Lecture 18: CTL Model Checking (cont.)
Intro to timed automata

Huan Zhang

huan@huan-zhang.com

Slides adapted from Prof. Sayan Mitra's slides in Fall 2021

Logistics

HW 1 graded - contact TA Sanil Arun Chawla <schawla7@illinois.edu> **by Saturday (3/24)** for regrade request

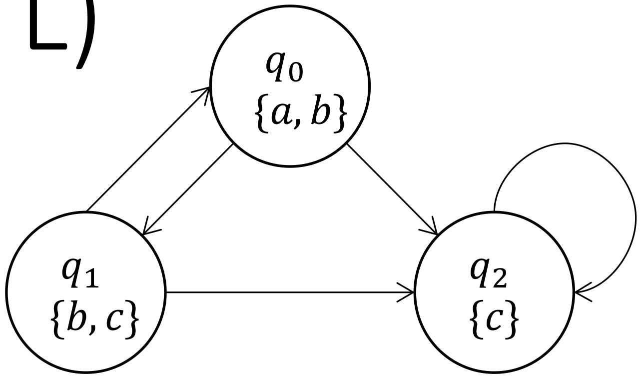
Midterm project presentation: **3/26** and **3/28**. (Next week!)

- **5-min** presentations for each team. (5% of final grade) + 2 min Q/A & Feedback
- Submit your slides on Canvas (due on **3/25)**. We will compile all slides into a single file for fast switching.
- Presentation includes **problem setting**, proposed **methodology**, and **initial results**.

In addition: each person should give feedback for 3 projects that interest you most on each day. (total 6 feedbacks; count towards the **5% class participation** grades), due **3/29**

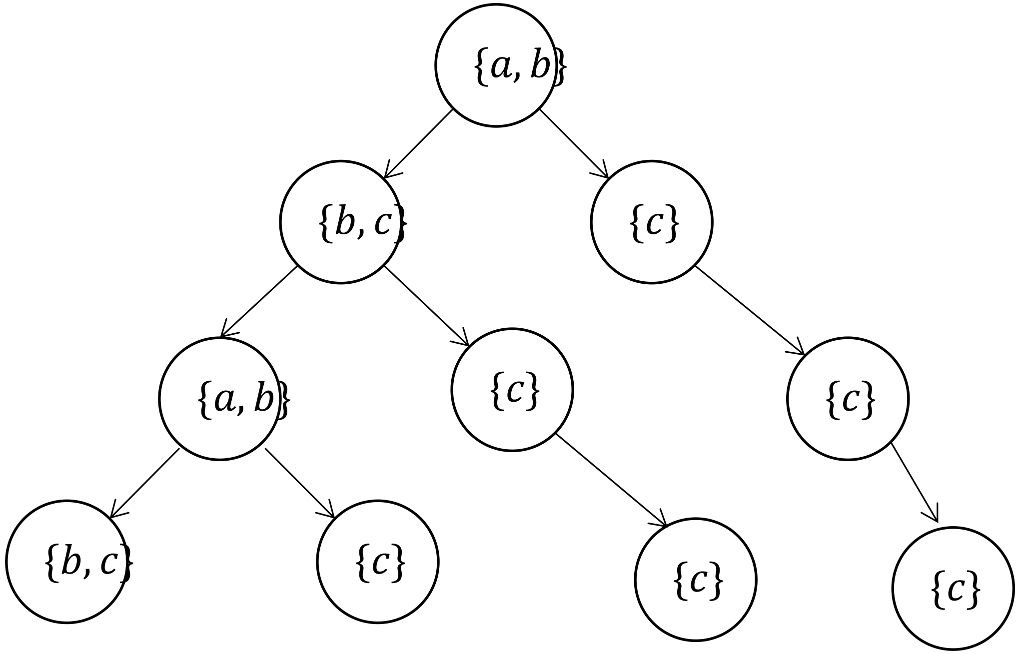
Feedback will be submitted to Canvas and also shared to the class (use the **template** on Canvas)

Review Computation tree logic (CTL)



Unfolding the automaton

We get a tree, representing all possible compuations



A CTL formula allows us to specify subsets of paths in this tree

Review: CTL quantifiers

Path quantifiers

E: Exists some path
A: All paths

Temporal operators

X: Next state
U: Until (“p U q” means “p holds until q holds”)
F: Eventually (some time in future)
G: Globally (always)

Visualizing CTL semantics

Path quantifiers

E: Exists some path

A: All paths

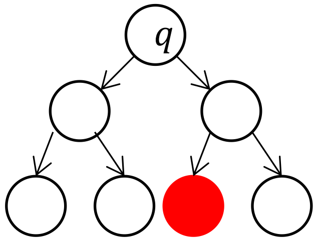
Temporal operators

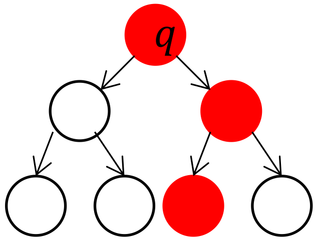
X: Next state

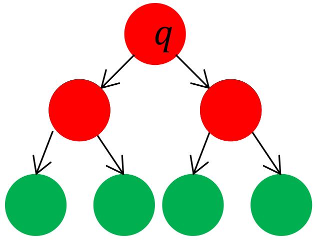
U: Until

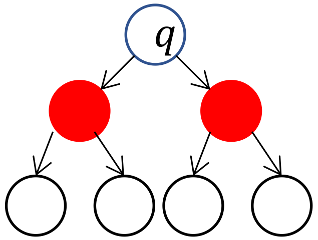
F: Eventually

G: Globally

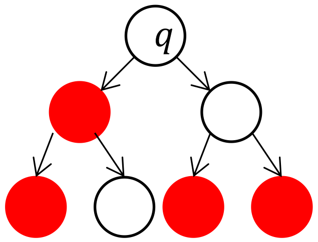


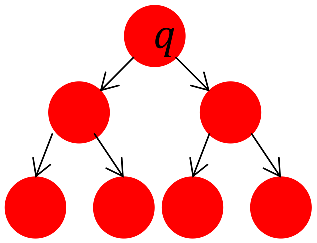


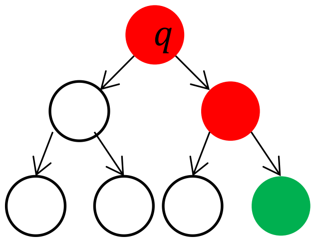


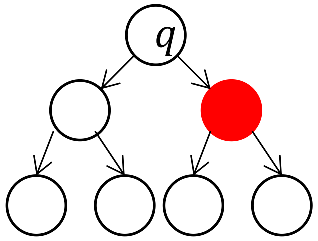


$q \vDash ?$

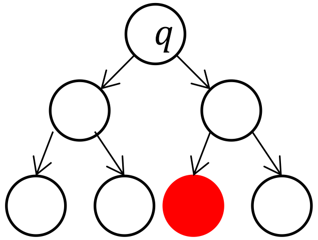




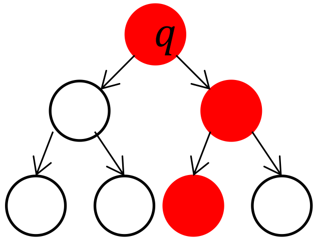




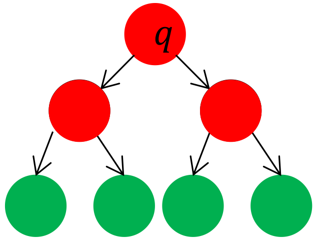
Visualizing CTL semantics



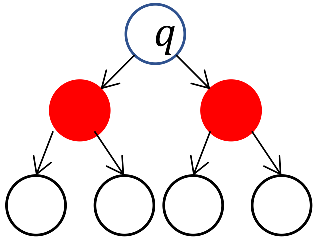
$q \vDash EF\ red$



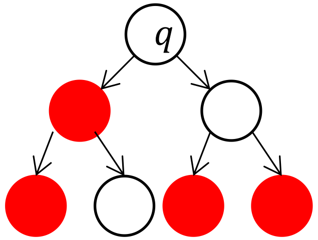
$q \vDash EG\ red$



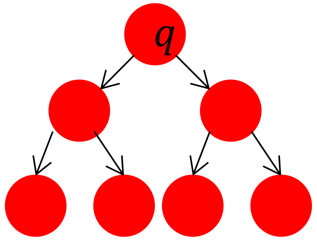
$q \vDash A\ [red\ U\ green]$



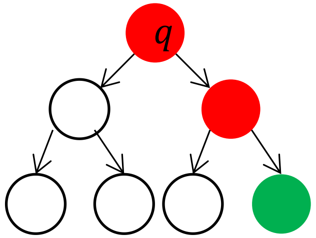
$q \vDash AX\ red$



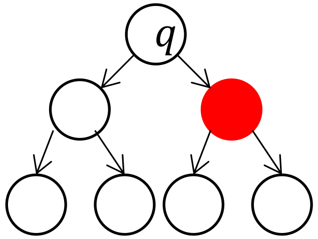
$q \vDash AF\ red$



$q \vDash AG\ red$



$q \vDash E\ [red\ U\ green]$



$q \vDash EX\ red$

Universal CTL operators

$\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{G}$ can be used to derive other operators

