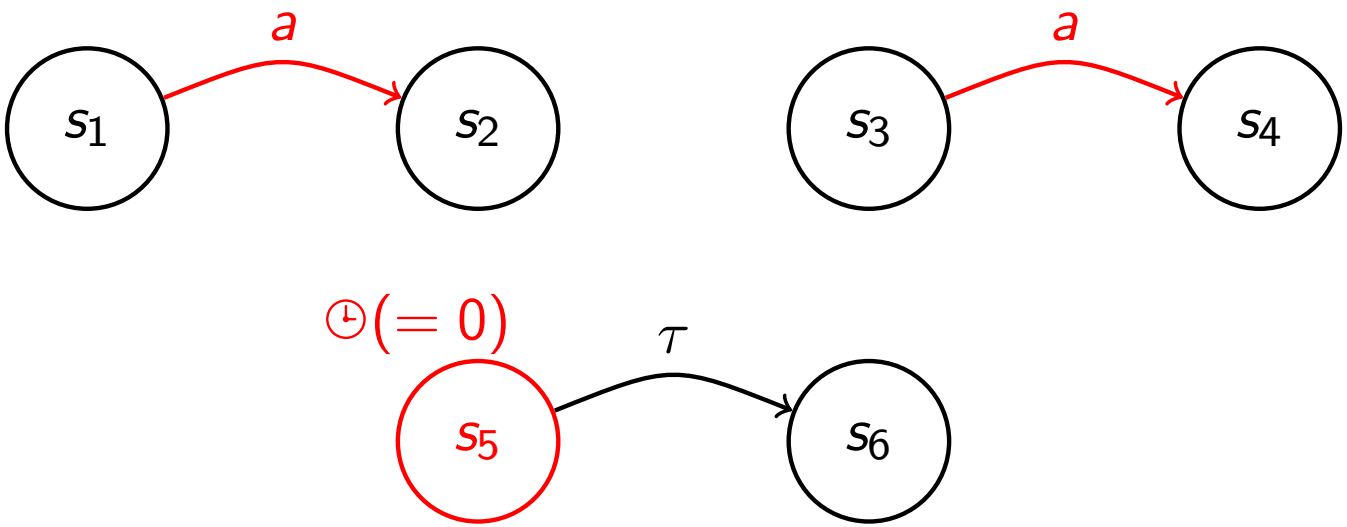
Timed Automata: Systems

- Synchronisation step (or zero delay)
- Internal step (or zero delay)
- Delay step



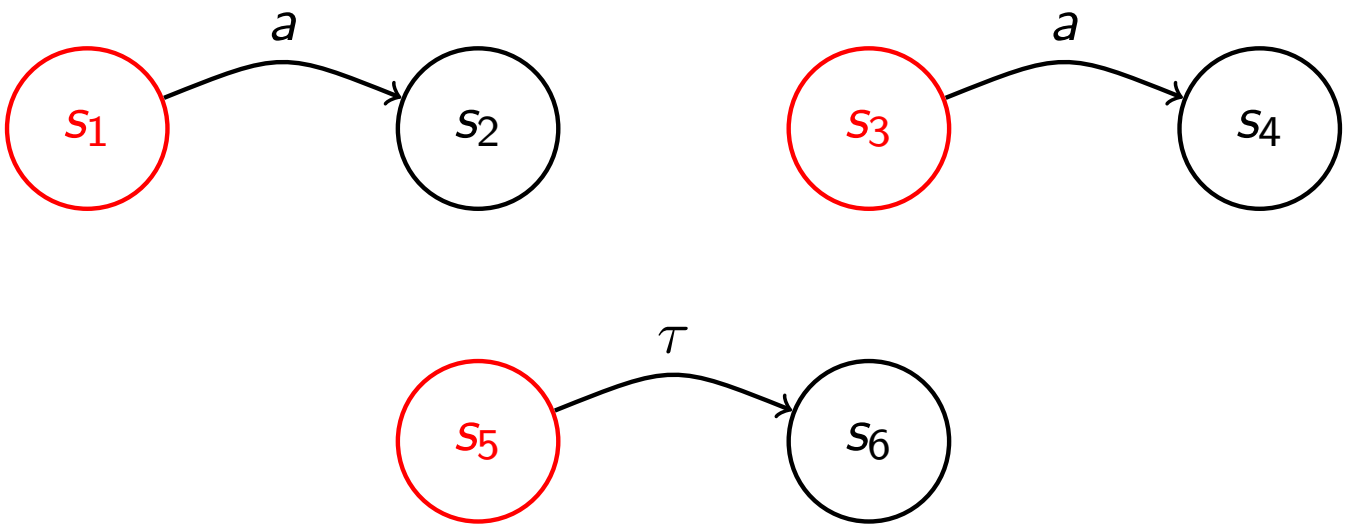
Timed Automata: Systems

- Synchronisation step (or zero delay)
- Internal step (or zero delay)
- Delay step



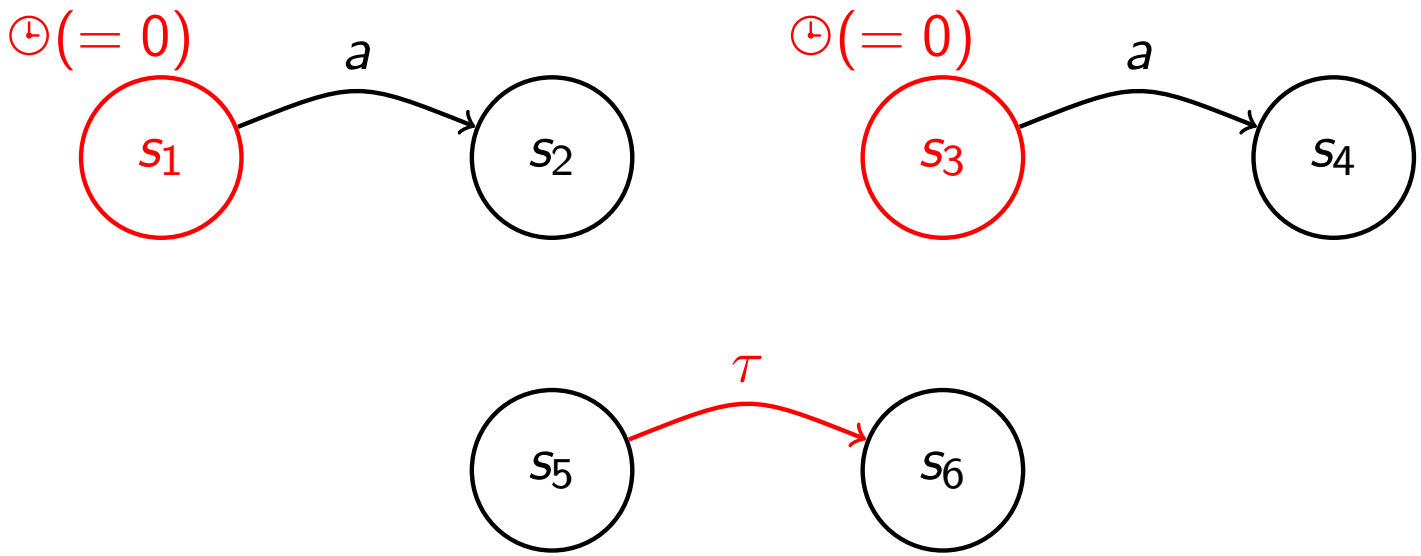
Timed Automata: Systems

- Synchronisation step (or zero delay)
- Internal step (or zero delay)
- Delay step



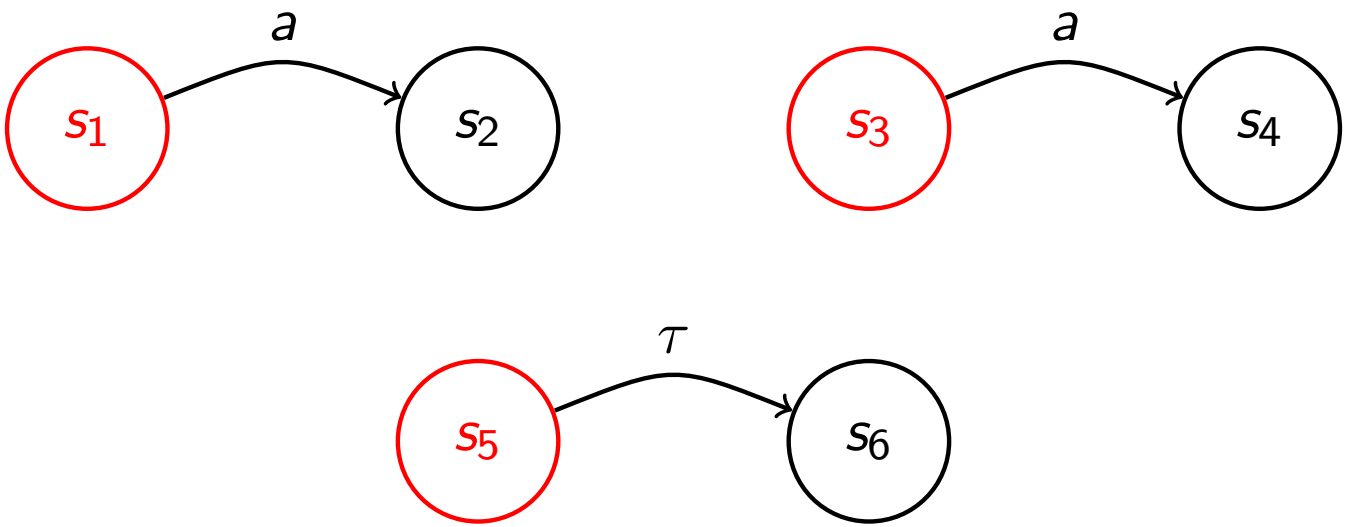
Timed Automata: Systems

- Synchronisation step (or zero delay)
- Internal step (or zero delay)
- Delay step



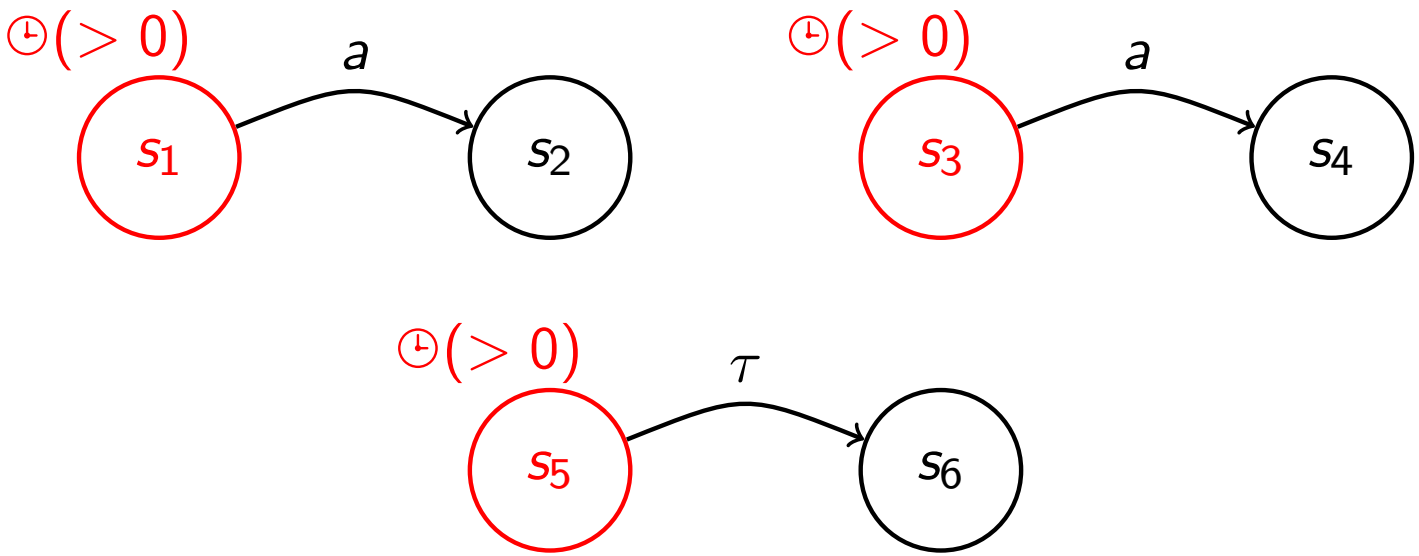
Timed Automata: Systems

- Synchronisation step (or zero delay)
- Internal step (or zero delay)
- Delay step

