Model Checking

Lecture 2

Three important decisions when choosing system properties:

1. automata vs. logic
2. branching vs. linear time
3. safety vs. liveness

The three decisions are orthogonal, and they lead to substantially different model-checking problems.

If only universal properties are of interest,

why not omit the path quantifiers?

LTL (Linear Temporal Logic)

- safety & liveness
- linear time

[Pnueli 1977; Lichtenstein & Pnueli 1982]

LTL Syntax

$$\varphi \;::=\; a \mid \varphi \wedge \varphi \mid \neg\, \varphi \mid \bigcirc\, \varphi \mid \varphi\, \mathsf{U}\, \varphi$$

LTL Model

infinite trace $t = t_0\ t_1\ t_2 \ldots$
(sequence of observations)

Language of deadlock-free state-transition graph K at state q :

L(K,q) = set of infinite traces of K starting at q

$(K,q) \models^{\forall} \varphi$ iff for all $t \in L(K,q)$, $t \models \varphi$

$(K,q) \models^{\exists} \varphi$ iff exists $t \in L(K,q)$, $t \models \varphi$

LTL Semantics

t	$\models a$	iff	$a \in t_0$
t	$\models \varphi \wedge \psi$	iff	$t \models \varphi$ and $t \models \psi$
t	$\models \neg\varphi$	iff	not $t \models \varphi$
t	$\models \bigcirc\, \varphi$	iff	$t_1\, t_2 \ldots \models \varphi$
t	$\models \varphi \,U\, \psi$	iff	exists $n \geq 0$ s.t. 1. for all $0 \leq i < n$, $t_i\, t_{i+1} \ldots \models \varphi$ 2. $t_n\, t_{n+1} \ldots \models \psi$

Defined modalities

$\bigcirc$	X	next
U	U	until
$\Diamond \varphi = \text{true } U \varphi$	F	eventually
$\Box \varphi = \neg \Diamond \neg\varphi$	G	always
$\varphi W \psi = (\varphi U \psi) \vee \Box\varphi$	W	waiting-for (weak-until)

Summary of modalities

STL	$\exists\bigcirc$ $\forall\bigcirc$ $\exists\Diamond$ $\forall\Box$ $\exists$U $\forall$W
CTL	all of the above and $\exists\Box$ $\forall\Diamond$ $\exists$W $\forall$U
LTL	$\bigcirc$ $\Diamond$ $\Box$ U W

Important properties

Invariance $\Box\, a$ safety

$\Box\, \neg\, (pc1=in \wedge pc2=in)$

Sequencing $a\ W\ b\ W\ c\ W\ d$ safety

$\Box\, (pc1=req \Rightarrow$

$(pc2\neq in)\ W\ (pc2=in)\ W\ (pc2\neq in)\ W\ (pc1=in))$

Response $\Box\, (a \Rightarrow \Diamond\, b)$ liveness

$\Box\, (pc1=req \Rightarrow \Diamond\, (pc1=in))$

Composed modalities

$\Box\Diamond\, a$	infinitely often a
$\Diamond\Box\, a$	almost always a

Where did fairness go ?

Unlike in CTL, fairness can be expressed in LTL !

So there is no need for fairness in the model.

Weak (Buchi) fairness :

$\neg \Diamond \Box$ (enabled $\wedge \neg$ taken) =

$\Box \Diamond$ (enabled $\Rightarrow$ taken)

Strong (Streett) fairness :

($\Box \Diamond$ enabled) $\Rightarrow$ ($\Box \Diamond$ taken)

Starvation freedom, corrected

$$\Box\Diamond\,(pc2{=}in \Rightarrow \bigcirc\,(pc2{=}out)) \Rightarrow$$
$$\Box\,(pc1{=}req \Rightarrow \Diamond\,(pc1{=}in))$$

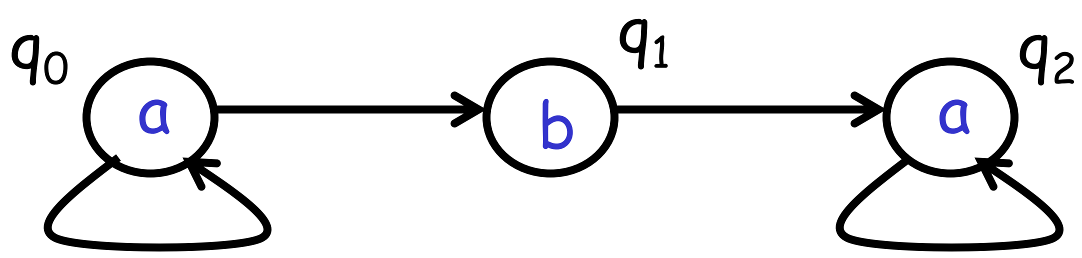
CTL cannot express fairness

Must happen on a finite prefix

$\forall\Diamond\Box\, a \neq \forall\Diamond\, \forall\Box\, a$

$\exists\Box\Diamond\, b \neq \exists\Box\, \exists\Diamond\, b$

Must be an infinite run



LTL cannot express branching

Possibility $\forall\Box\,(a \;\Rightarrow \exists\Diamond\, b)$

So, LTL and CTL are incomparable.

(There are branching logics that can express fairness, e.g., CTL^* = CTL + LTL, but they lose the computational attractiveness of CTL.)

System property: 2x2x2 choices

-safety (finite runs) vs. liveness (infinite runs)

-linear time (traces) vs. branching time (trees)

-logic (declarative) vs. automata (operational)

Specification Automata

Syntax, given a set A of atomic observations:

S	finite set of states
$S_0 \subseteq S$	set of initial states
$\to \subseteq S \times S$	transition relation
ϕ: $S \to PL(A)$	where the formulas of PL are
	$\varphi \quad ::= \quad a \mid \varphi \wedge \varphi \mid \neg \varphi$
	for $a \in A$