$true\ U\ f \equiv F\ f$

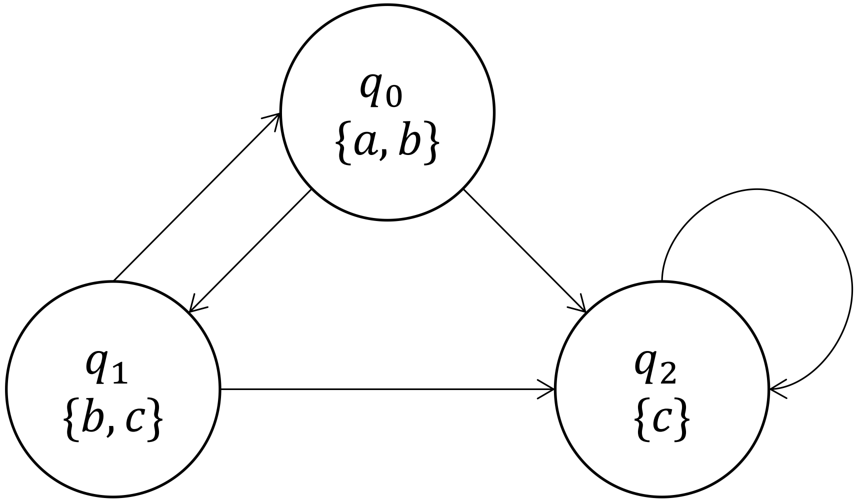
$Gf \equiv \neg F(\neg f)$

All combinations can be expressed using $\boldsymbol{EX}, \boldsymbol{EU}, \boldsymbol{EG}$

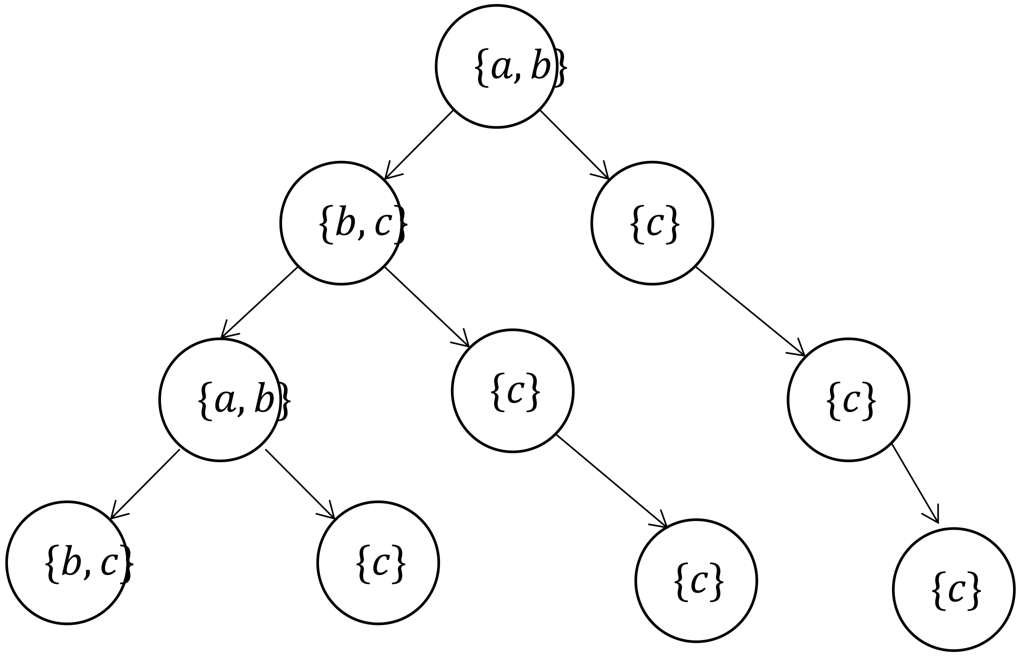
AXf	AGf	AFf	$A[f_1 U f_2]$
$\neg EX(\neg f)$	$\neg EF(\neg f)$	$\neg EG(\neg f)$	$\neg(E[(\neg f_1) U \neg(f_1 \vee f_2)] \vee EG(\neg f_2))$
EX	EG	EF	EU
EX	EG	$E(true\ U\ f)$	EU

Algorithm for deciding $\mathcal{A} \vDash f$

Since all CTL operators can be expressed by EX, EU, EG, we just need to figure out how to check these operators



$\mathcal{A}$

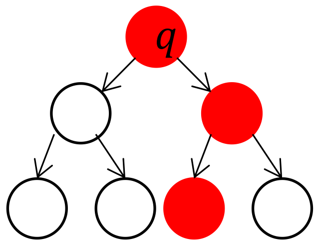


CTL: e.g., **AG** (a => **AF** b)

$CheckEG(f_1, Q, T, L)$

From $\mathcal{A}$ we construct a new automaton $\mathcal{A}' = \langle Q', T', L' \rangle$ such that

$Q' = \{q \in Q \mid f_1 \in label(q)\}$



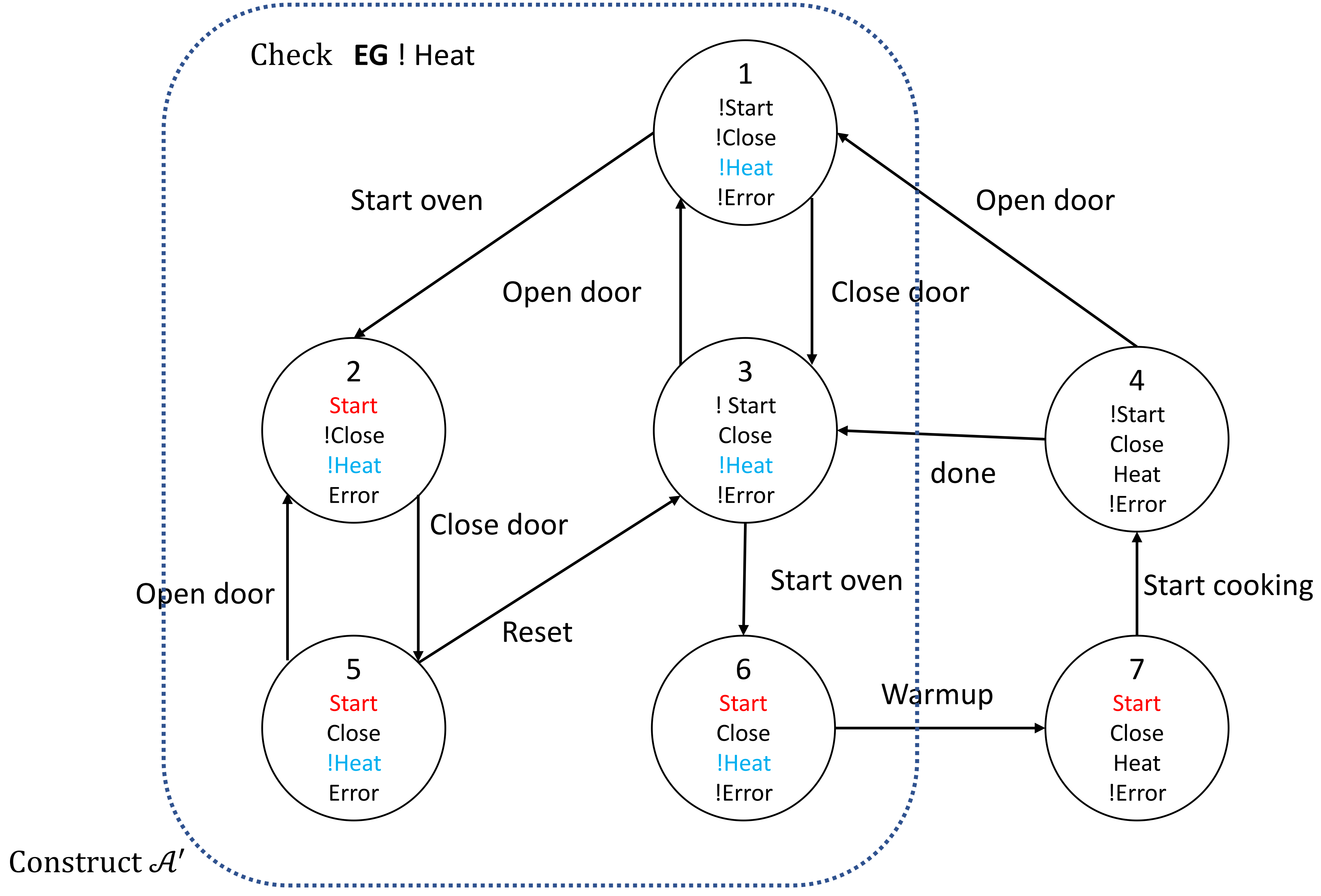
$T' = \{\langle q_1, q_2 \rangle \in T \mid q_1 \in Q'\}$