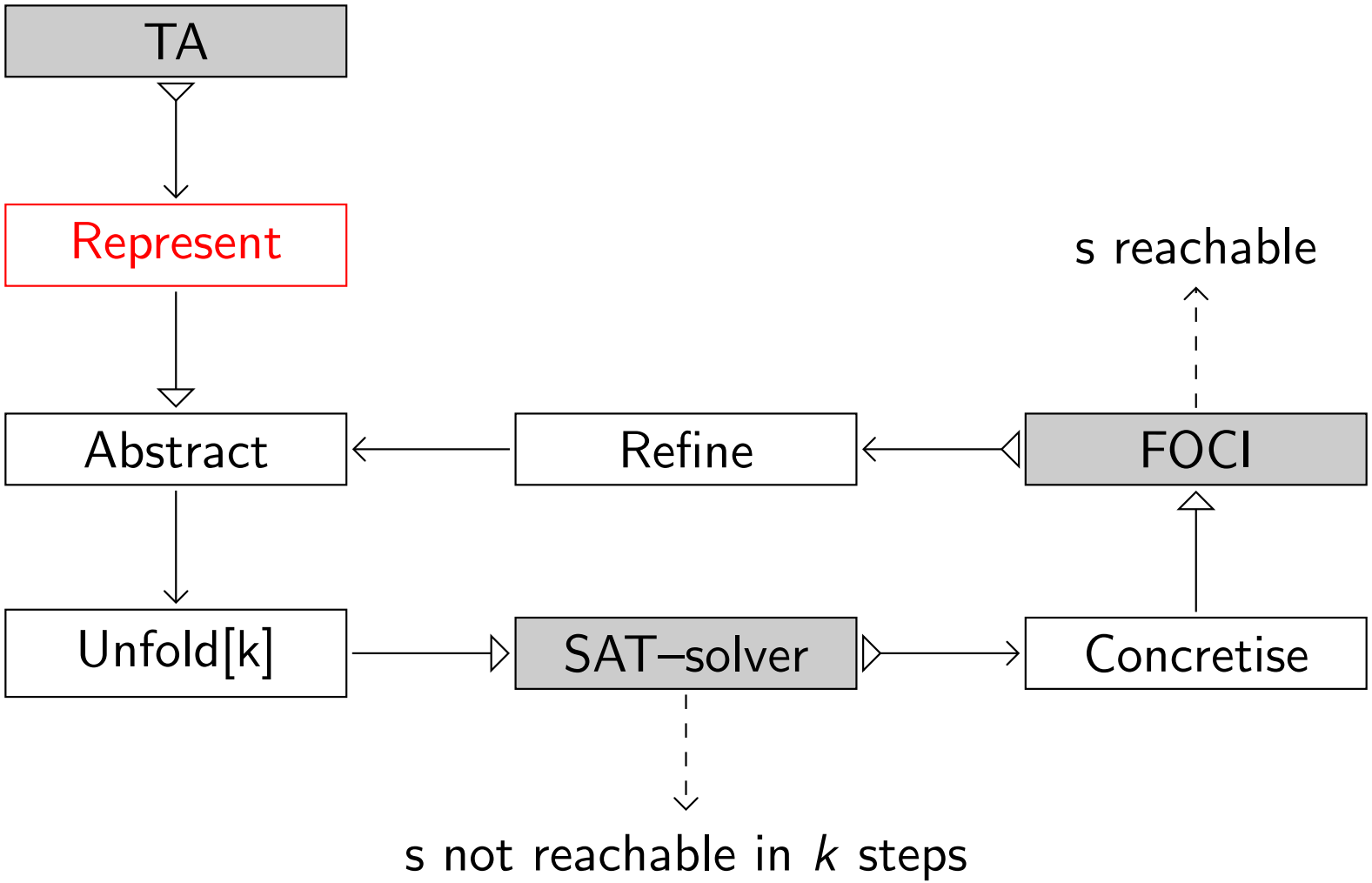


Timed Automata: Systems

- Synchronisation step (or zero delay)
- Internal step (or zero delay)
- Delay step



Representation

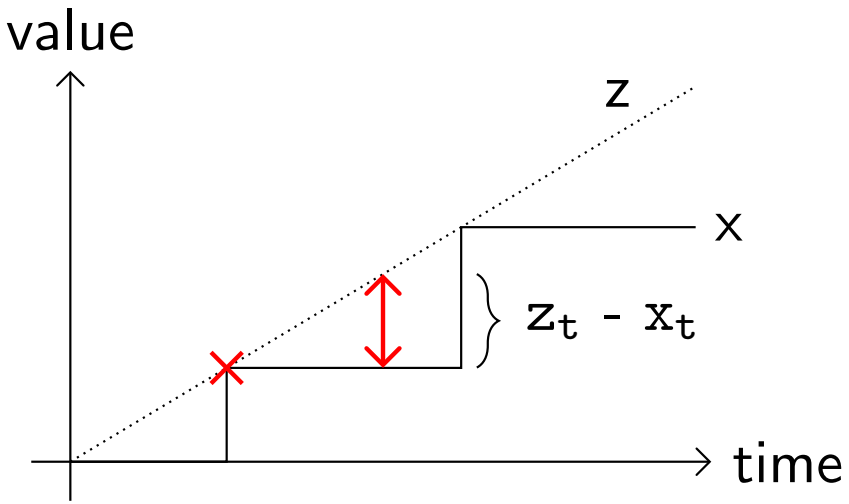


Representation: Basic Components

	TA	formula
state	s	$\mathtt{s}_0, \mathtt{s}_1, \mathtt{s}_2, \ldots$: TA at s in step i
clock	x	$\mathtt{x}_0, \mathtt{x}_1, \mathtt{x}_2, \ldots$: Time where x was last reset
value of clock	x	$\mathtt{z}_t$ - $\mathtt{x}_t$

Representation: Basic Components

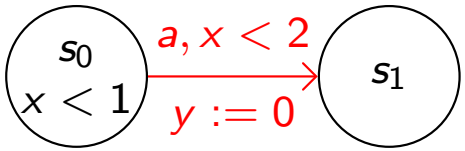
	TA	formula
state	s	$s_0, s_1, s_2, \ldots$: TA at s in step i
clock	x	$x_0, x_1, x_2, \ldots$: Time where x was last reset
value of clock	x	z_t - x_t



why z_t?

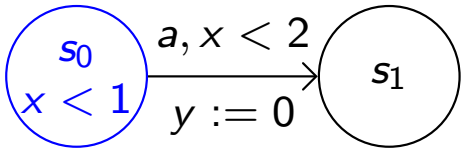
Representation: Transitions

Action transition:



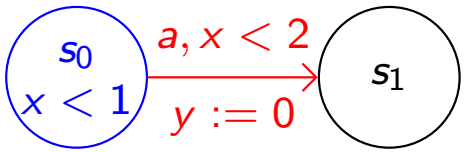
$$\mathtt{s0}_t \wedge \mathtt{s1}_{t+1} \wedge \mathtt{a}_t \wedge (\mathtt{z}_t - \mathtt{x}_t < 2) \wedge (\mathtt{z}_t = \mathtt{z}_{t+1}) \wedge (\mathtt{x}_{t+1} = \mathtt{x}_t) \wedge (\mathtt{y}_{t+1} = \mathtt{z}_{t+1})$$

Delay transition:



$$\mathtt{s0}_t \wedge \mathtt{s0}_{t+1} \wedge (\mathtt{z}_t < \mathtt{z}_{t+1}) \wedge (\mathtt{x}_t = \mathtt{x}_{t+1}) \wedge (\mathtt{y}_t = \mathtt{y}_{t+1}) \wedge \neg \mathtt{a}_t \wedge \neg \mathtt{b}_t$$

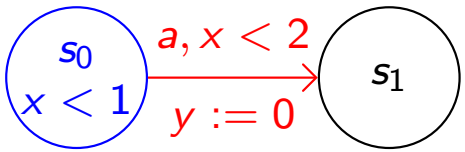
Transition choice:



$$(\mathtt{s0}_t \wedge \mathtt{s1}_{t+1} \wedge \ldots) \vee (\mathtt{s0}_t \wedge \mathtt{s0}_{t+1} \wedge \ldots)$$

Representation: Automaton

Transition choice:

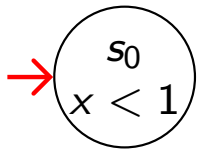


$$(\mathtt{s0}_t \wedge \mathtt{s1}_{t+1} \wedge \ldots) \vee$$
$$(\mathtt{s0}_t \wedge \mathtt{s0}_{t+1} \wedge \ldots)$$

Mutual exclusion:

$$\neg(\mathtt{s0}_t \wedge \mathtt{s1}_t)$$
$$\mathtt{s0}_t \rightarrow (\mathtt{z}_t - \mathtt{x}_t < 1)$$
$$\neg(\mathtt{a}_t \wedge \mathtt{b}_t) \wedge \neg(\mathtt{a}_t \wedge \tau_t) \wedge \neg(\mathtt{b}_t \wedge \tau_t)$$

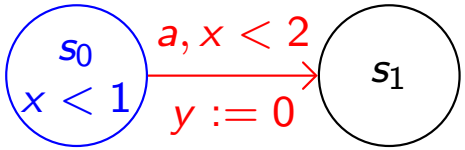
Initial constraints:



$$\mathtt{s0}_0 \wedge (\mathtt{z}_0 = 0) \wedge (\mathtt{x}_0 = 0) \wedge (\mathtt{y}_0 = 0)$$

Representation: Automaton

Transition choice:

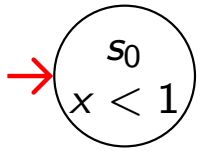


$$(\mathtt{s0}_t \wedge \mathtt{s1}_{t+1} \wedge \ldots) \vee$$
$$(\mathtt{s0}_t \wedge \mathtt{s0}_{t+1} \wedge \ldots)$$

Mutual exclusion:

$$\neg(\mathtt{s0}_t \wedge \mathtt{s1}_t)$$
$$\mathtt{s0}_t \rightarrow (\mathtt{z}_t - \mathtt{x}_t < 1)$$
$$\neg(\mathtt{a}_t \wedge \mathtt{b}_t) \wedge \neg(\mathtt{a}_t \wedge \tau_t) \wedge \neg(\mathtt{b}_t \wedge \tau_t)$$

Initial constraints:



$$\mathtt{s0}_0 \wedge (\mathtt{z}_0 = 0) \wedge (\mathtt{x}_0 = 0) \wedge (\mathtt{y}_0 = 0)$$