Language L(M) of specification automaton

$M = (S, S_0, \rightarrow, \phi)$:

infinite trace $t_0, t_1, \ldots \in L(M)$

iff

there exists a infinite run $s_0 \rightarrow s_1 \rightarrow \ldots$ of M

such that

for all $0 \leq i$, $t_i \models \phi(s_i)$

Linear semantics of specification automata:

language containment

$$(K,q) \models^L M \qquad \text{iff} \qquad L(K,q) \subseteq L(M)$$

state-transition graph

state of K

specification automaton

infinite traces

$L^{fin}(K,q)$ = set of finite traces of K starting at q

$L^{fin}(M)$ defined as follows:

finite trace $t_0, \ldots, t_n \in L^{fin}(M)$

iff

there exists a finite run $s_0 \to s_1 \to \ldots \to s_n$ of M

such that

for all $0 \leq i \leq n$, $t_i \models \phi(s_i)$

$$(K,q) \models^{L} M$$

iff

$$L(K,q) \subseteq L(M)$$

iff

$$L^{fin}(K,q) \subseteq L^{fin}(M)$$

Proof requires three facts:

- K is deadlock-free
    - every state in K has a transition from it
- M is finite-branching:
    - number of transitions from a state in M is bounded
- Konig's lemma
    - A finite-branching infinite tree has an infinite path