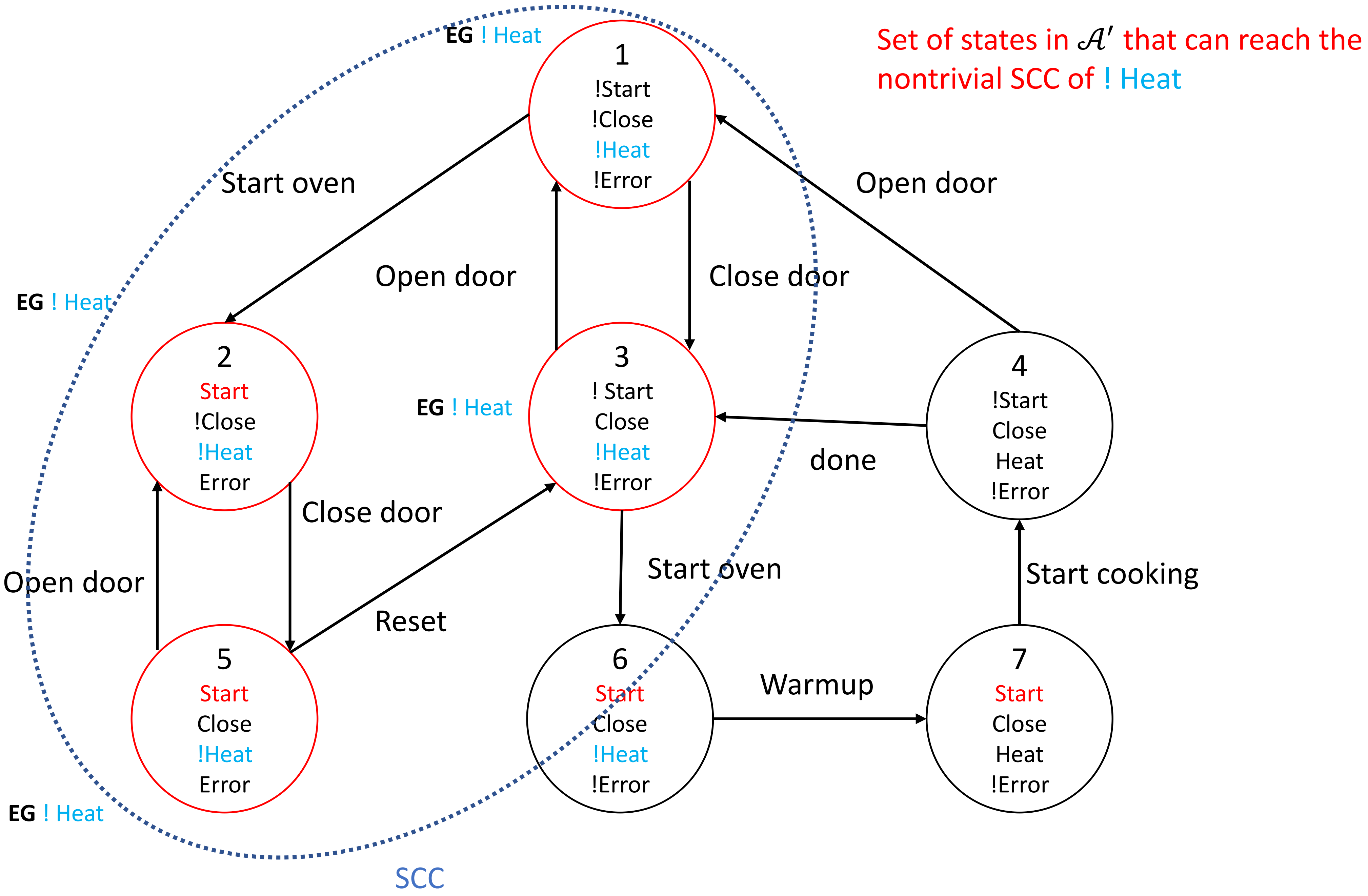
$L': Q' \to 2^{AP}\ \forall\, q' \in Q',\ L'(q') := L(q')$

Claim. q $\vDash EGf_1$ iff

(1) $q \in Q'$

(2) $\exists \alpha \in Execs_{\mathcal{A}}$, with $\alpha.fstate = q$ and $\alpha.lstate$ is in a nontrivial ***Strongly Connected Components*** C of the graph $\langle Q', T' \rangle$


Check EG ! Heat
1
!Start
!Close
!Heat
!Error
Start oven
Open door
Close door
Open door
2
Start
!Close
!Heat
Error
3
! Start
Close
!Heat
!Error
4
!Start
Close
Heat
!Error
done
Close door
Start oven
Start cooking
Open door
Reset
5
Start
Close
!Heat
Error
6
Start
Close
!Heat
!Error
Warmup
7
Start
Close
Heat
!Error
Construct $\mathcal{A}'$



Set of states in $\mathcal{A}'$ that can reach the nontrivial SCC of ! Heat
EG ! Heat
1
!Start
!Close
!Heat
!Error
Start oven
Open door
Close door
Open door
EG ! Heat
2
Start
!Close
!Heat
Error
EG ! Heat
3
! Start
Close
!Heat
!Error
4
!Start
Close
Heat
!Error
done
Close door
Start oven
Start cooking
Open door
Reset
5
Start
Close
!Heat
Error
6
Start
Close
!Heat
!Error
Warmup
7
Start
Close
Heat
!Error
EG ! Heat
SCC

Claim. $\mathcal{A}$, q $\vDash EGf_1$ iff

(1) $q \in Q'$ and

(2) $\exists\alpha \in Execs_{\mathcal{A}}$, with $\alpha.fstate = q$ and $\alpha.lstate$ is in a nontrivial SCC C of the graph $\langle Q', T' \rangle$

Proof. Suppose $\mathcal{A}$, q $\vDash EGf_1$

Consider any execution α with $\alpha.fstate = q$. Obviously, $q \vDash f_1$ and so, $q \in Q'$.

Since Q is finite α can be written as $\alpha = \alpha_0\alpha_1$ where α_0 is finite and every state in α_1 repeats infinitely many times.

Let C be the states in α_1. $C \in Q'$.

Consider any two q_1 and q_2 states in C, we observe that $q_1 \rightleftarrows q_2$, and therefore C is a SCC.

Consider (1) and (2). We construct a path $\alpha = \alpha_0\alpha_1$ such that $\alpha_0.fstate = q$ and $\alpha_0 \in Q'$and $\alpha_1$visits some states infinitely often.

$CheckEG(f_1, Q, T, L)$

Let $Q' = \{q \in Q \mid f_1 \in label(q)\}$

Let $\mathbb{C}$ be the set of nontrivial SCCs of $\langle Q', T' \rangle$

$\boldsymbol{T} = \ \cup_{C \in \mathbb{C}} \{q \mid q \in C\}$

for each $q \in \boldsymbol{T}$

  $label(q) \coloneqq label(q) \cup \{EGf_1\}$

while $\boldsymbol{T} \neq \emptyset$

  for each $q \in \boldsymbol{T}$

    $\boldsymbol{T} \coloneqq \boldsymbol{T} \setminus \{q\}$

    for each $q' \in Q'$ such that $(q', q) \in T'$

      if $\mathrm{E}Gf_1 \notin label(q')$ then

        $label(q') \coloneqq label(q') \cup \{EGf_1\}$

        $\boldsymbol{T} \coloneqq \boldsymbol{T} \cup \{q'\}$

Find all states in Q’ that can reach the SCCs

Proposition. For any state $label(q) \ni EGf_1$ iff $q \vDash EGf_1$.

Proposition. Finite Q therefore terminates and in $O(|Q| + |T|)$ steps.

$CheckEU(f_1, f_2, Q, T, L)$

Let $S = \{q \in Q \mid f_2 \in label(q)\}$

for each $q \in S$

  $label(q) := label(q) \cup \{E[f_1 U f_2]\}$

all states where f_2 is true already satsifies

while $S \neq \emptyset$

  for each $q' \in S$

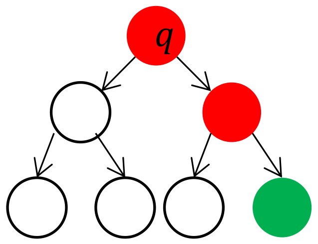
    $S := S \setminus \{q'\}$

    for each $q \in T^{-1}(q')$ *Check all states whose next state is q'*

      if $f_1 \in label(q)$ then *This if statement will be always true for EF f_2*

        $label(q) := label(q) \cup \{E[f_1 U f_2]\}$

          $S := S \cup \{q\}$



$E[f_1 U f_2]$

Proposition. For any state $label(q) \ni E[f_1 U f_2]$ iff $q \models E[f_1 U f_2]$.