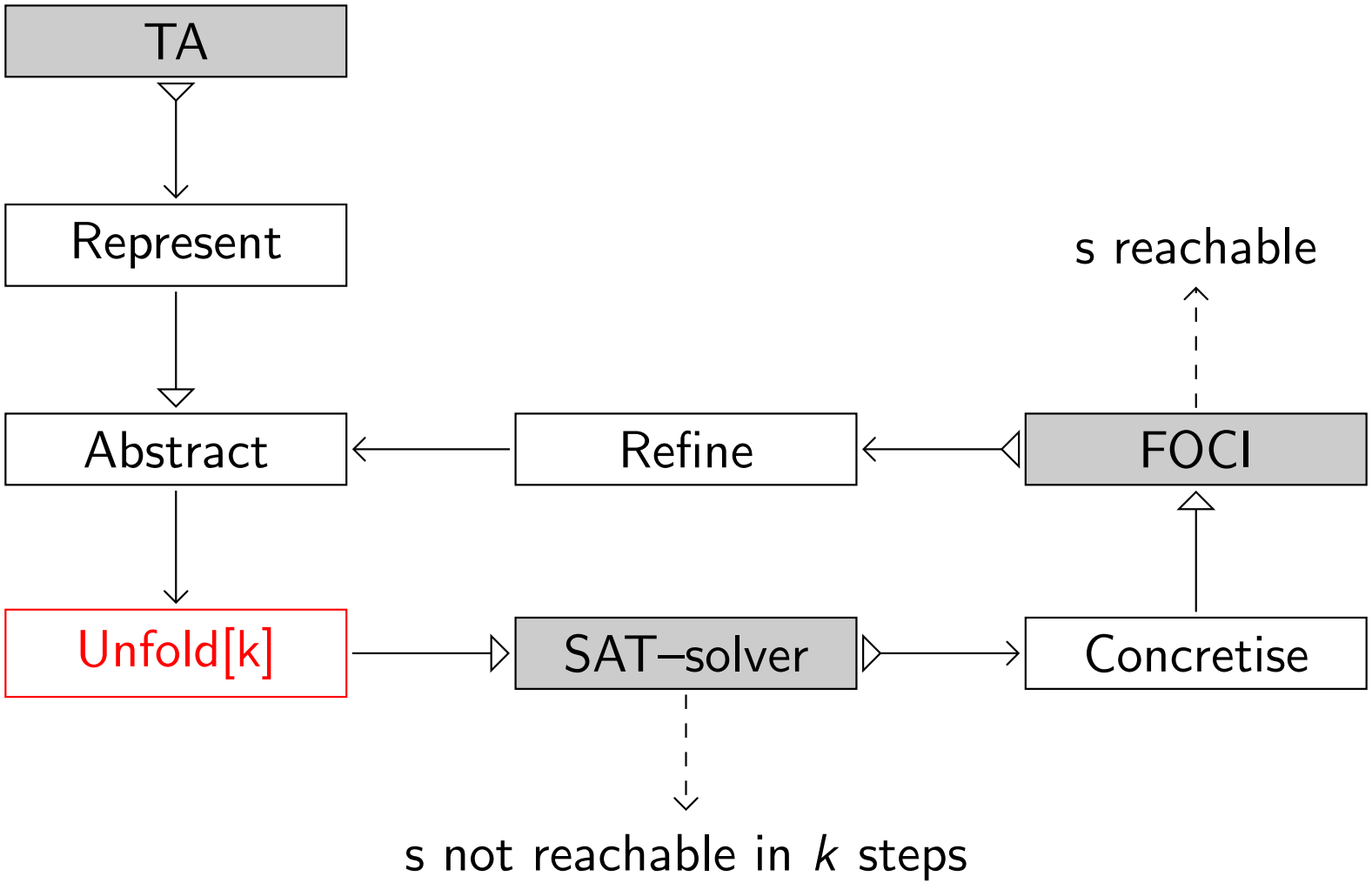
$\Big\}\ \varphi(\mathcal{A})$

Representation: Parallel Systems

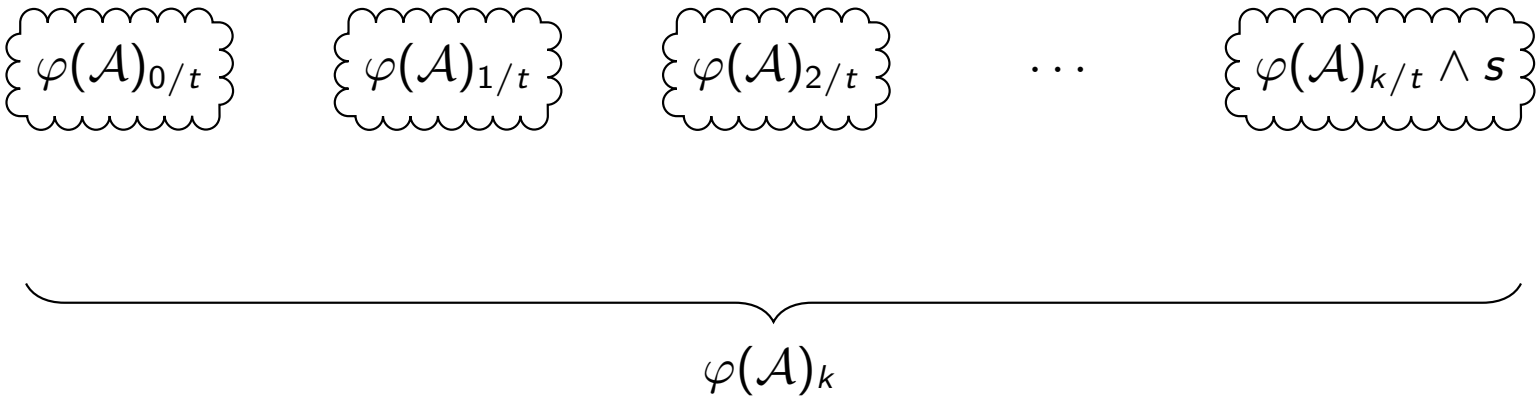
$$\varphi(A_1 \parallel A_2 \parallel \ldots \parallel A_n)$$
$$=$$
$$\varphi(\mathcal{A}_1) \wedge \varphi(\mathcal{A}_2) \wedge \ldots \wedge \varphi(\mathcal{A}_n)$$

▽ Product automaton representation is linear!

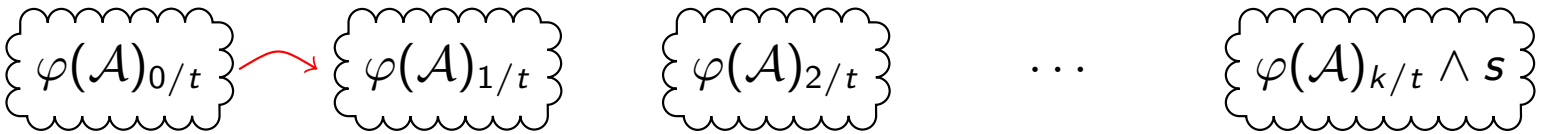
Unfolding


TA
Represent
Abstract
Refine
FOCI
s reachable
Unfold[k]
SAT–solver
Concretise
s not reachable in k steps