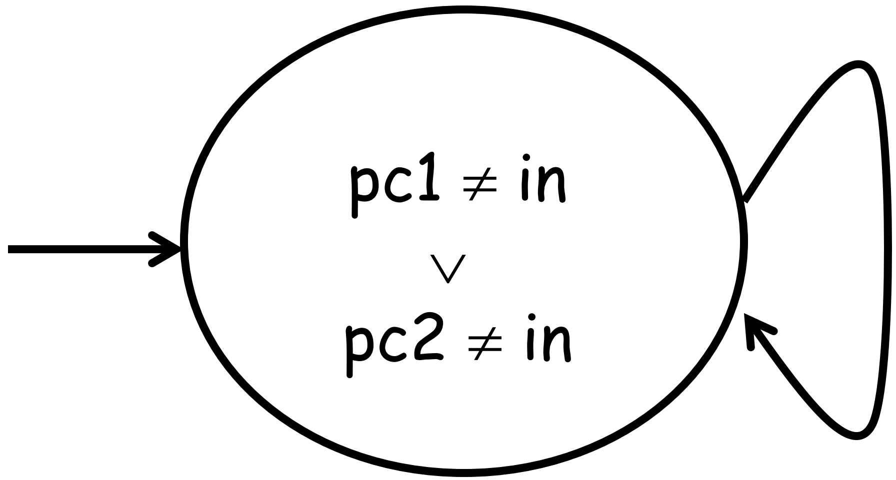
$(K,q) \models^{L} M$

iff

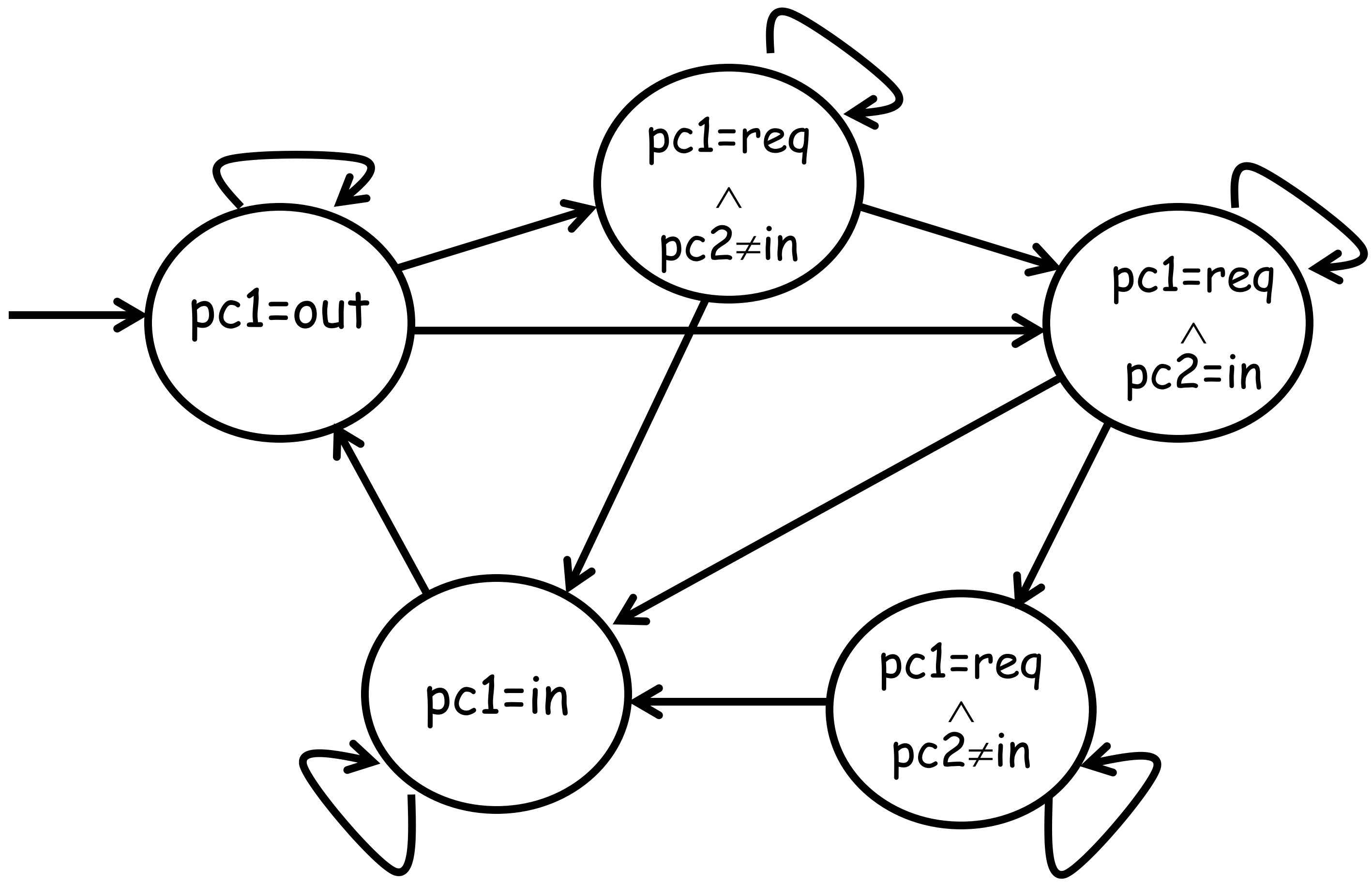
$L^{fin}(K,q) \subseteq L^{fin}(M)$

To verify $(K,q) \models^{L} M$, check finitary trace-containment

Invariance specification automaton

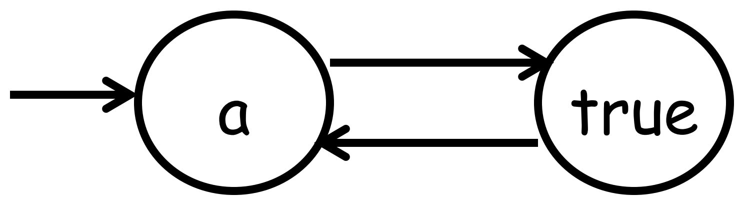


One-bounded overtaking specification automaton



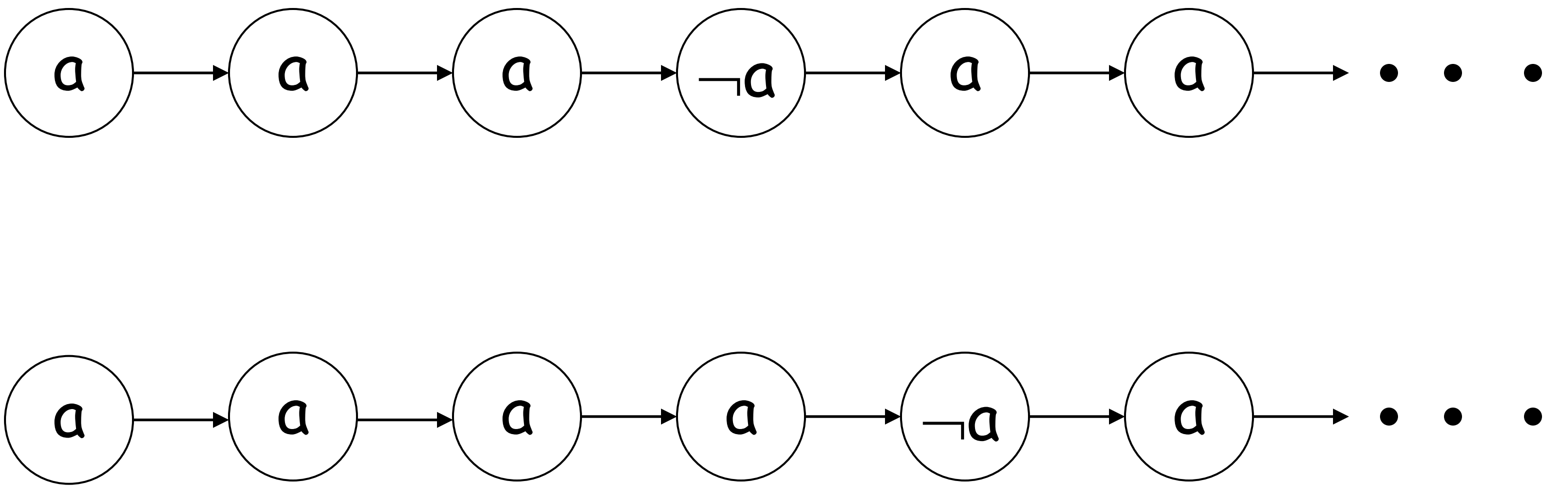
Automata are more expressive than logic, because temporal logic cannot count :

Let $A = \{ a \}$

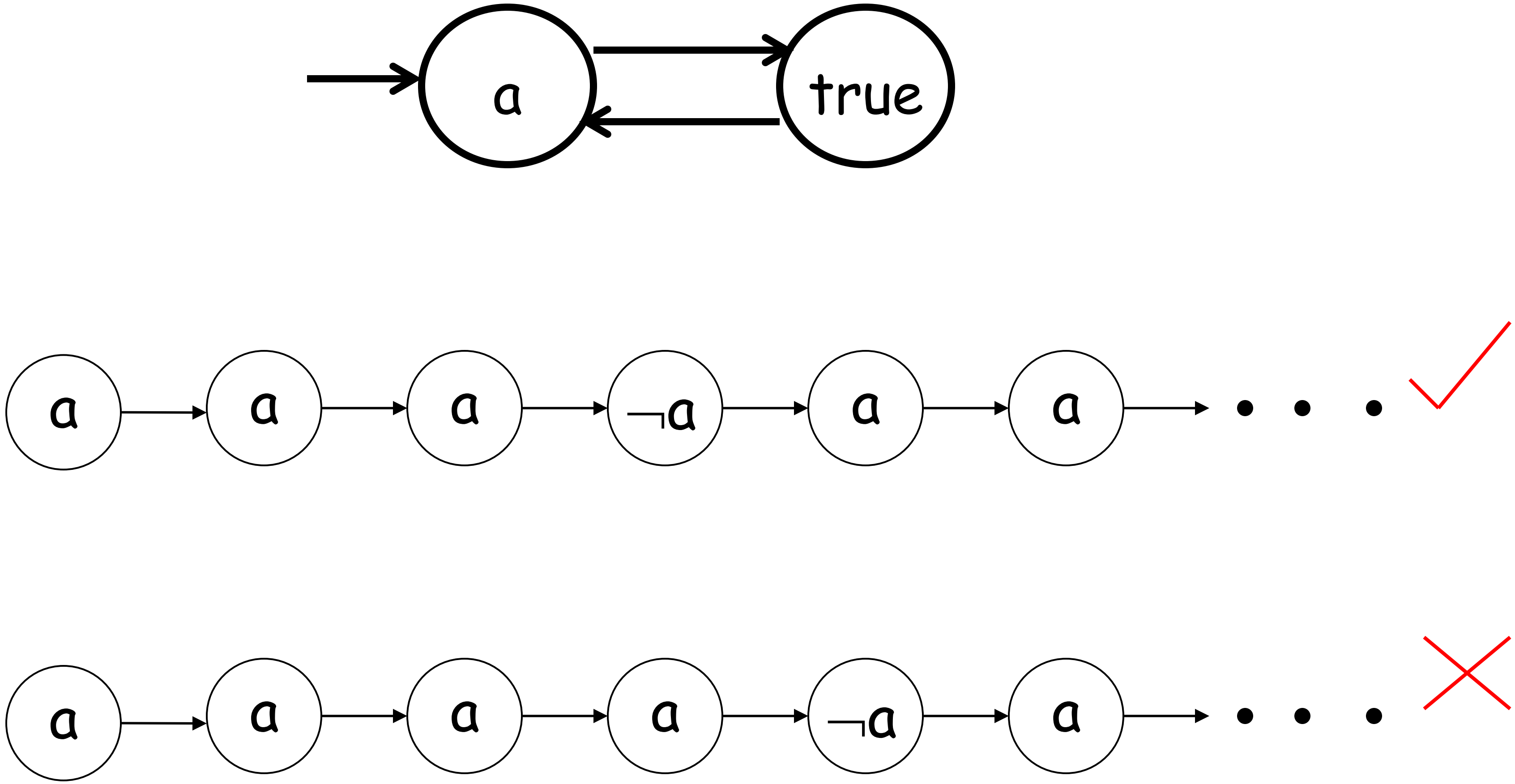


This cannot be expressed in LTL.