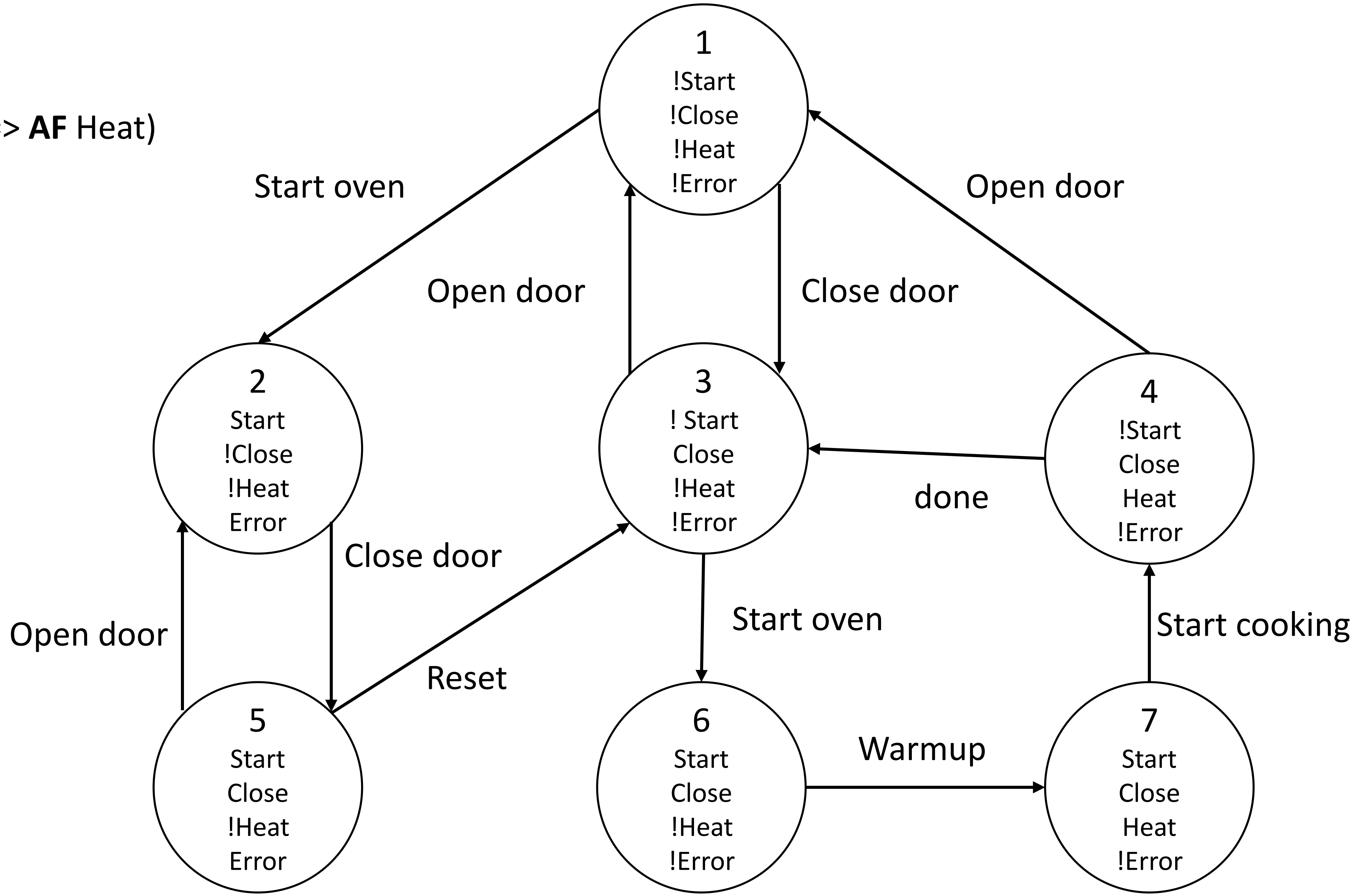
Proposition. Finite Q therefore terminates and in $O(|Q| + |T|)$ steps.

Putting it all together

Explicit model checking algorithm input$\mathcal{A} \vDash f$?
Structural induction over CTL formula: starting with APs at lowest depth and keep updating labels for each state

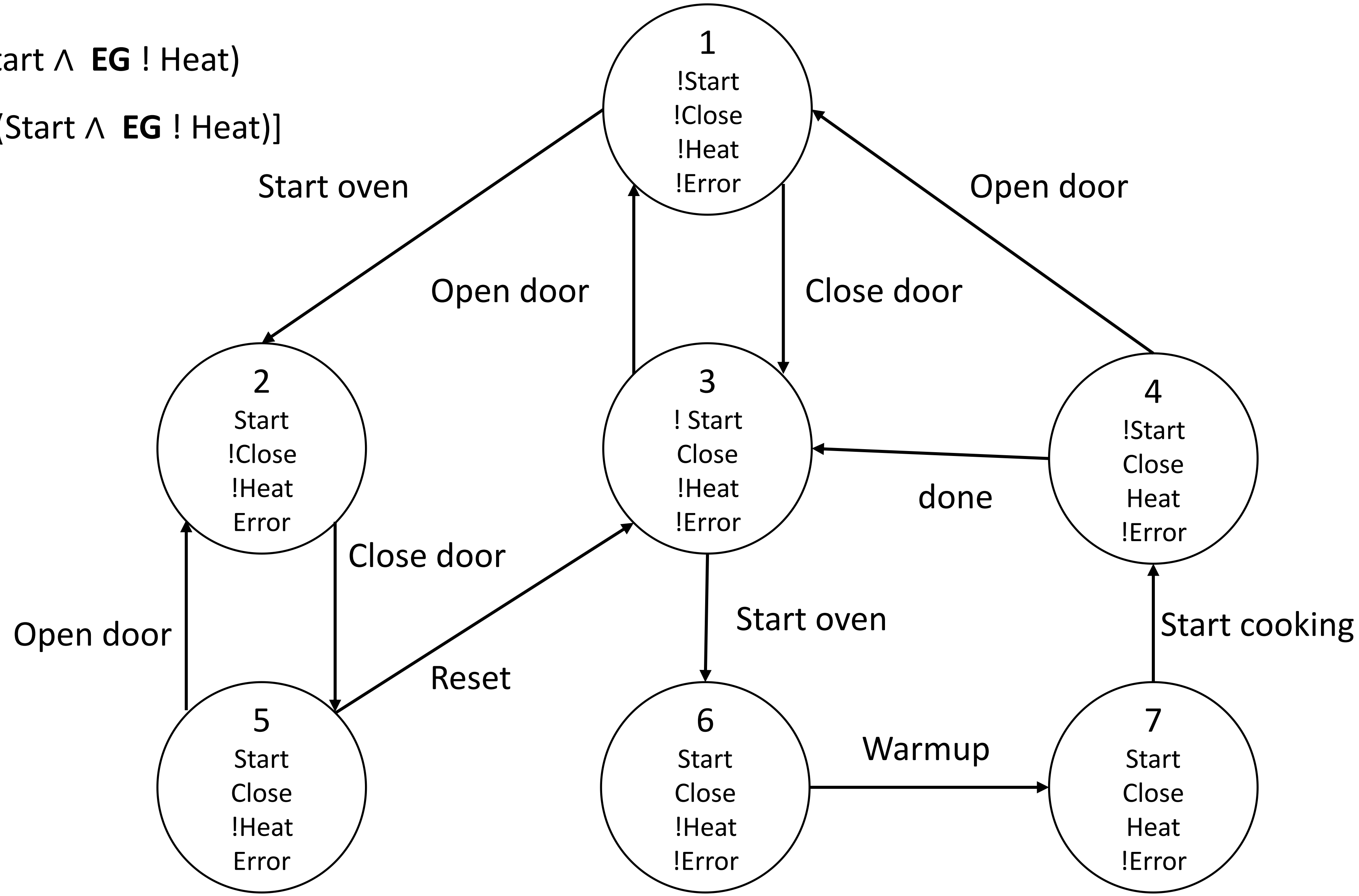
$f = p,$	for some $p \in AP, \forall q,\ label(q) \coloneqq label(q) \cup \{p\}$
$f = \neg f_1$	if $f_1 \notin label(q)$ then $label(q) \coloneqq label(q) \cup f$
$f = f_1 \wedge f_2$	if $f_1,\ f_2 \in label(q)$ then $label(q) \coloneqq label(q) \cup f$
$f = EXf_1$	if $\exists q' \in$ Q such that $(q, q') \in T$ and $f_1 \in label(q')$ then $label(q) \coloneqq label(q) \cup f$
$f = E[f_1 U f_2]$	$CheckEU(f_1, f_2, Q, T, L)$
$f = EGf_1$	$CheckEG(f_1,\ Q, T, L)$

AG (Start => **AF** Heat)