Unfolding: Timed Automata

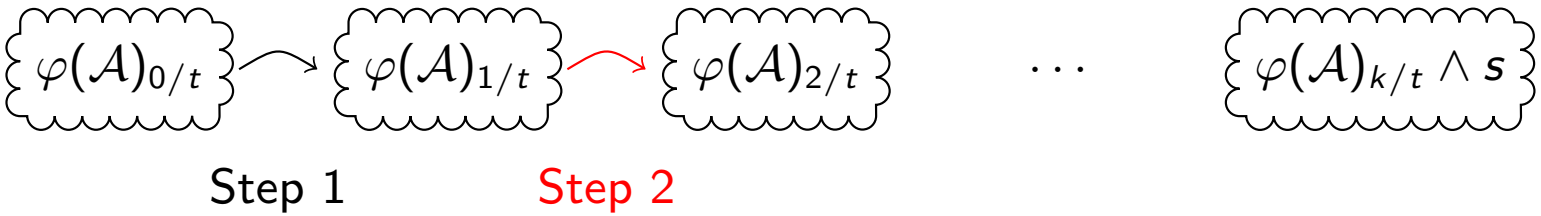


Unfolding: Timed Automata

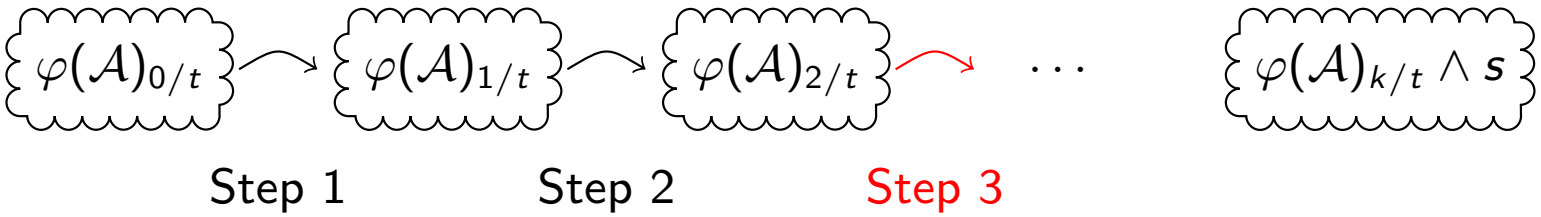


Step 1

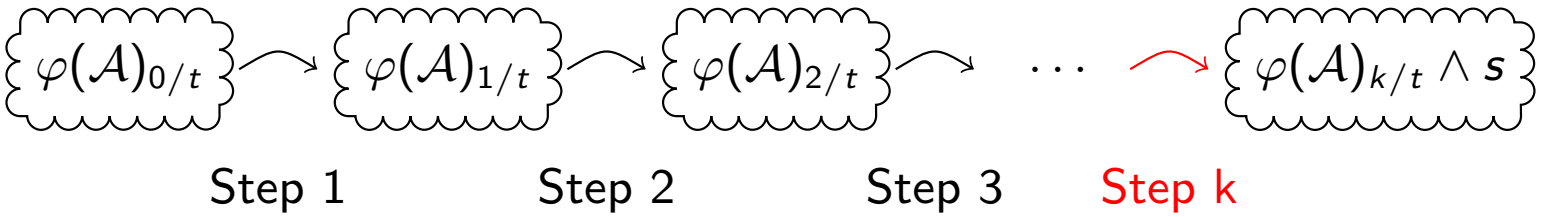
Unfolding: Timed Automata



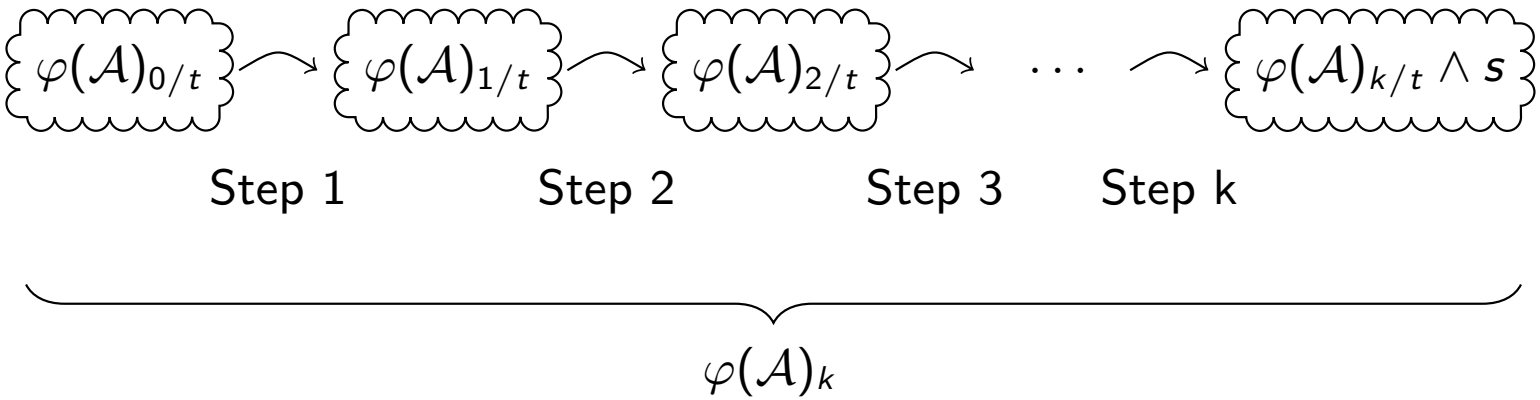
Unfolding: Timed Automata



Unfolding: Timed Automata



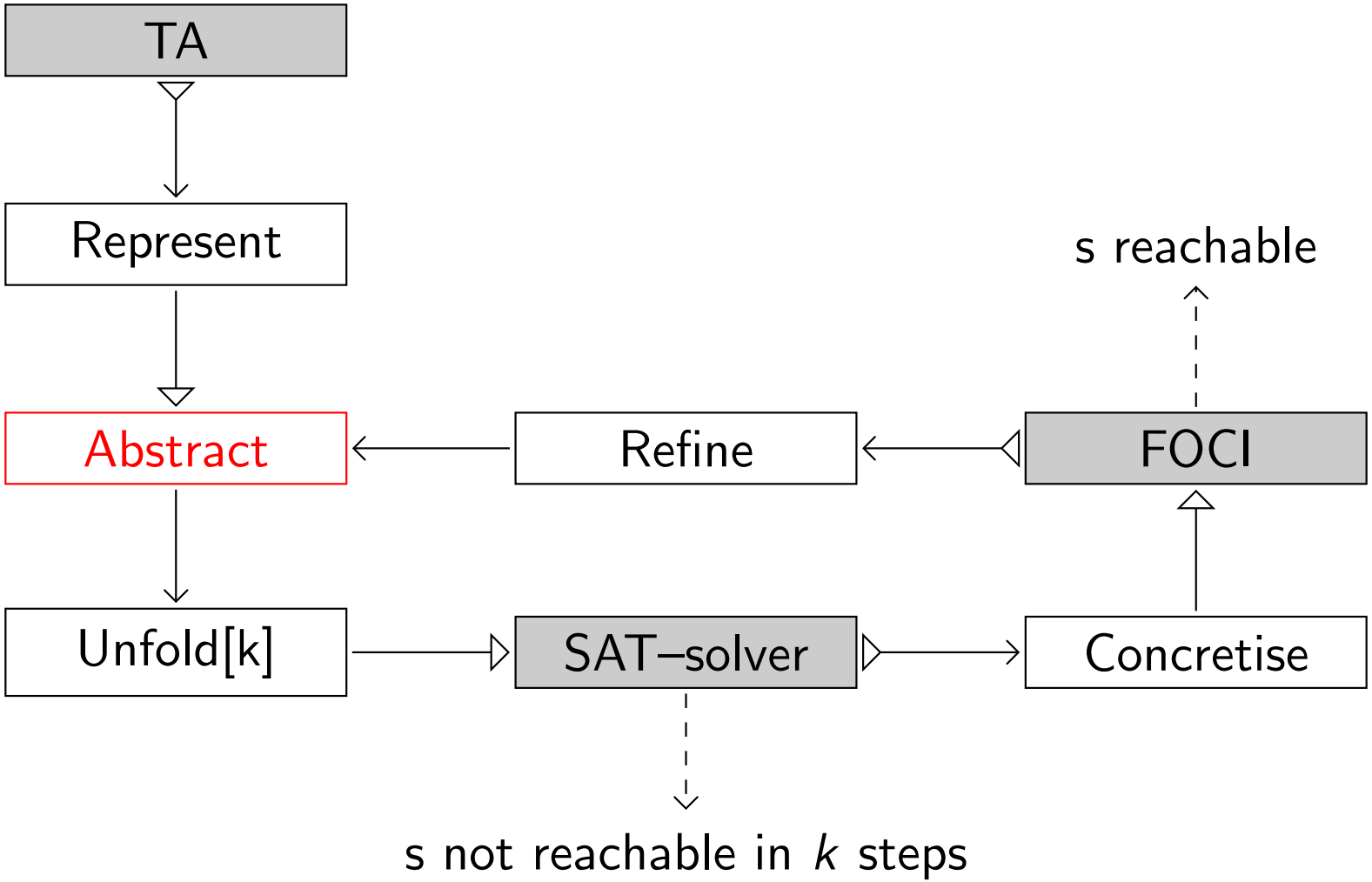
Unfolding: Timed Automata



Model of $\varphi(\mathcal{A})_k$ = Trace of $\mathcal{A}$ of length k

details

Abstraction



Abstraction: Idea

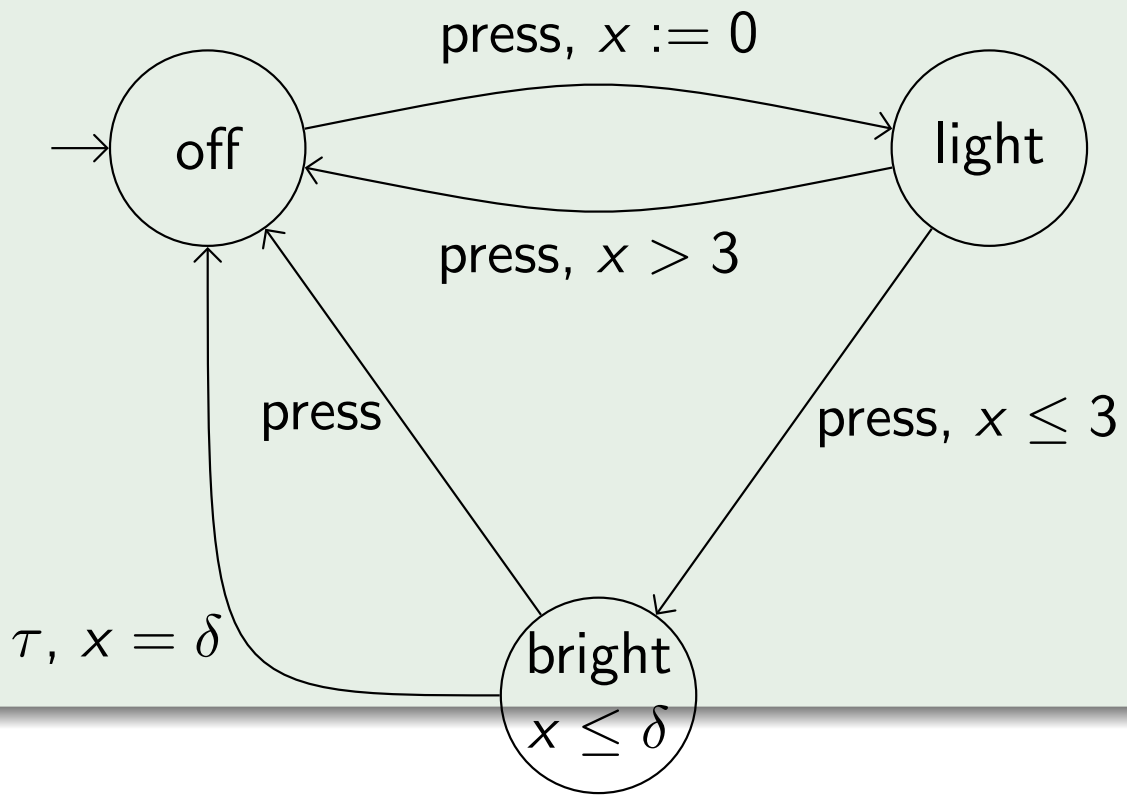
- More possible behaviour = less constraints
- Abstract system safe $\rightsquigarrow$ conrete system safe
- Approach: Fewer symbols in $\varphi(\mathcal{A})$ (more efficient)
- Abstraction by Merging Omission: $\alpha(\varphi(\mathcal{A}))$

details

Abstraction by Merging Omission: Example

Example (Abstraction by Omission)

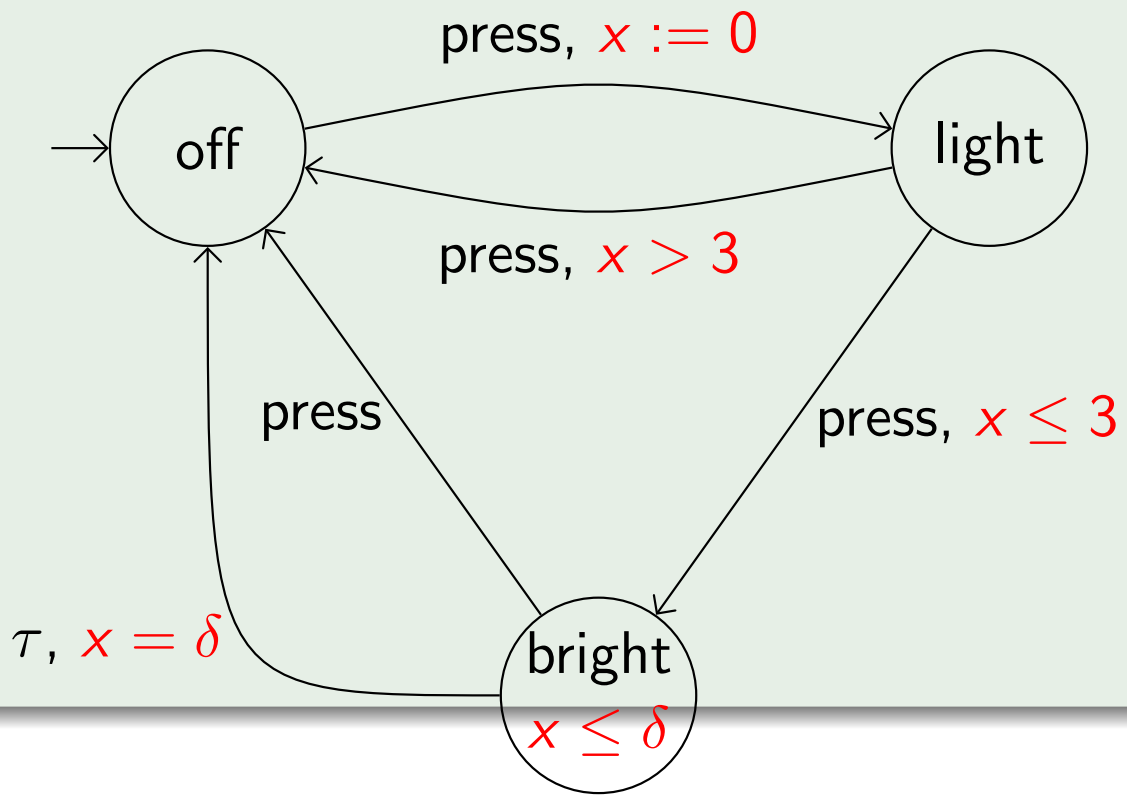
$\mathcal{AS} = \{x\}$



Abstraction by Merging Omission: Example

Example (Abstraction by Omission)

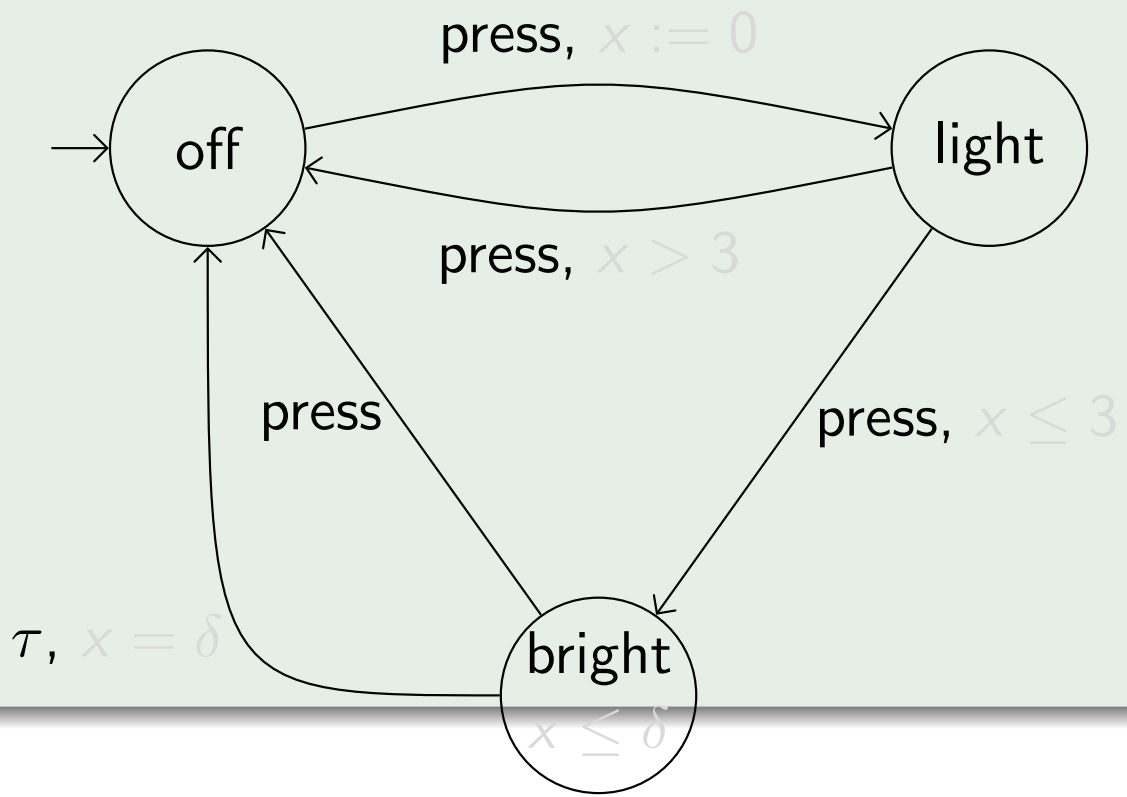
$\mathcal{AS} = \{x\}$



Abstraction by Merging Omission: Example

Example (Abstraction by Omission)