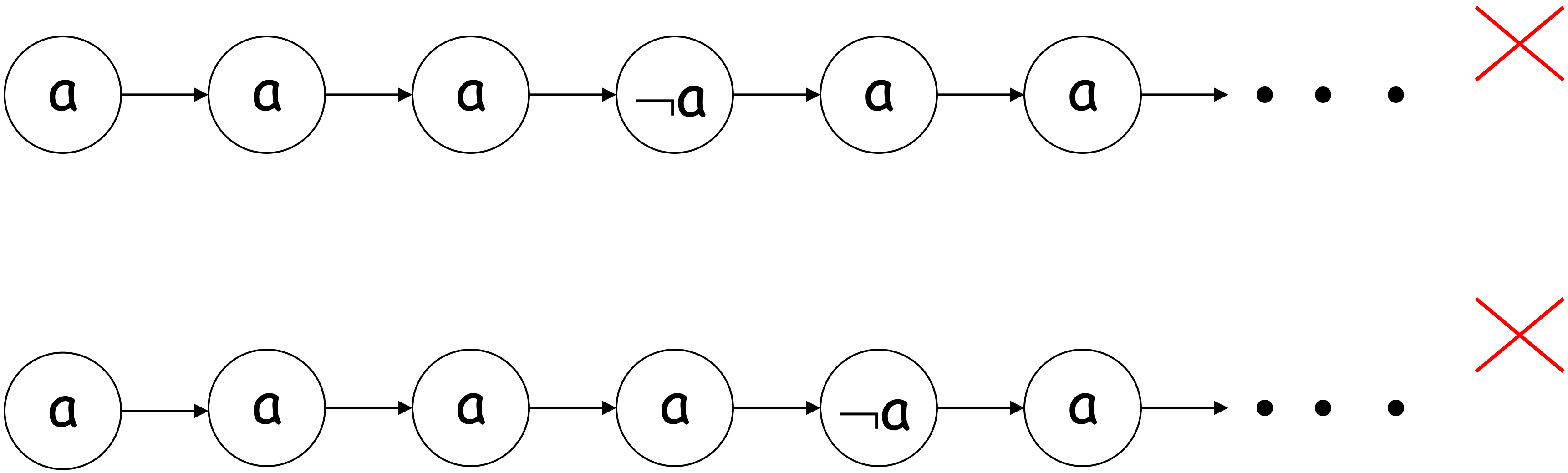
(How about $a \wedge \Box (a \Rightarrow \bigcirc\bigcirc a)$?)



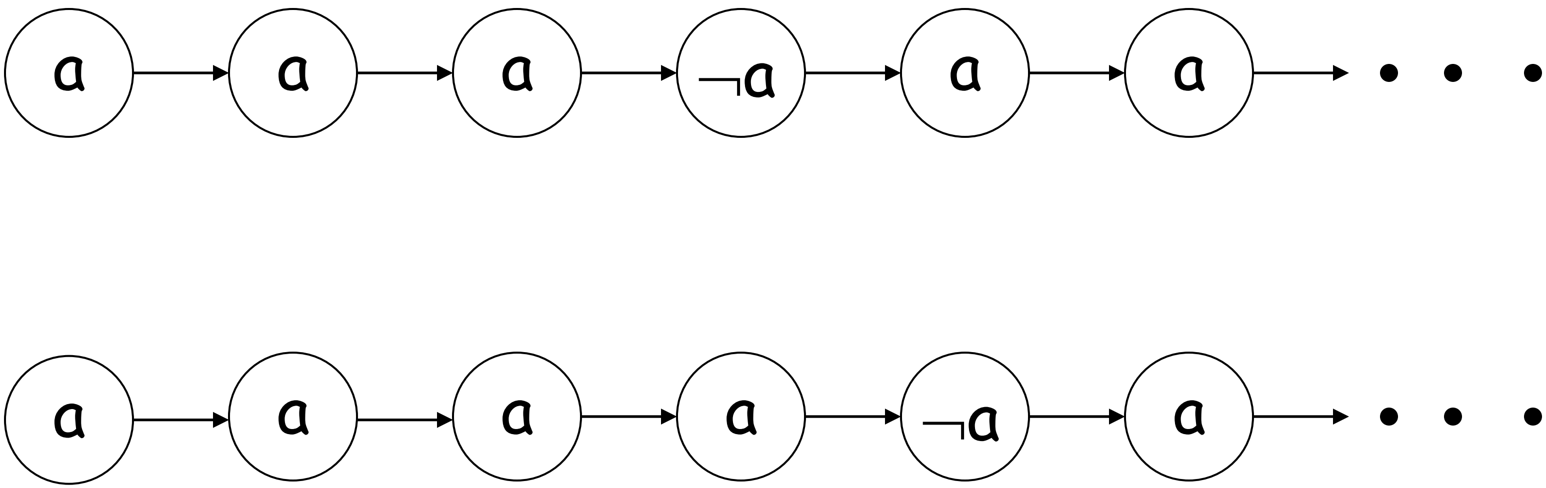
a
a
a
¬a
a
a
a
a
a
a
¬a
a


a
true
a a a ¬a a a
a a a a ¬a a

$$a \wedge \Box\,(a \Rightarrow \bigcirc\bigcirc\, a)$$



In fact, no LTL formula with at most two occurrences of ○ can distinguish between the two traces.



Proof?