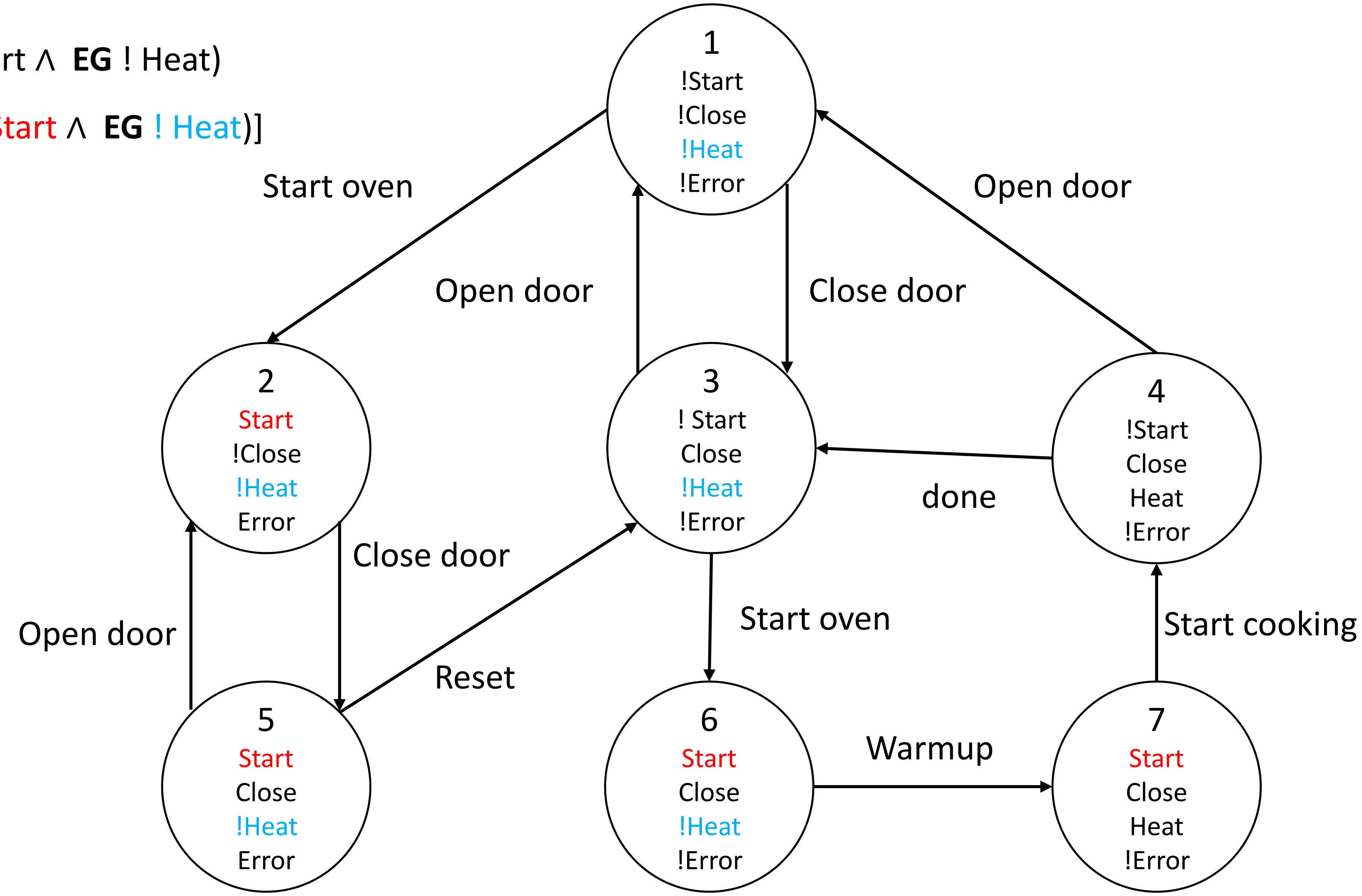
$$\mathbf{!\,EF}\ (\text{Start} \wedge\ \mathbf{EG}\ !\,\text{Heat})$$

$$\mathbf{!\,[}\text{True}\ \mathbf{EU}\ (\text{Start} \wedge\ \mathbf{EG}\ !\,\text{Heat})]$$



! EF (Start ∧ **EG** ! Heat)

! [True **EU** (Start ∧ **EG** ! Heat)]

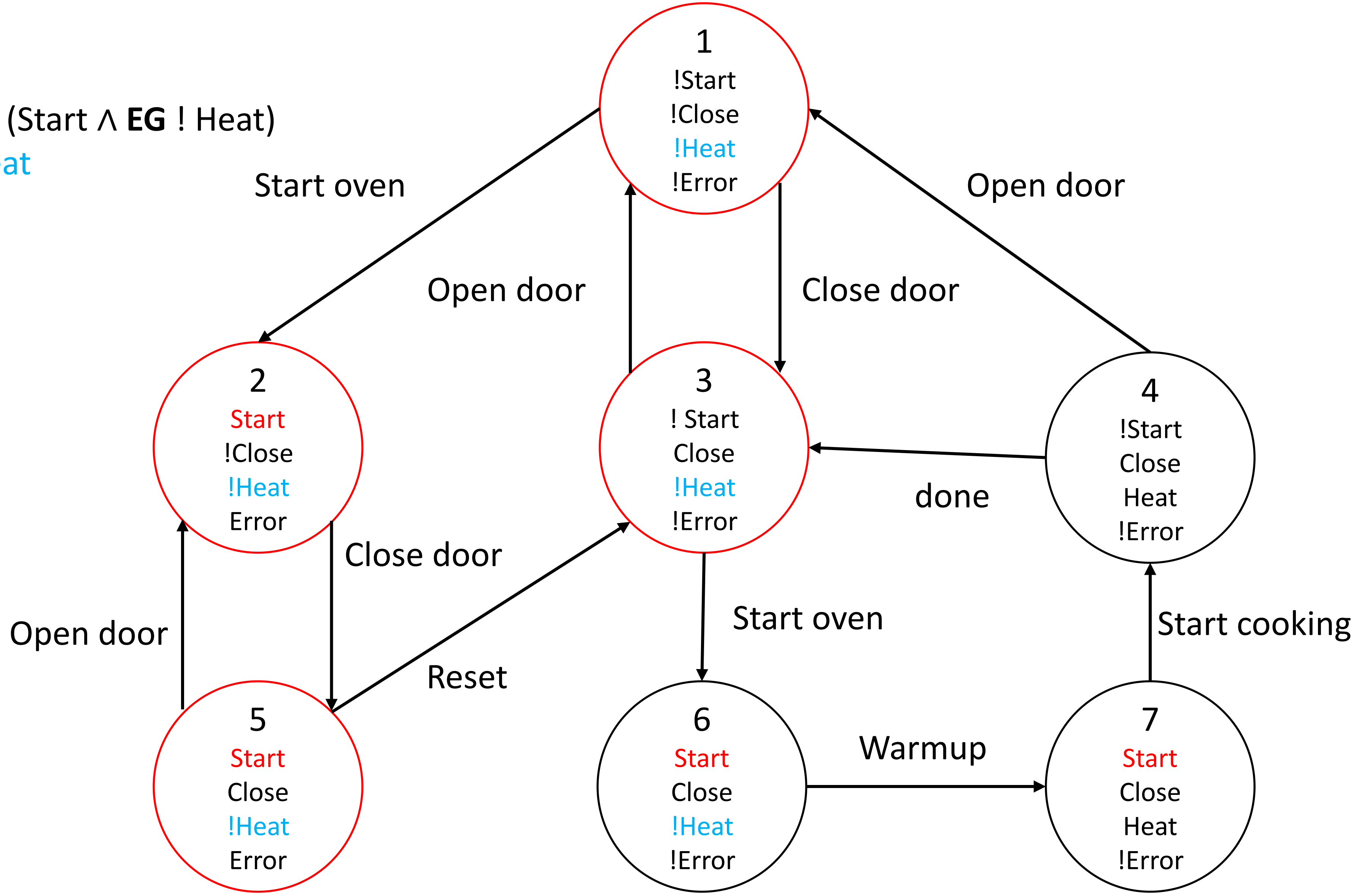


Nontrivial SCC of ! Heat

Goal: ! EF (Start ∧ **EG** ! Heat)