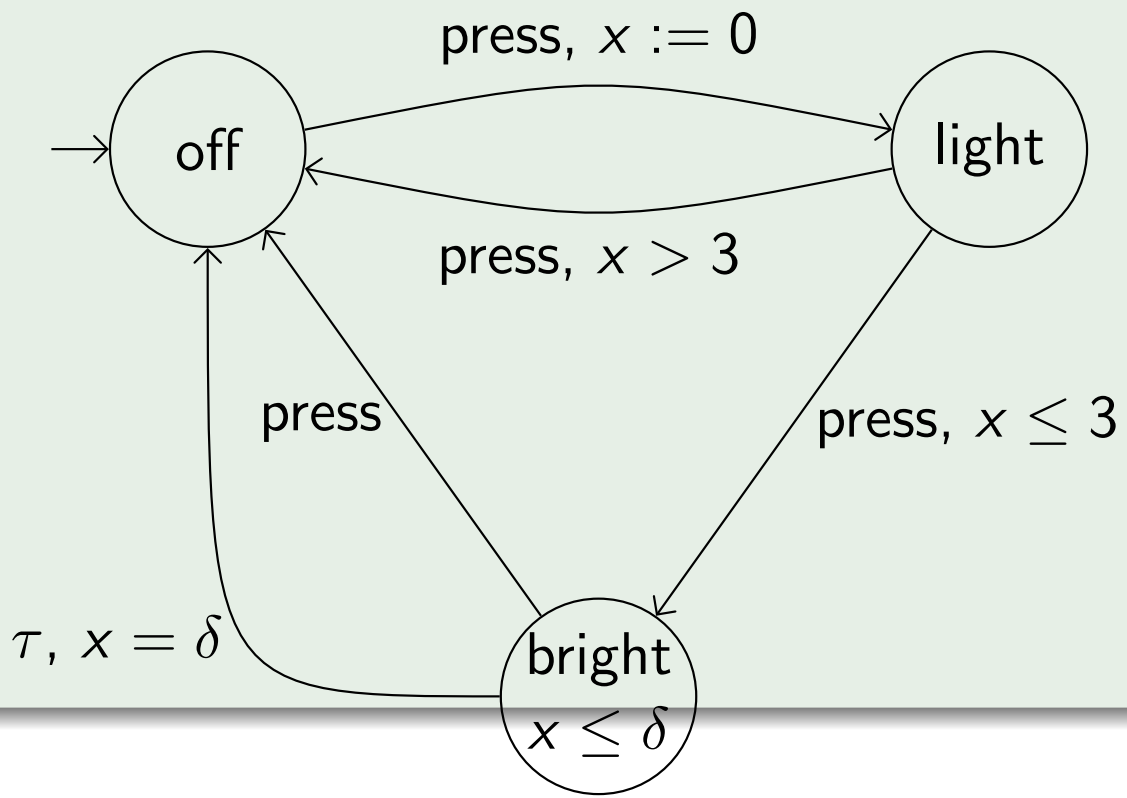
$\mathcal{AS} = \{x\}$



Abstraction by Merging Omission: Example

Example (Abstraction by Merging)

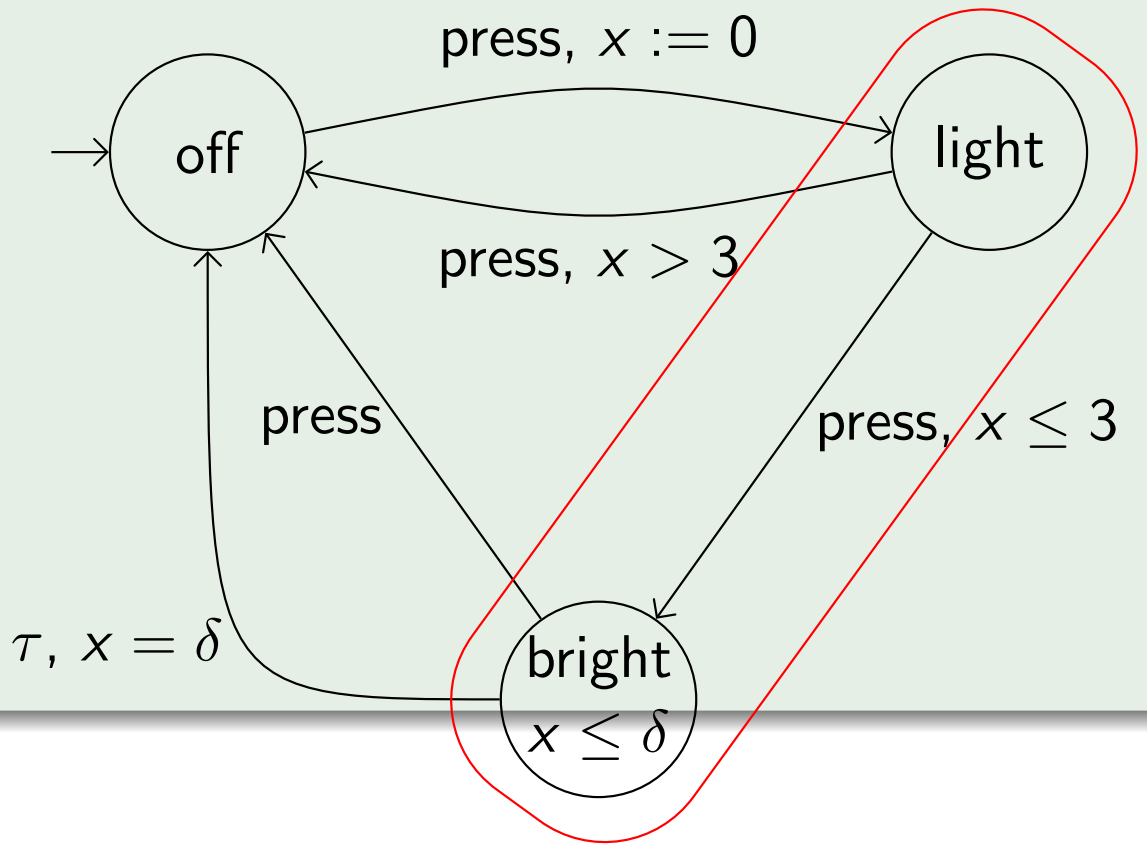
$\gamma(\text{light}) = \text{on}$
$\gamma(\text{bright}) = \text{on}$



Abstraction by Merging Omission: Example

Example (Abstraction by Merging)

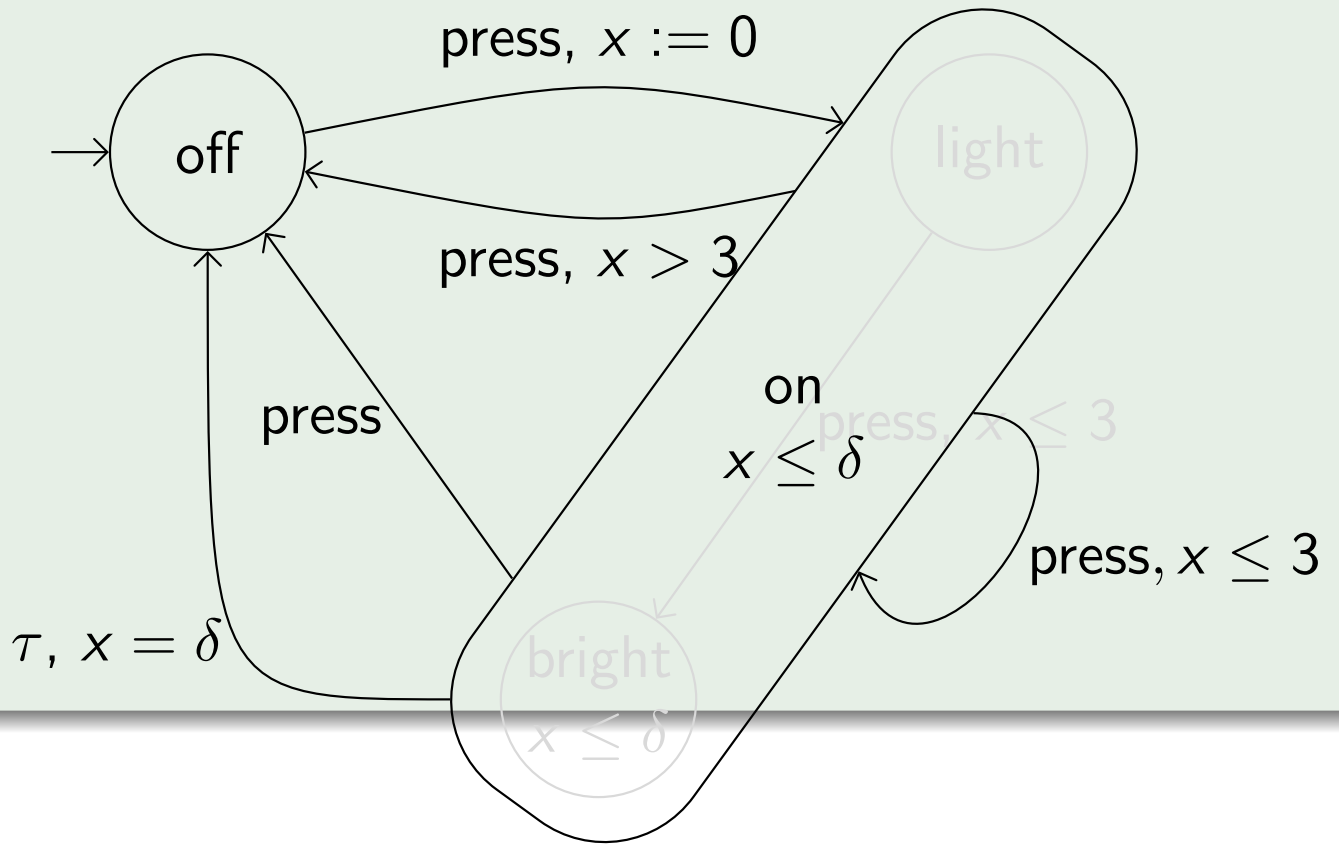
$\gamma(\text{light}) = \text{on}$
$\gamma(\text{bright}) = \text{on}$



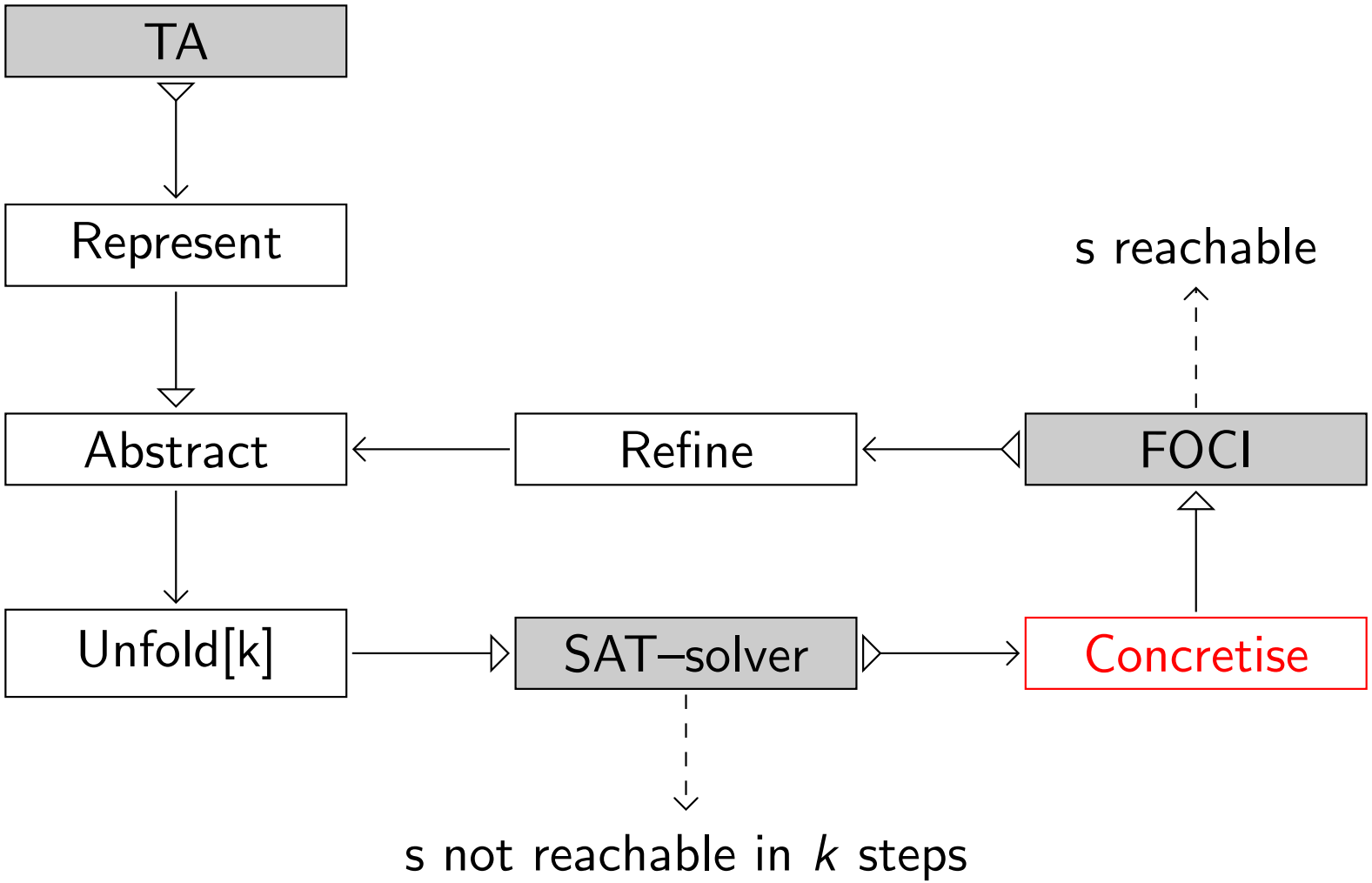
Abstraction by Merging Omission: Example