Checking language containment between finite automata is PSPACE-complete !

$$L(K,q) \subseteq L(M)$$

iff

$$L(K,q) \cap \text{complement}(\ L(M)\ ) = \emptyset$$

involves determinization
(subset construction)

In practice:

1. use monitor automata
2. use simulation as a sufficient condition

Monitor Automata

Syntax:

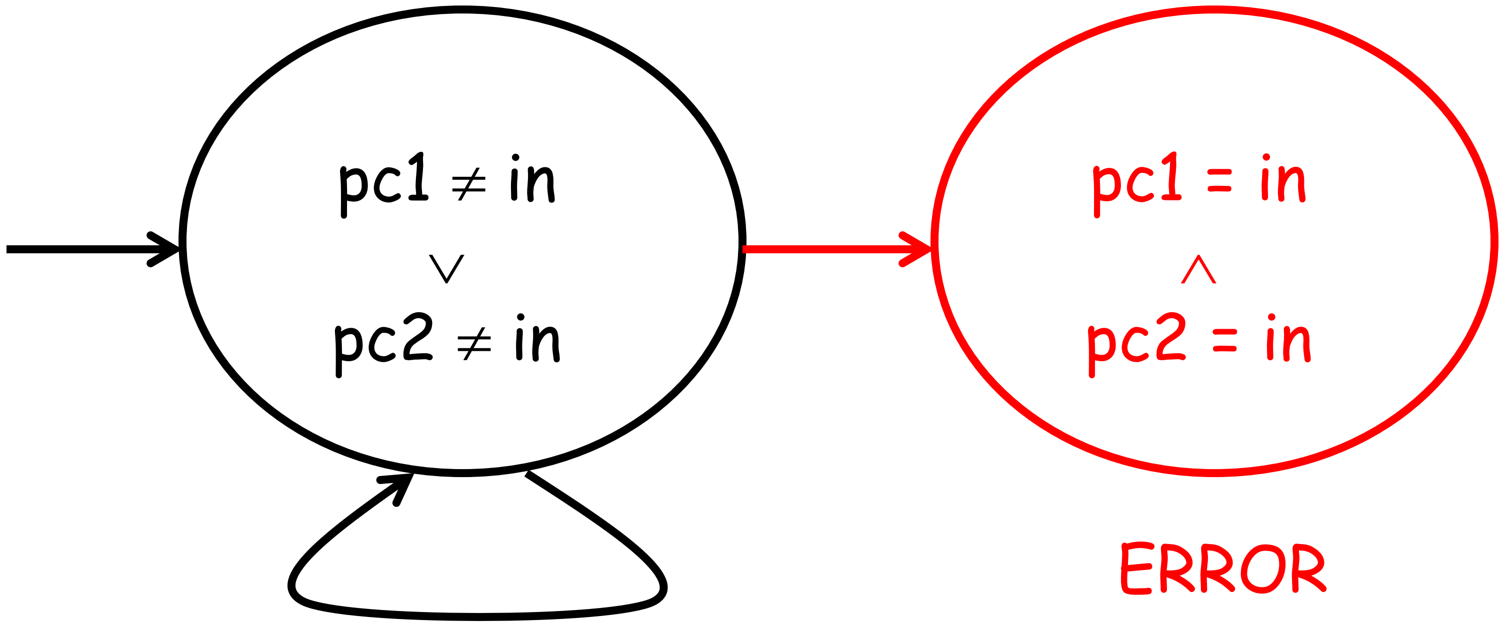
same as specification automata,
except also set $E \subseteq S$ of error states

Semantics:

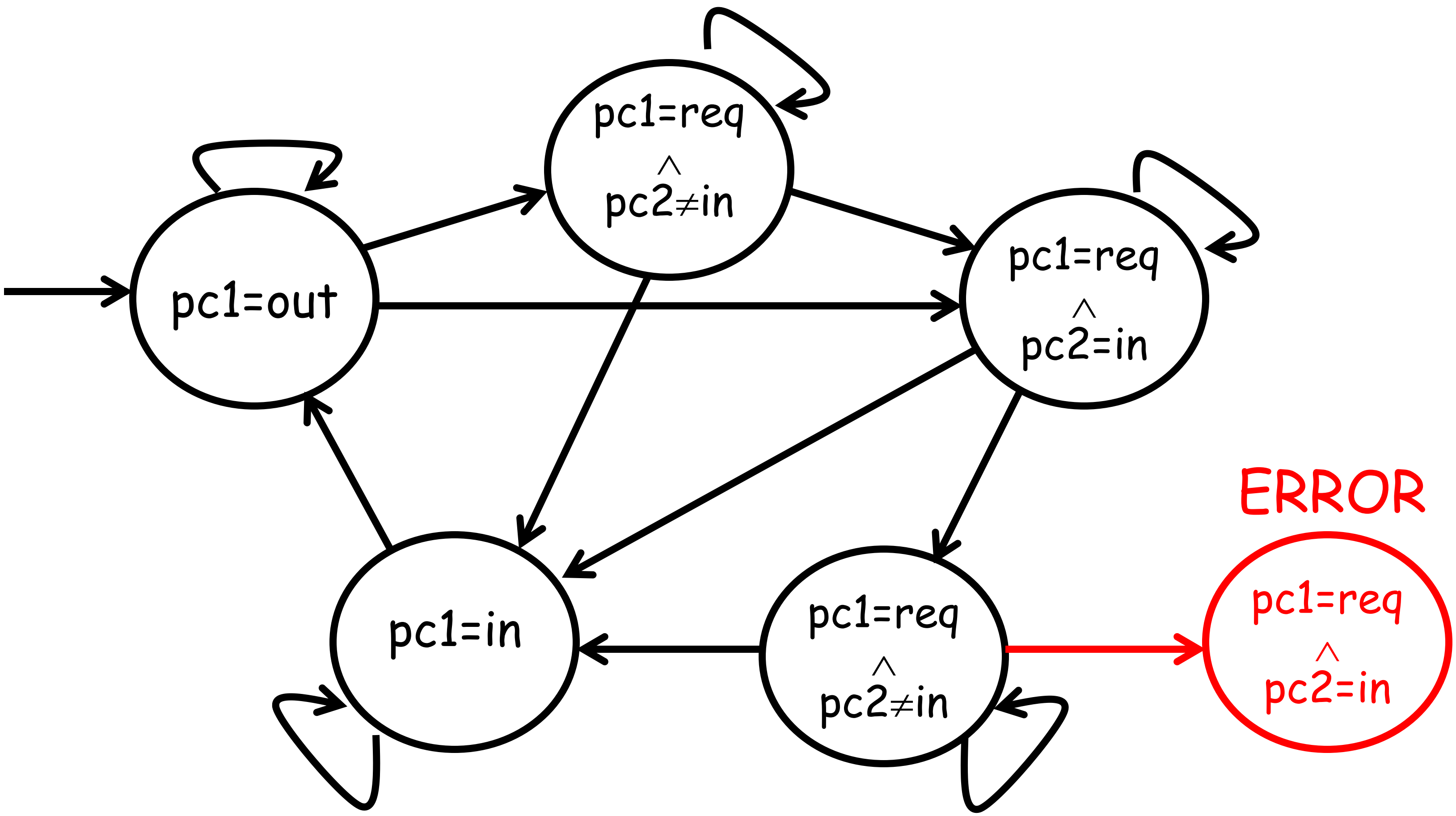
define $L(M)$ s.t. runs must end in error states