Start, ! Heat

EG ! Heat



Set of states that can reach the nontrivial SCC of ! Heat

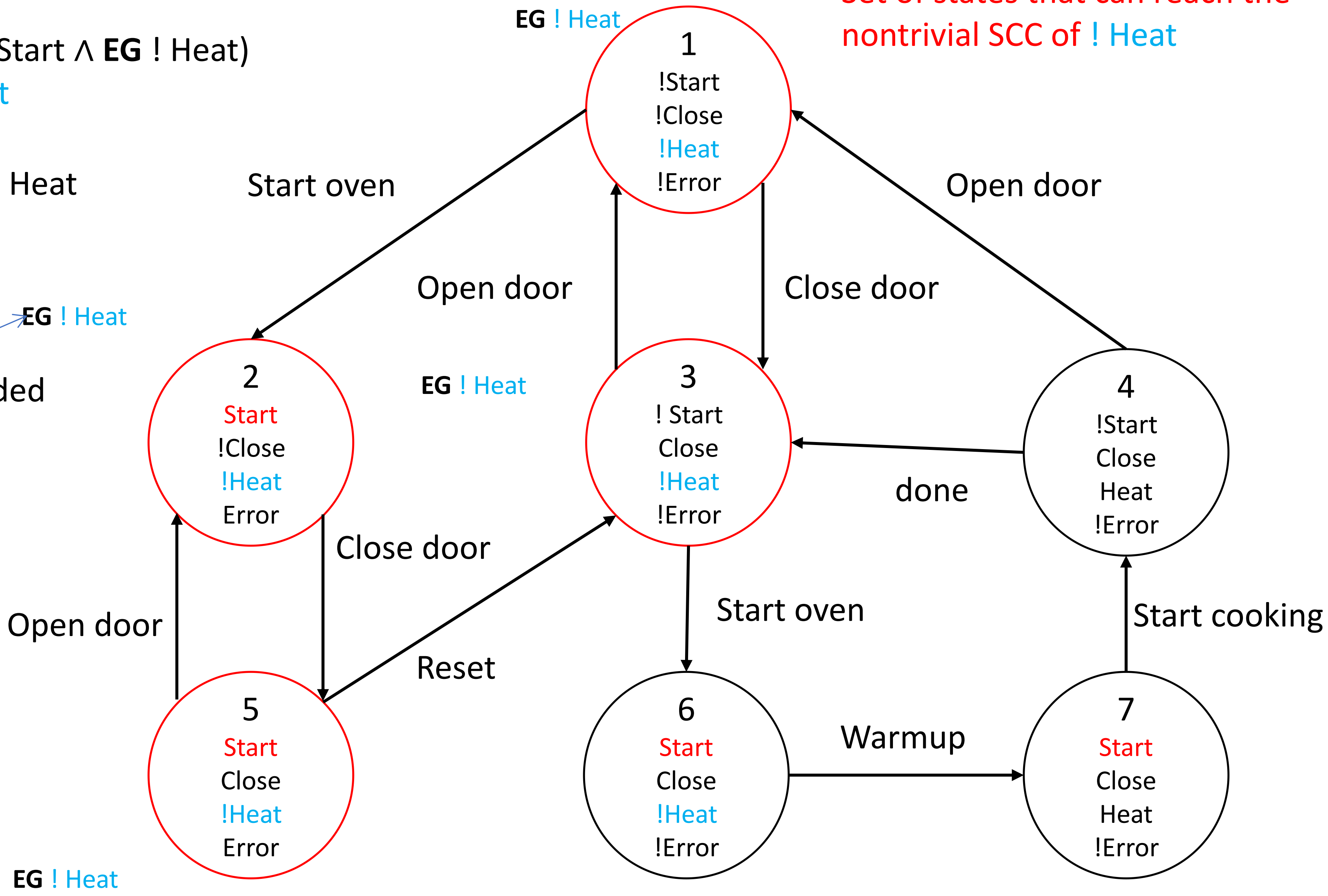
Goal: ! EF (Start ∧ **EG** ! Heat)

Start, ! Heat

EG ! Heat

Start ∧ **EG** ! Heat

New label added for node 2

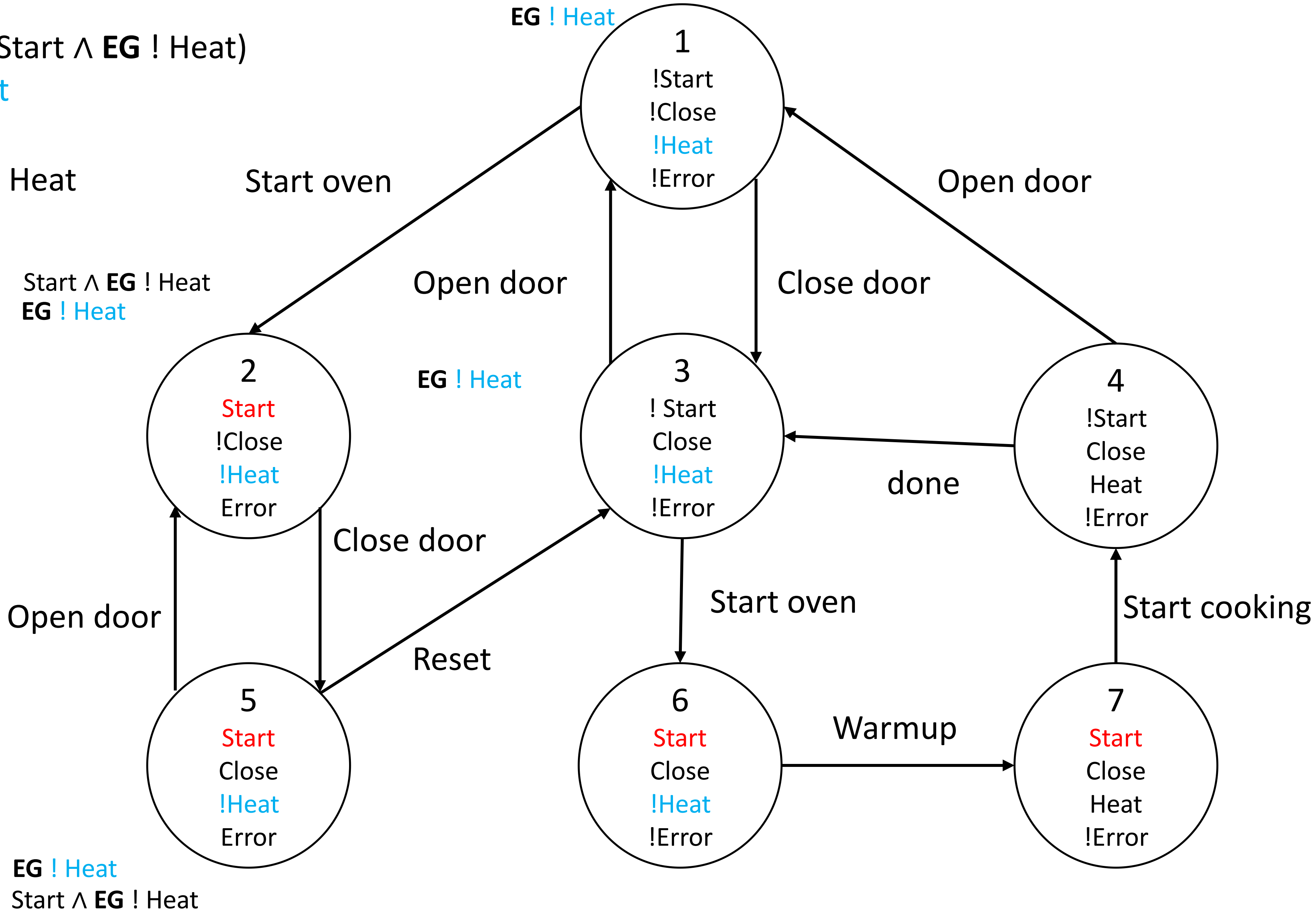


Goal: ! EF (Start ∧ **EG** ! Heat)

Start, ! Heat

EG ! Heat

Start ∧ **EG** ! Heat



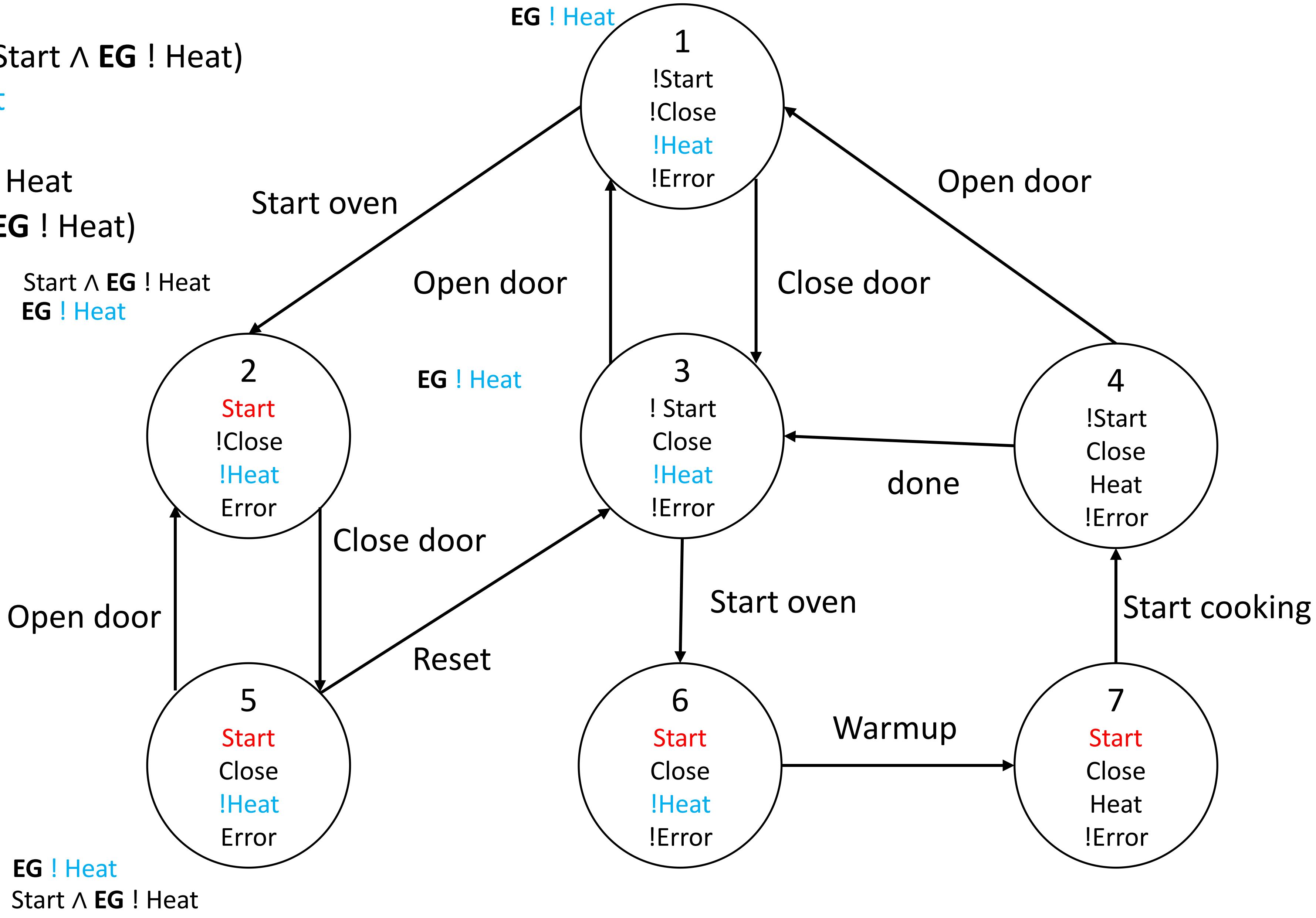
Goal: ! EF (Start ∧ **EG** ! Heat)