Example (Abstraction by Merging)

$\gamma(\text{light}) = \text{on}$
$\gamma(\text{bright}) = \text{on}$

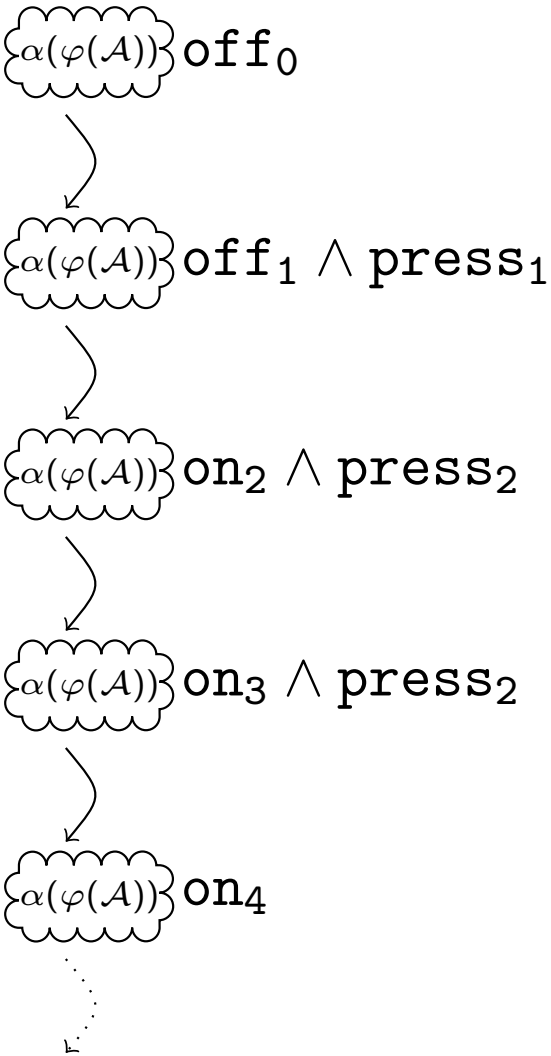


Concretisation



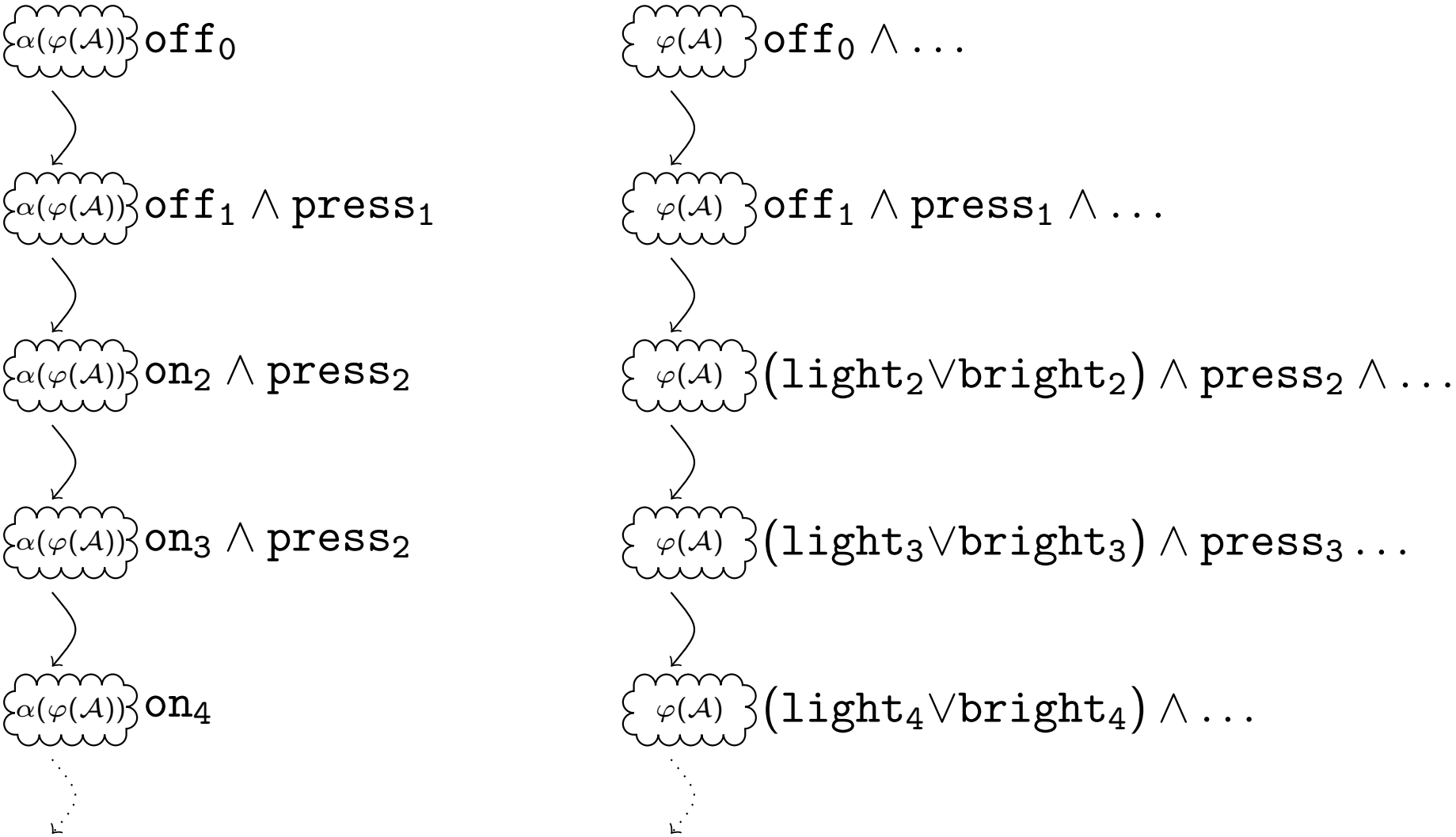
Concretisation: Idea

Translate abstract counterexample into concrete system

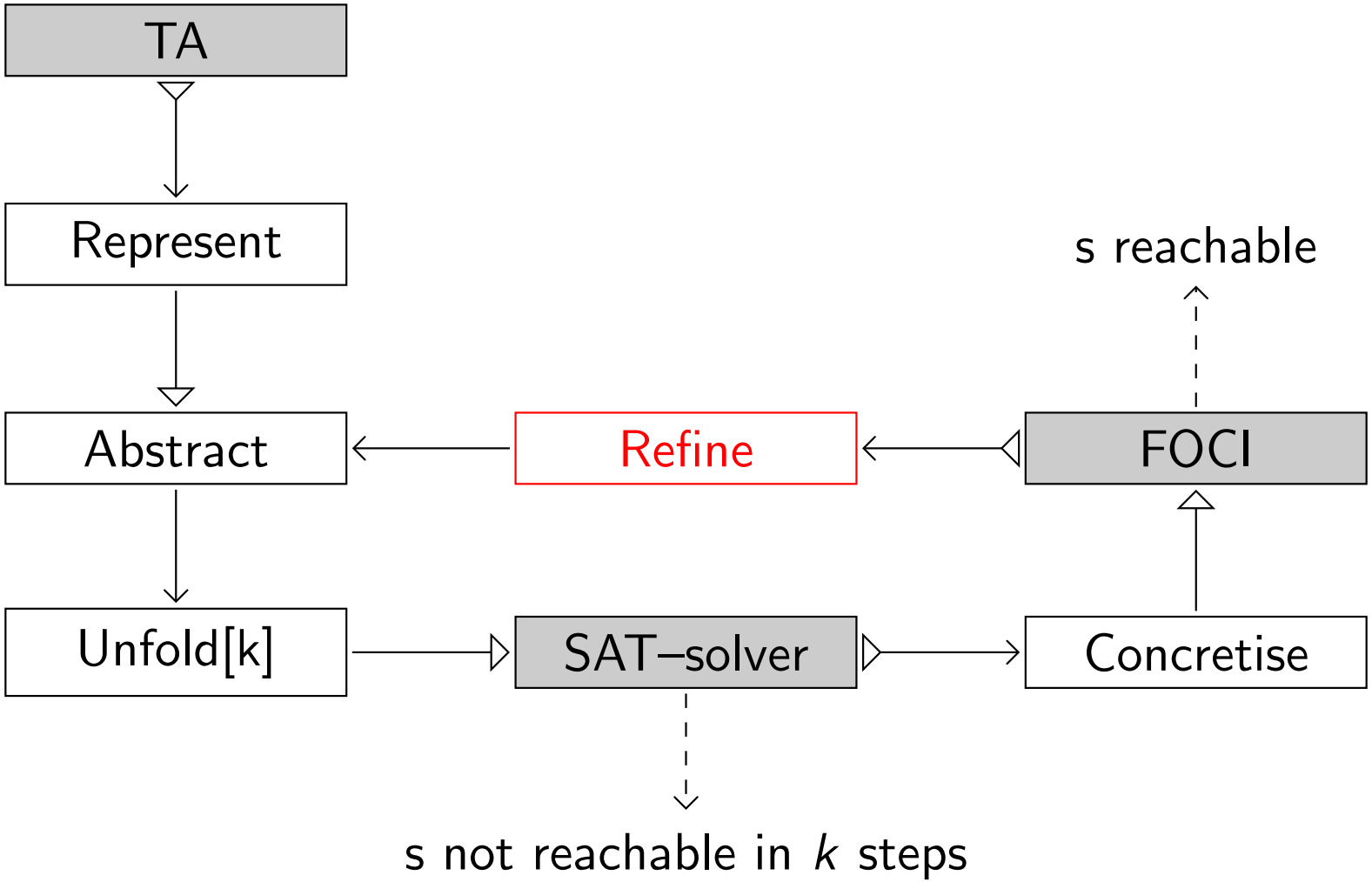


Concretisation: Idea

Translate abstract counterexample into concrete system



Refinement using Craig Interpolants



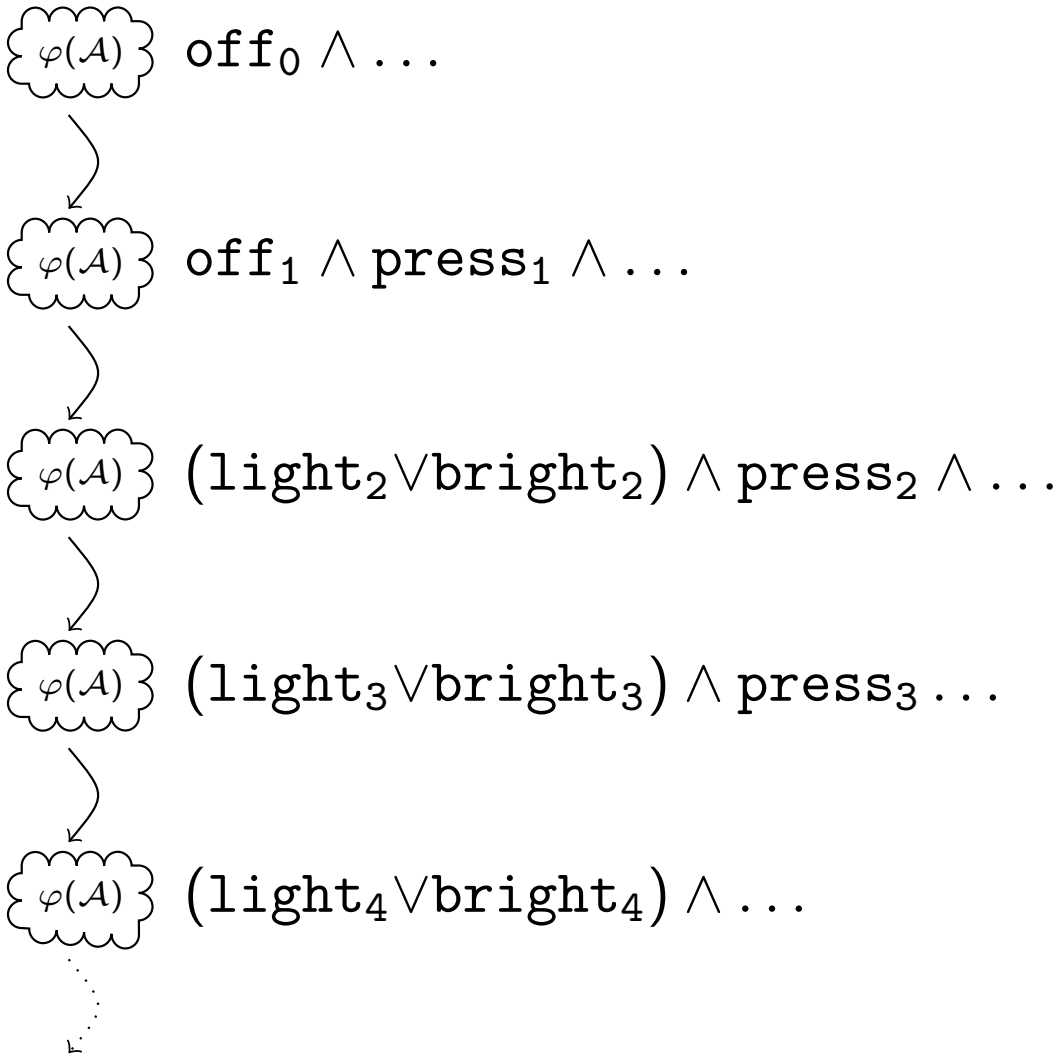
Craig Interpolants: Definition

Definition (Craig Interpolant)