$(K,q) \models^C M$ iff $L(K,q) \cap L(M) = \varnothing$

Invariance monitor automaton



One-bounded overtaking monitor automaton



Specification automaton

$M \rightleftarrows$

-describe correct traces

-check language containment (exponential)

Monitor automaton

complement(M)

-describe error traces

-check emptiness (linear): reachability of error states

"All safety verification is reachability checking."

In practice:

1. use monitor automata
2. use simulation as sufficient condition

Branching semantics of specification automata:

simulation

$(K,q) \models^{B} M$

iff

there exists a simulation relation $R \subseteq Q \times S$

s.t. $(q,s) \in R$ for some initial state s of M

states of K

states of M

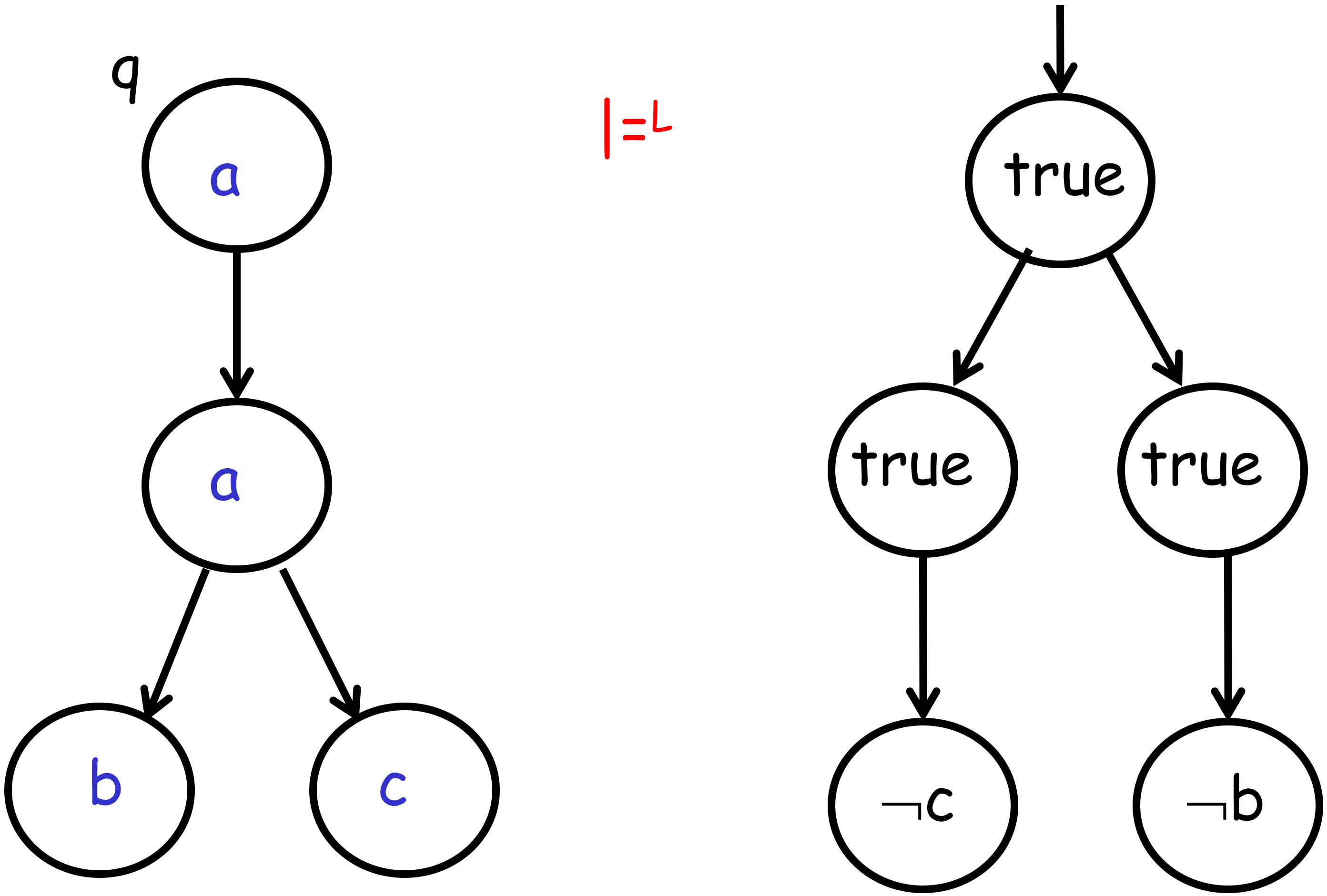
$R \subseteq Q \times S$ is a simulation relation