Start, ! Heat

EG ! Heat

Start ∧ **EG** ! Heat

EF (Start ∧ **EG** ! Heat)



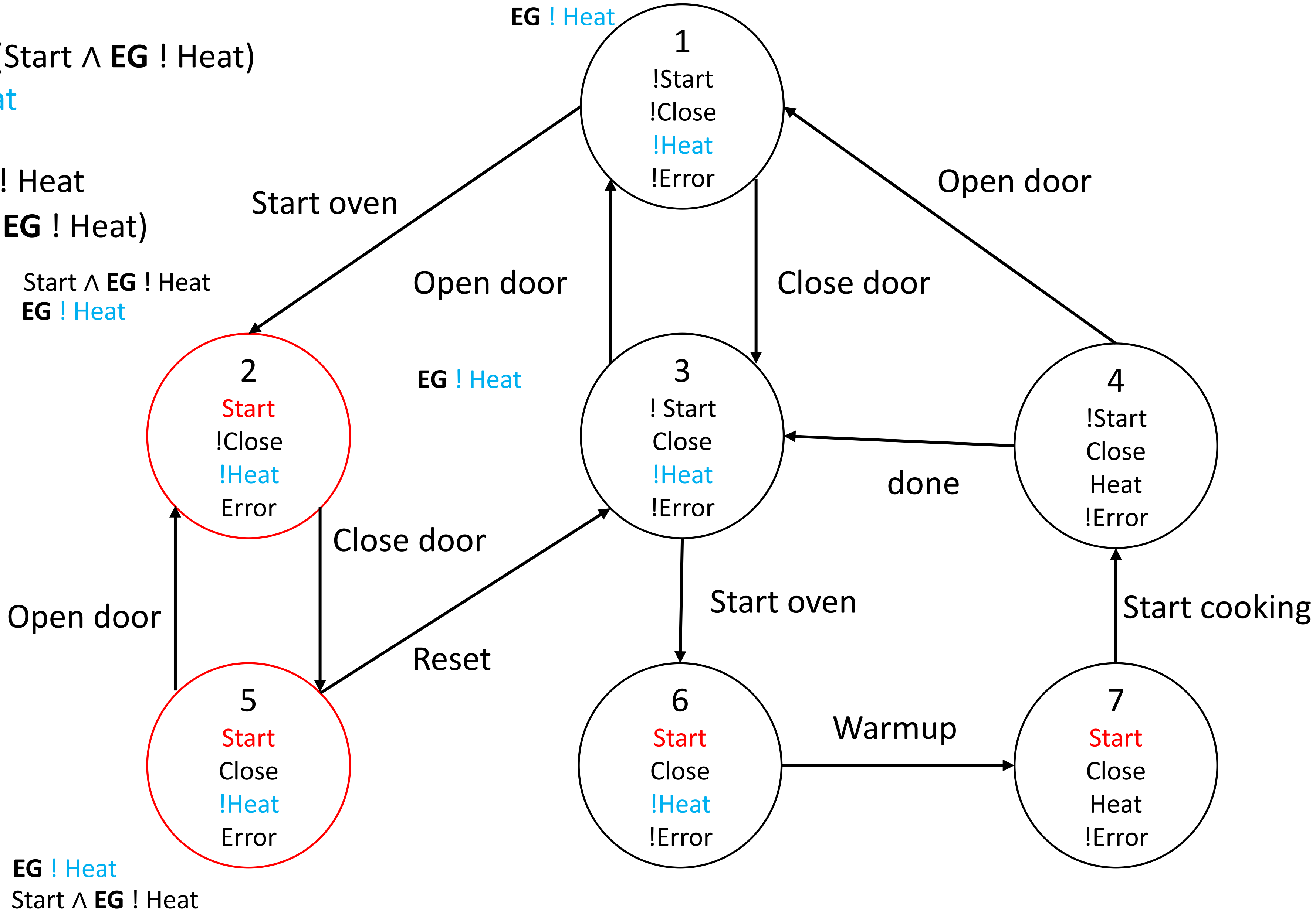
Goal: ! EF (Start ∧ **EG** ! Heat)

Start, ! Heat

EG ! Heat

Start ∧ **EG** ! Heat

EF (Start ∧ **EG** ! Heat)



Goal: ! EF (Start ∧ **EG** ! Heat)

Start, ! Heat

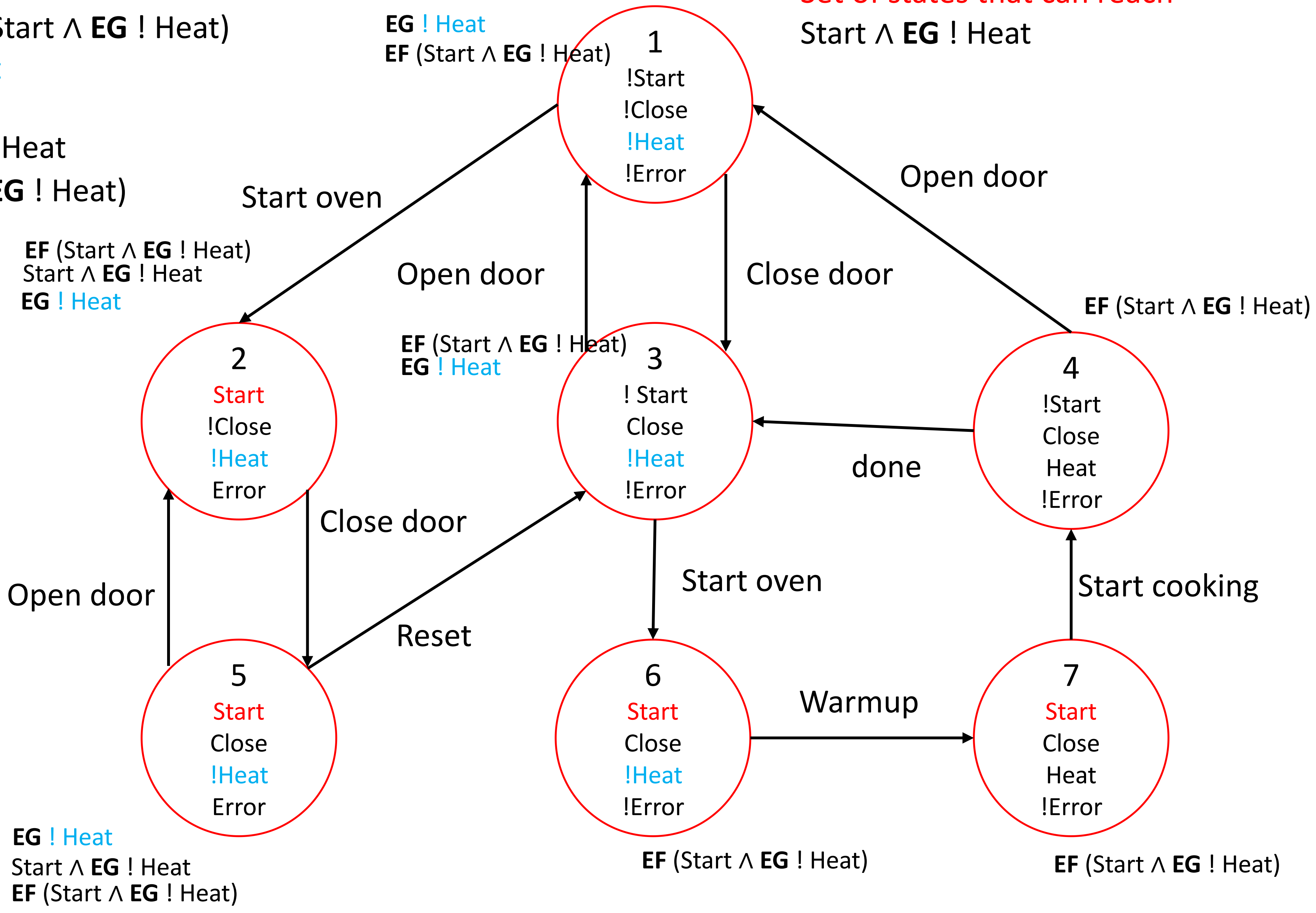
EG ! Heat

Start ∧ **EG** ! Heat

EF (Start ∧ **EG** ! Heat)