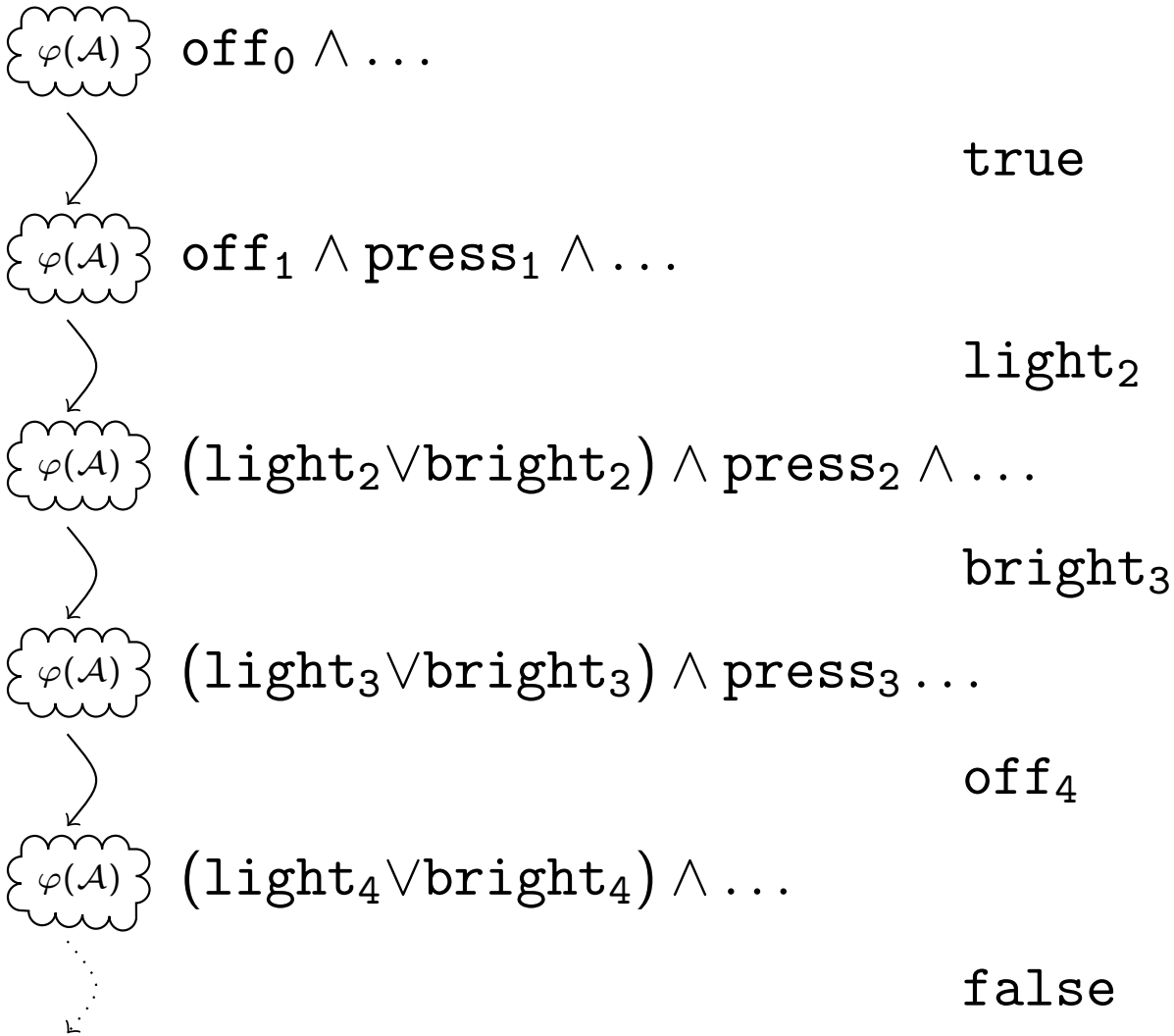
For $\models \neg(A \wedge B)$, Craig interpolant C iff

- $\models A \rightarrow C$ (Over-approximation of prefix)
- $\models \neg(C \wedge B)$ (Under-approximation of neg. suffix)
- Contains only common symbols (Only relevant information)