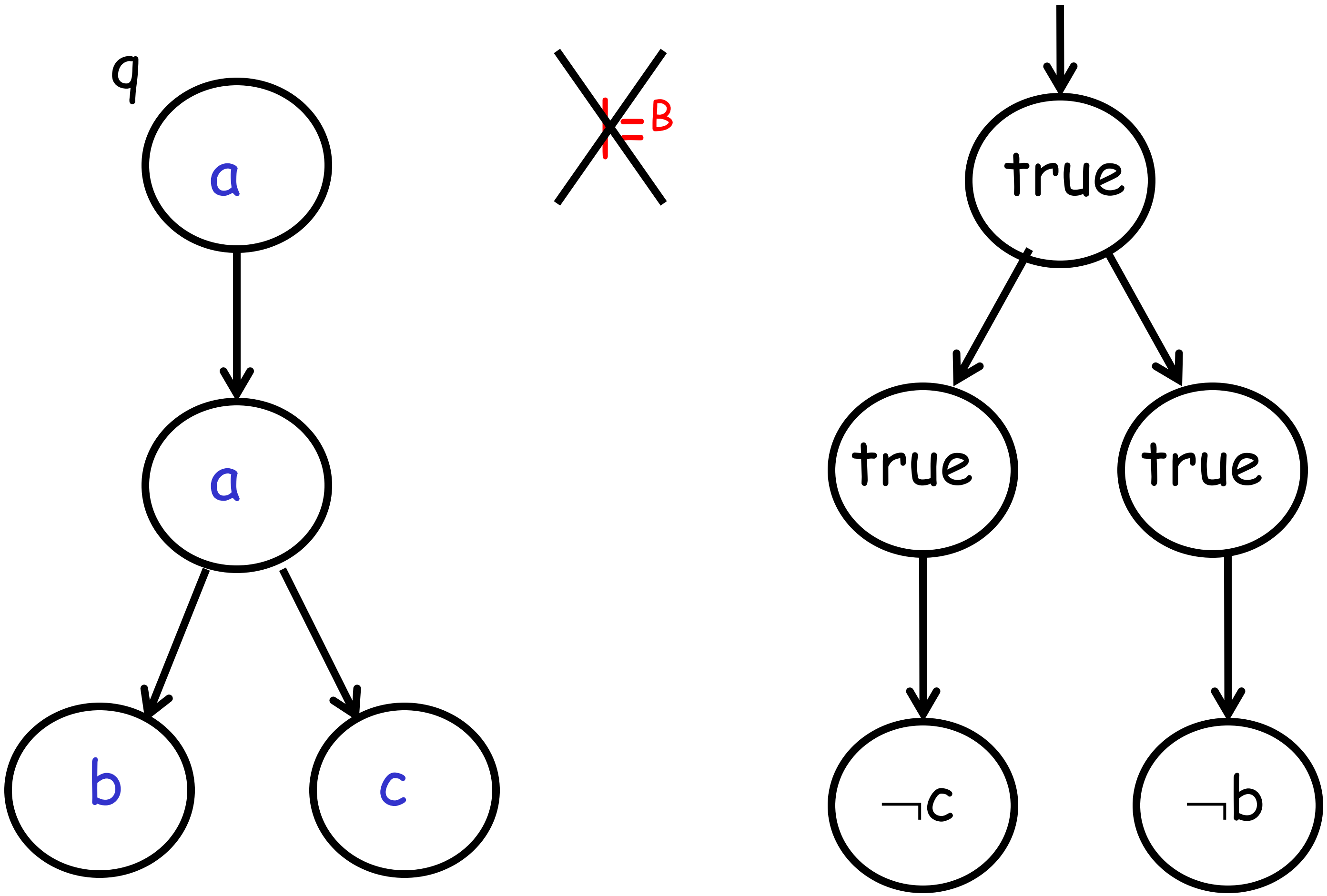
iff

$(q,s) \in R$ implies

1. $[q] \models \phi(s)$
2. for all q' s.t. $q \rightarrow q'$,
   exists s' s.t. $s \rightarrow s'$ and $(q',s') \in R$.

[Milner 1974]


q
a
a
b
c
|=L
true
true
true
¬c
¬b


q
a
a
b
c
⊭B
true
true
true
¬c
¬b

involves only traces (hence linear !)

$(K,q) \models^{L} M$ — M language contains (K,q) : exponential check

$\Uparrow \qquad \cancel{\Downarrow}$

$(K,q) \models^{B} M$ — M simulates (K,q) : quadratic check

involves states (hence branching !)

In practice, simulation is usually the “right” notion.

(If there is language containment, but not simulation, this is usually accidental, not by design.)

Branching semantics of specification automata, alternative definition:

trace-tree containment

$(K,q) \models^{B} M \quad \text{iff} \quad T(K,q) \subseteq T(M)$

finite trace trees

Omega Automata

- -safety & liveness (infinite runs !)
- -specification vs. monitor automata
- -linear (language containment) vs. branching (simulation) semantics

We discuss only the linear specification case.

Specification Omega Automata

Syntax as for finite automata,
in addition an acceptance condition:

Buchi: $BA \subseteq S$

Language L(M) of specification omega-automaton

$M = (S, S_0, \rightarrow, \phi, BA)$:

infinite trace $t_0, t_1, \ldots \in L(M)$

iff

there exists an infinite run $s_0 \rightarrow s_1 \rightarrow \ldots$ of M

such that

1. $s_0 \rightarrow s_1 \rightarrow \ldots$ satisfies BA
2. for all $i \geq 0$, $t_i \models \phi(s_i)$

Let $Inf(s) = \{ p \mid p = s_i \text{ for infinitely many } i \}$.

The infinite run s satisfies the acceptance condition BA

iff

Buchi: $Inf(s) \cap BA \neq \emptyset$

Linear semantics of specification omega automata:

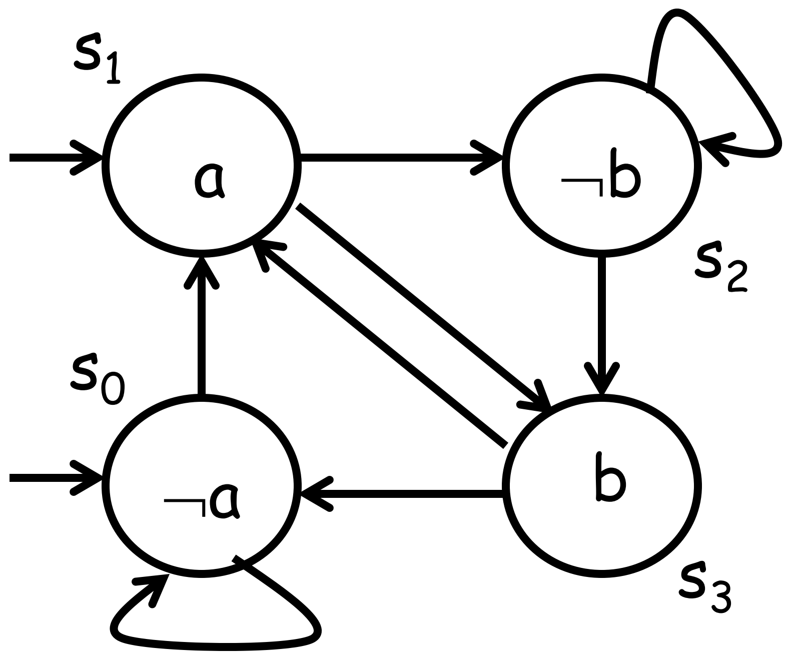
omega-language containment

$(K,q) \models^{L} M \quad \text{iff} \quad L(K,q) \subseteq L(M)$

infinite traces

Response specification automaton :