Set of states that can reach

Start ∧ **EG** ! Heat



! EF (Start ∧ **EG** ! Heat)

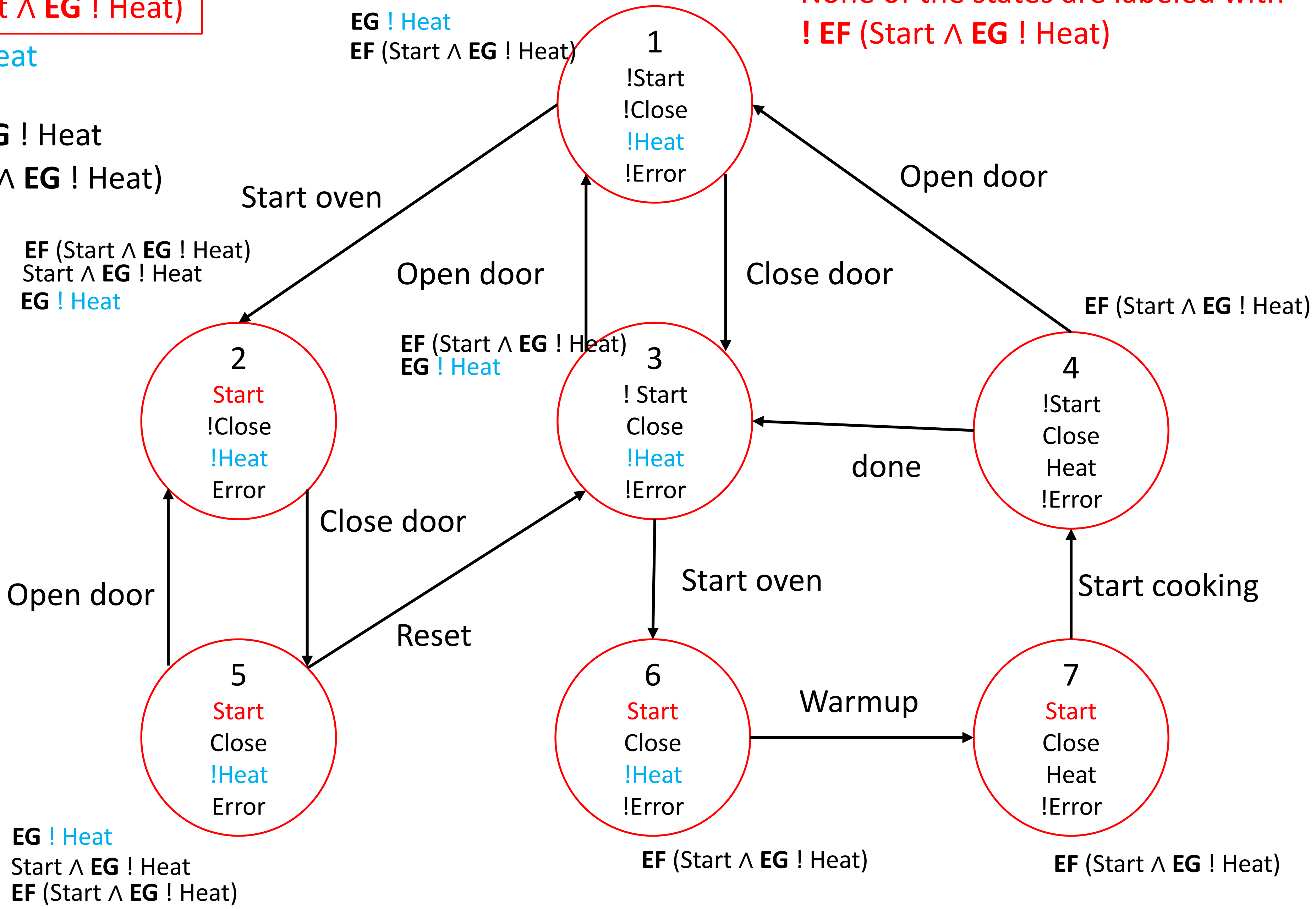
Start, ! Heat

EG ! Heat

Start ∧ **EG** ! Heat

EF (Start ∧ **EG** ! Heat)

None of the states are labeled with
! EF (Start ∧ **EG** ! Heat)



Next topic: Timed Automata & Reachability

- We have studied hybrid automaton

automaton Bouncingball(c,h,g)

variables: x: Reals := h, v: Reals := 0

actions: bounce

transitions:

bounce

pre $x = 0 \wedge v < 0$

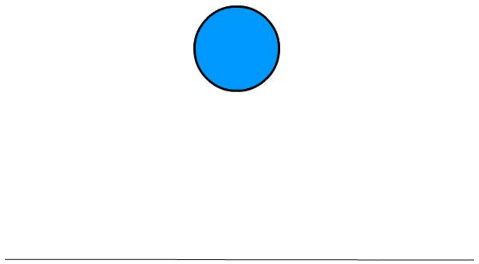
eff v := -cv

trajectories:

freefall

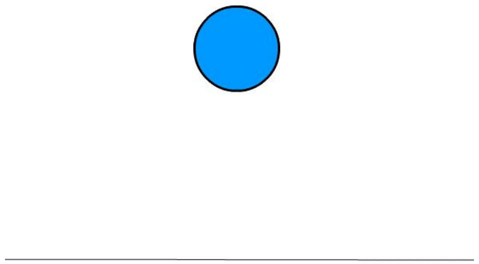
evolve d(x) = v; d(v) = -g

invariant $\boldsymbol{x \geq 0}$



Next topic: Timed Automata & Reachability

- We have studied hybrid automaton
- However, verification for general hyrbid automaton is in general difficult



- Special classes of hybrid automaton:
  - (Alur-Dill's) Timed Automata
  - Rectangular initialized hybrid automata
  - Linear hybrid automata
- Verification is feasible for these classes
  - New techniques: abstraction (will be covered after ~2 weeks)

Clocks and Clock Constraints

- A **clock variable** x is a continuous (analog) variable of type real such that along any trajectory τ of x, for all t ∈ $\tau.dom,\ (\tau \downarrow x)(t) = t$.
- In other words, d(x) = 1

- For a set X of clock variables, the set Φ(X) of **integral clock constraints** are expressions defined by the syntax:
  g ::= x $\leq q \mid x \geq q \mid \neg\, g \mid g_1 \wedge g_2$
  where $x \in X\ and\ q \in \mathbb{Z}$

- Examples: x = 10; x ∈ [2, 5] are valid clock constraints
- What do clock constraints look like?

Example: “smart” light switch

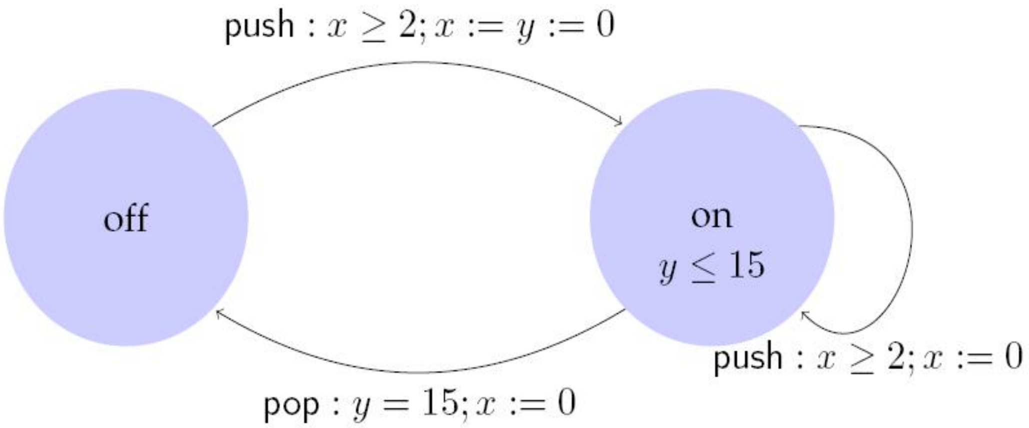
Description
Switch can be turned on whenever at least 2 time units have elapsed since the last turn off. Switches can be turned off 15 time units after the last on.

automaton Switch
variables x, y:Real := 0, loc: {on,off} := off

transitions
push
pre x ≥ 2
integral clock constraints
eff if loc = off **then** x,y := 0; loc := on
else x := 0
pop
pre y = 15 /\ loc = on
eff x := 0; loc = off

trajectories
invariant loc = off \/ y ≤ 15
evolve d(x) = 1; d(y) = 1

clock variables



Integral Timed Automata

- **Definition.** A **integral timed automaton** $\mathcal{A} = \langle V,\ \Theta,\ A,\ \mathcal{D},\ \mathcal{T} \rangle$ where
  - V = X ∪ $\{l\}$, where X is a set of n clocks and l is a discrete state variable of finite type L
  - A is a finite set
  - $\mathcal{D}$ is a set of transitions such that
    - The preconditions are described by clock constraings $\Phi(X)$
    - $\langle x, l \rangle_a \rightarrow \langle x', l' \rangle$ implies either $x' = x$ or $x' = 0$ (time is reset to 0, or no change)
  - $\mathcal{T}$ set of clock trajectories for the clock variables in X