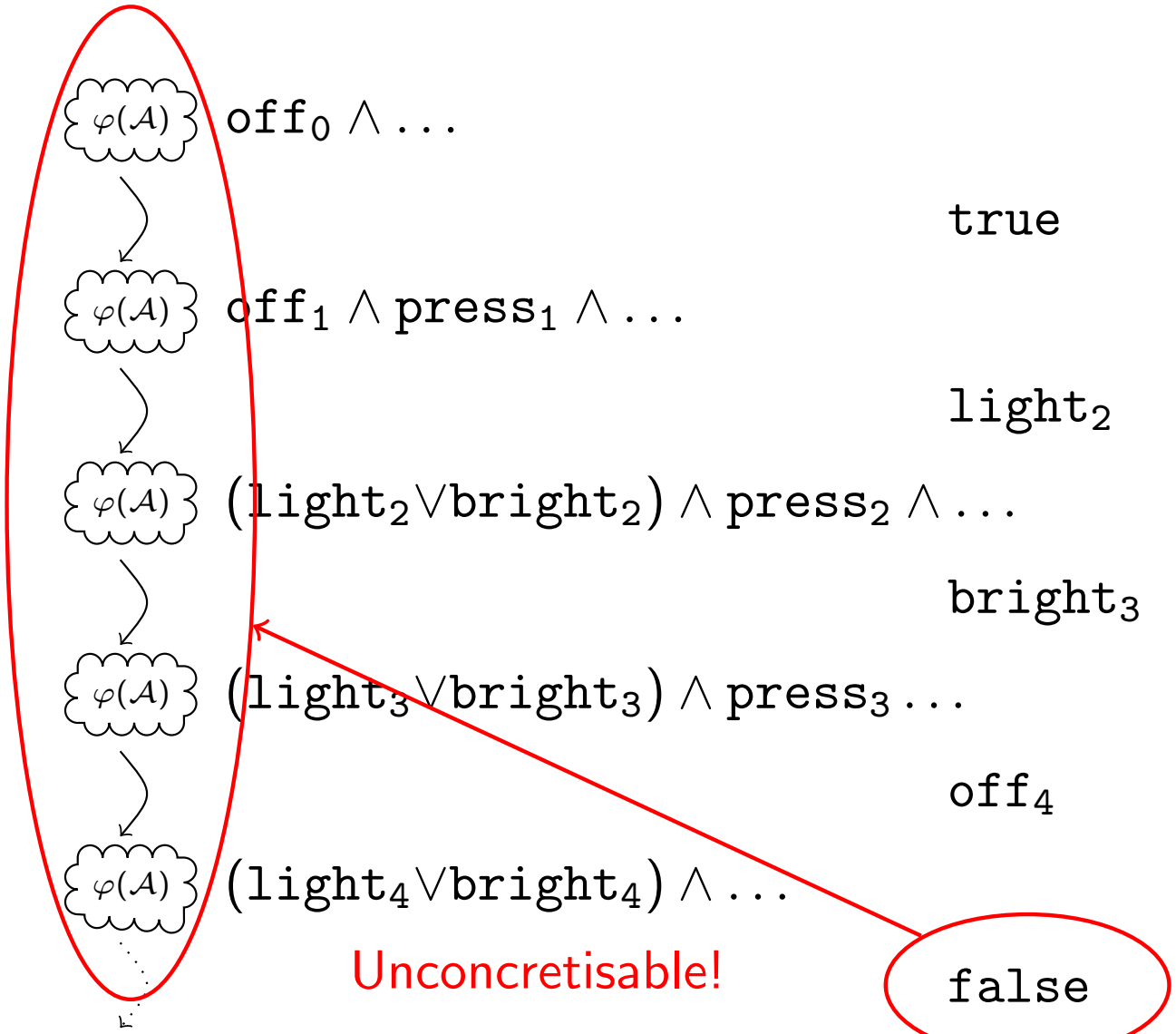
Refinement using Craig Interpolants



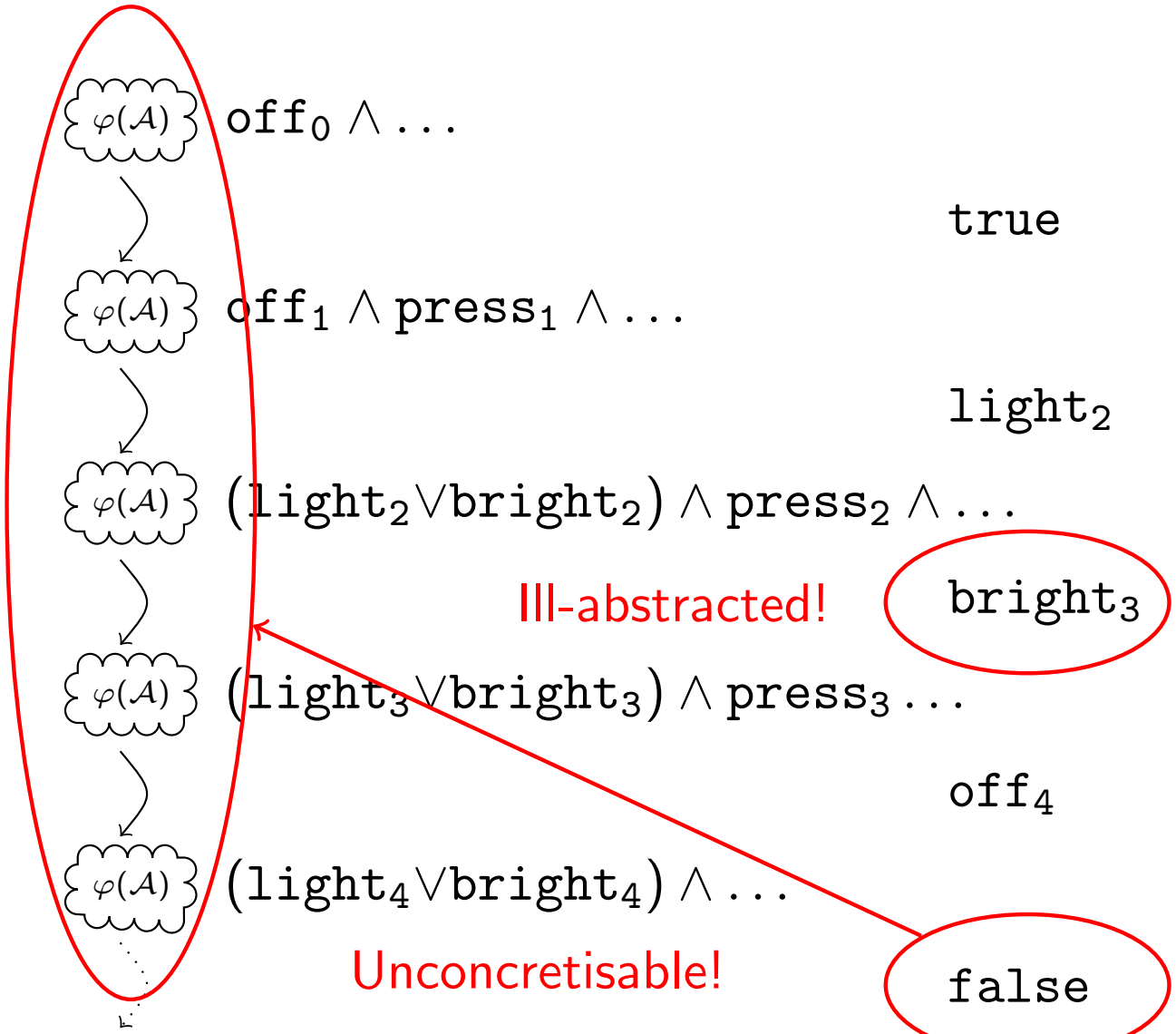
Refinement using Craig Interpolants



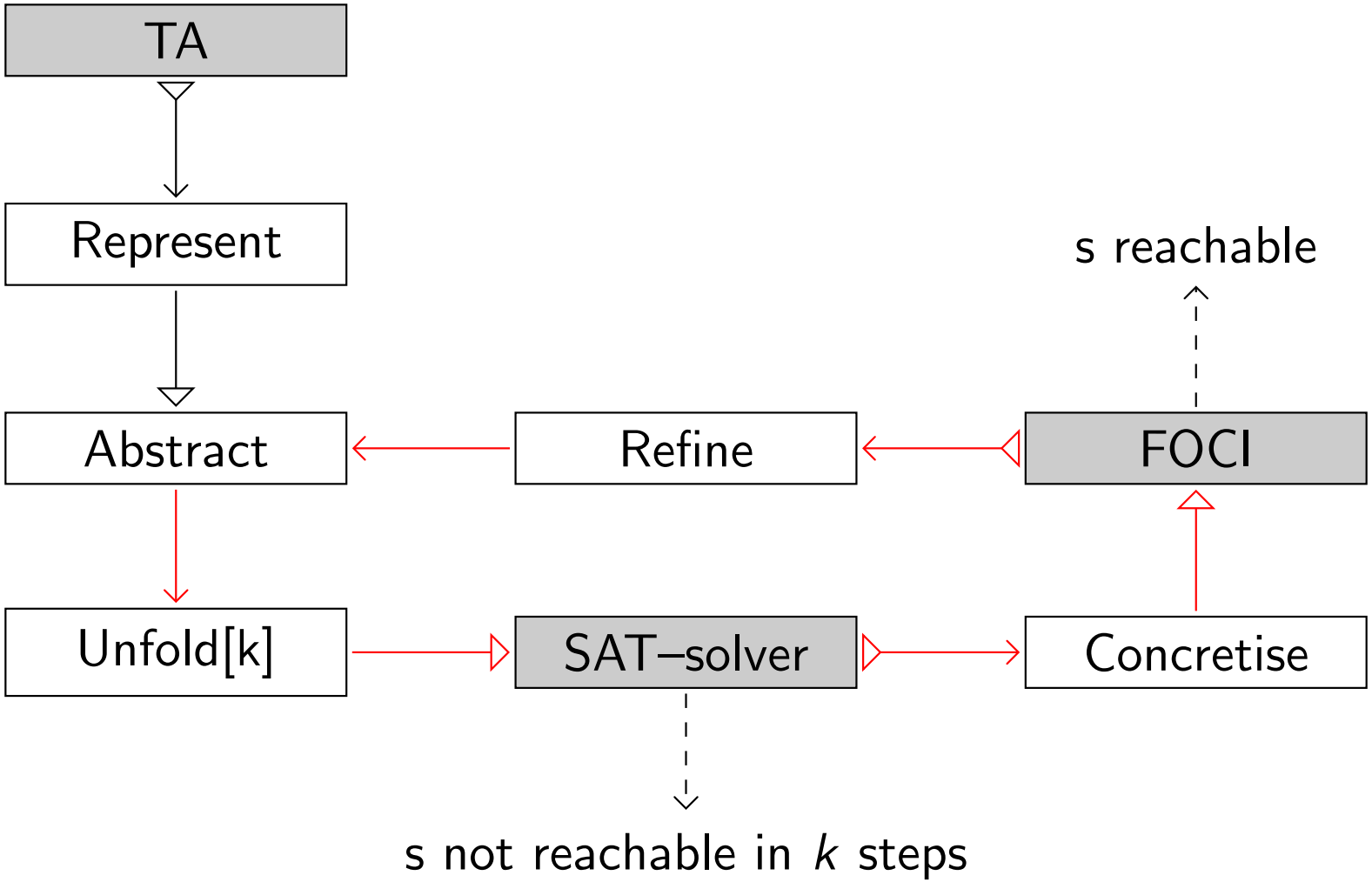
Refinement using Craig Interpolants



Refinement using Craig Interpolants



Abstraction Refinement



Preliminary Experimental Results

	Strategy	Complete Run	Hereof: SAT solver calls	Hereof: Last Iteration
TGC	$1/2k$	16 : 889	14 : 594	8 : 163
TGC	$1/3k$	15 : 651	13 : 081	5 : 445
TGC	$1/4k$	11 : 312	9 : 665	6 : 665
Optimised TGC	$1/2k$	16 : 631	14 : 334	7 : 866
Optimised TGC	$1/3k$	16 : 957	14 : 439	6 : 423
Optimised TGC	$1/4k$	11 : 063	9 : 451	6 : 279
Intelligent Light	$1/2k$	2 : 596	0 : 568	
Intelligent Light	$1/3k$	1 : 467	0 : 388	
Intelligent Light	$1/4k$	0 : 843	0 : 201	

Future Work

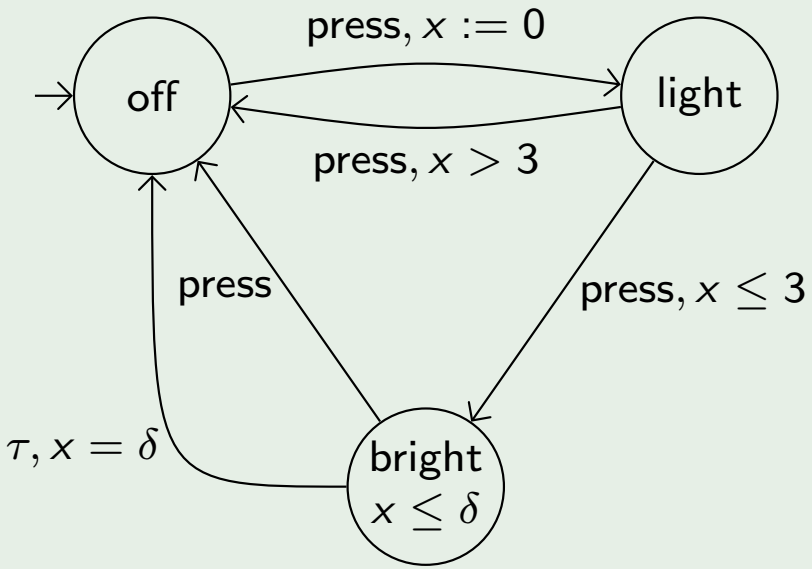
- Performance comparison to case studies
- Logarithmic encoding for states
- Performance improvements using pseudo-boolean constraints and isomorphy inference
- Better heuristics

Conclusion

- Abstraction refinement for real-time systems
- Bounded model checking with SAT and linear arithmetic representation
- Uniform abstraction in logic
- Linear parallel composition

Timed Automata: Semantics

Example (Intelligent Light Controller)



$$(\text{off}, [x = 0]) \xrightarrow{2} (\text{off}, [x = 2]) \xrightarrow{\text{press}}$$

$$(\text{light}, [x = 0]) \xrightarrow{2.5} (\text{light}, [x = 2.5]) \xrightarrow{\text{press}}$$

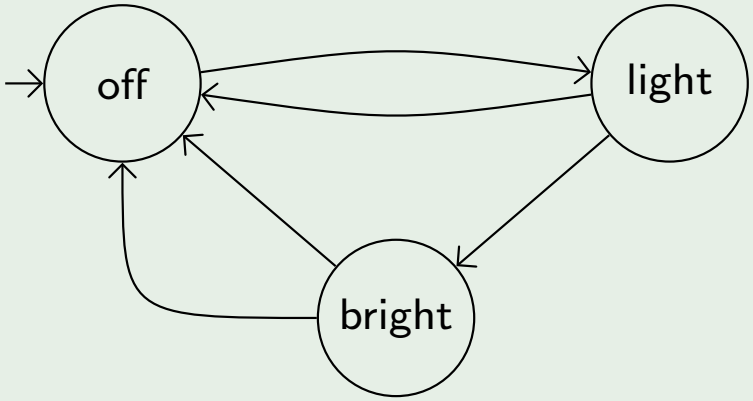
$$(\text{bright}, [x = 2.5]) \xrightarrow{\delta - 2.5}$$

$$(\text{bright}, [x = \delta]) \xrightarrow{\tau} (\text{off}, [x = \delta])$$

back

Unfolding: Example

Example ("Reduced" Intelligent Light Controller)



$$\left(\begin{array}{ll}(\mathtt{off}_0 \wedge \mathtt{light}_1) & \vee\\ (\mathtt{light}_0 \wedge \mathtt{off}_1) & \vee\\ (\mathtt{light}_0 \wedge \mathtt{bright}_1) & \vee\\ (\mathtt{bright}_0 \wedge \mathtt{off}_1) & \vee\\ (\mathtt{bright}_0 \wedge \mathtt{off}_1) & \vee\\ (\mathtt{off}_0 \wedge \mathtt{off}_1) & \vee\\ (\mathtt{light}_0 \wedge \mathtt{light}_1) & \vee\\ (\mathtt{bright}_0 \wedge \mathtt{bright}_1) & \end{array}\right) \wedge \left(\begin{array}{ll}(\mathtt{off}_1 \wedge \mathtt{light}_2) & \vee\\ (\mathtt{light}_1 \wedge \mathtt{off}_2) & \vee\\ (\mathtt{light}_1 \wedge \mathtt{bright}_2) & \vee\\ (\mathtt{bright}_1 \wedge \mathtt{off}_2) & \vee\\ (\mathtt{bright}_1 \wedge \mathtt{off}_2) & \vee\\ (\mathtt{off}_1 \wedge \mathtt{off}_2) & \vee\\ (\mathtt{light}_1 \wedge \mathtt{light}_2) & \vee\\ (\mathtt{bright}_1 \wedge \mathtt{bright}_2) & \end{array}\right) \wedge .$$

back

Abstraction by Merging Omission: Details

$$\alpha(L) = \begin{cases} L & \text{if } \mathit{conts}(L) \cap (\mathcal{AS} \cup \Sigma) = \emptyset \\ \gamma(L) & \text{if } \mathit{conts}(L) \cap \Sigma \neq \emptyset, L \text{ positive} \\ \texttt{true} & \text{otherwise} \end{cases}$$

back

Representation of Clock Values: z_t - x_t

Consider (unconditioned) delay transition in state s_0.

Representation of Clock Values: z_t - x_t

Consider (unconditioned) delay transition in state s_0.
With z:

$$s0_t \wedge s0_{t+1} \wedge \wedge(x_{t+1} = x_t) \wedge (y_{t+1} = y_t) \wedge (z_{t+1} > z_t)$$

Representation of Clock Values: z_t - x_t

Consider (unconditioned) delay transition in state s_0.
With z:

$$s0_t \wedge s0_{t+1} \wedge \wedge(x_{t+1} = x_t) \wedge (y_{t+1} = y_t) \wedge (z_{t+1} > z_t)$$

Without z:

$$s0_t \wedge s0_{t+1} \wedge \wedge(x_{t+1} - x_t = a_t) \wedge (y_{t+1} - y_t = a_t) \wedge (a_t > 0)$$

back

Representation of Clock Values: z_t - x_t

Nine steps, consider only clocks. Average of 15 executions:

Representation of Clock Values: z_t - x_t

Nine steps, consider only clocks. Average of 15 executions:

# clocks	with z	without z	factor
5	0.040s	1.30s	32
6	0.045s	2.04s	45
7	0.050s	3.00s	60
10	0.055s	8.50s	150

back