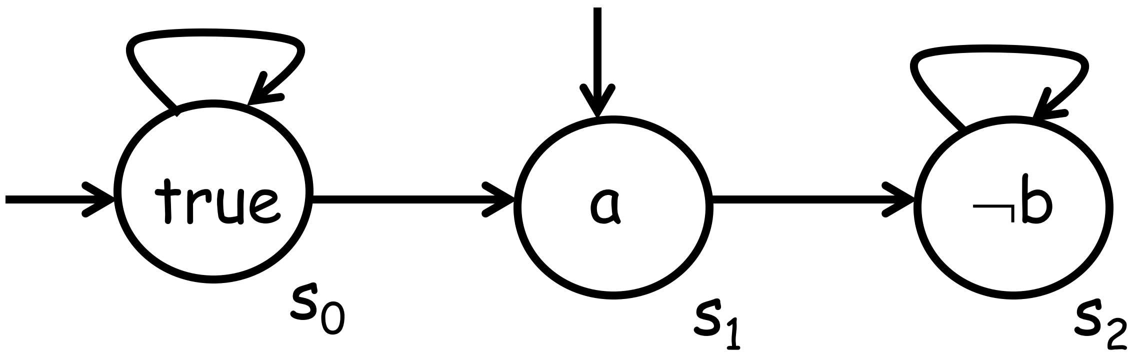
$\Box\,(a \Rightarrow \Diamond b)$ assuming $(a \wedge b) = \text{false}$



Buchi condition $\{ s_0, s_3 \}$

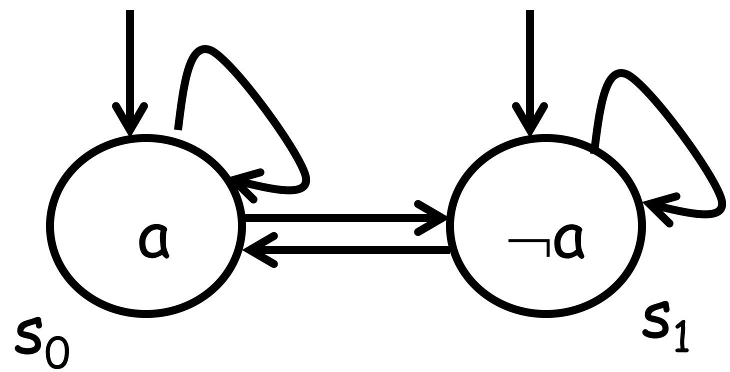
Response monitor automaton :

$\Box(a \Rightarrow \Diamond b)$ assuming $(a \wedge b) = \text{false}$



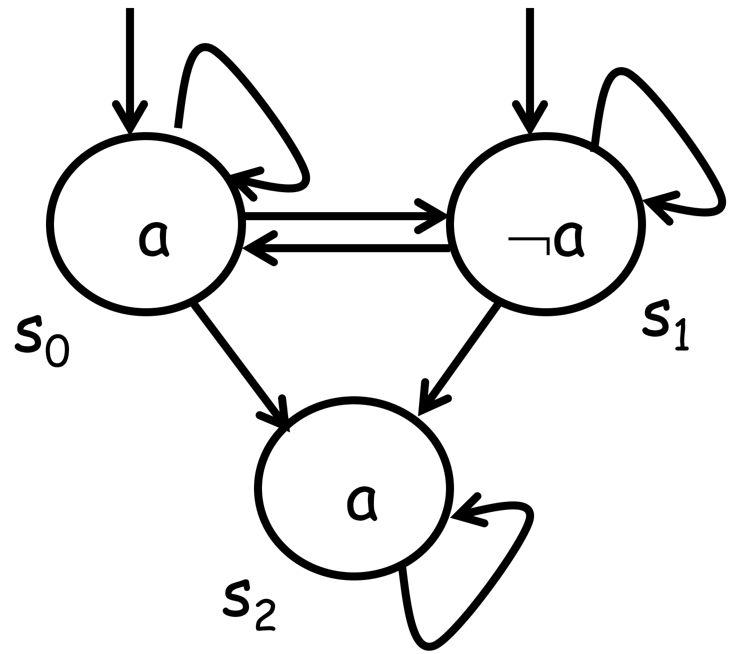
Buchi condition $\{ s_2 \}$

$\square\Diamond\, a$



Buchi condition $\{ s_0 \}$

$\Diamond\Box\, a$



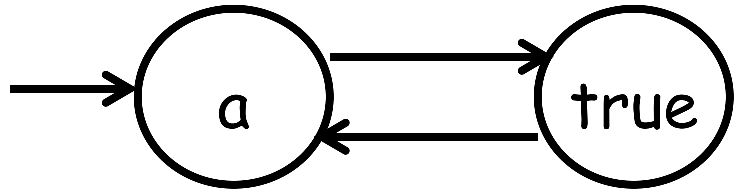
Buchi condition $\{ s_2 \}$

Omega automata are strictly more expressive than LTL.

Omega-automata: omega-regular languages

$\cup$

LTL: counter-free omega-regular languages



$$(\forall p)\,(\, p \wedge \bigcirc\neg p \wedge \Box(p \Leftrightarrow \bigcirc\bigcirc p) \;\Rightarrow\; \Box(p \Rightarrow a))$$

$$(\forall p)\,(\, p(0) \wedge \neg p(1) \wedge (\forall t)\,(p(t) \Leftrightarrow p(t+2)) \;\Rightarrow\; (\forall t)\,(p(t) \Rightarrow a(t)))$$

$$(a;\ true)^{\omega}$$