Formal Methods in Software Development

Ivano Salvo
and Igor Melatti

Computer Science Department

SAPIENZA
UNIVERSITÀ DI ROMA

Lesson **3**, October 8^{th}, 2019

Lesson 2d:

Summary of LTL Model Checking

LTL model checking: summary

The problem $\mathcal{M}, s \models \mathbf{A} f$ is transformed in the refutation of $\mathcal{M}, s \models \neg\mathbf{E}\, \neg f$. To verify $\mathcal{M}, s \models \mathbf{E} f$ where $\mathcal{M} = (S, R, L)$:

1. Build the set of formulas: CL(f).

2. For each state $s \in S$, compute sets K of formulas in CL(f) consistent with $L(s)$.

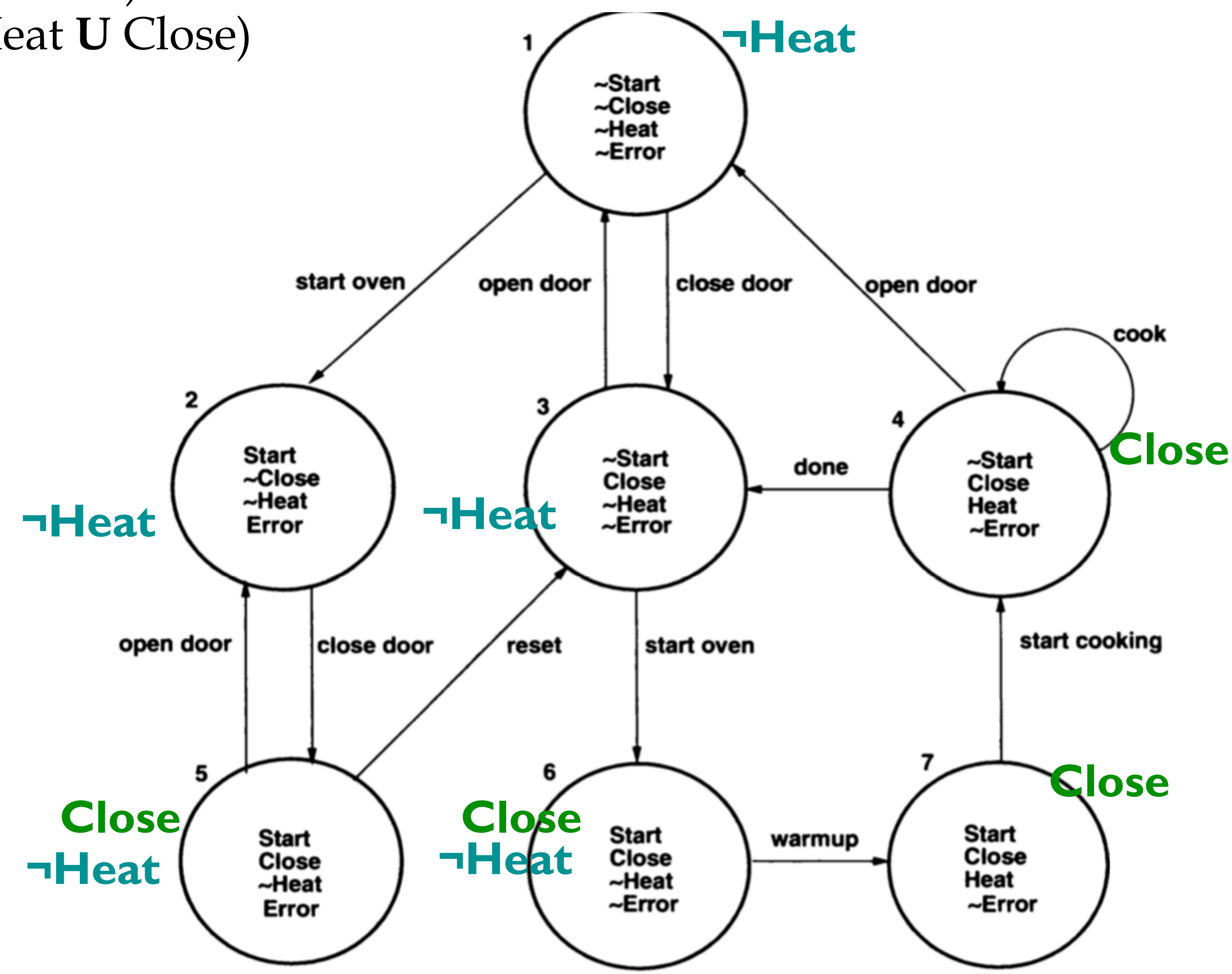
3. Build the graph $G^{\mathcal{M}, f}$, that contain an edge from (s, K) to (s', K') whenever (s, s') is in R, $\mathbf{X}\, g$ in K, and g in K'.

4. Find an eventuality sequence by finding strongly connected components of $G^{\mathcal{M}, f}$

[An **eventuality sequence** is an infinite path π in $G^{\mathcal{M}, f}$ such that if $g_1\, \mathbf{U}\, g_2 \in K$ for some atom (s, K) then there exists an atom (s', K') reachable from (s, K) along π, such that $g_2 \in K'$].

Example: microwave oven

A(¬Heat **U** Close)
≡ ¬**E** ¬ (¬Heat **U** Close)



LTL model checking: example 1

Taking $f \equiv (\neg\text{Heat } \mathbf{U} \text{ Close})$

- Compute the closure of f, $CL(\neg f)$:

  $\{\neg f, f, \mathbf{X} f, \neg \mathbf{X} f, \mathbf{X} \neg f, \text{Heat}, \neg\text{Heat}, \text{Close}, \neg\text{Close}\}$

- Compute atoms:

Not just subformulas!

  - $\{\neg\text{Heat}, \neg\text{Close}\} \subseteq L(1), L(2)$

    $K_1' = \{\neg\text{Heat}, \neg\text{Close}, f, \mathbf{X} f\}$

    $K_1'' = \{\neg\text{Heat}, \neg\text{Close}, \neg f, \neg\mathbf{X} f, \mathbf{X} \neg f\}$

  - $\{\neg\text{Heat}, \text{Close}\} \subseteq L(3), L(5), L(6)$

Close is not consistent with $\neg f$

    $K_2' = \{\neg\text{Heat}, \text{Close}, f, \mathbf{X} f\}$

    $K_2'' = \{\neg\text{Heat}, \text{Close}, f, \neg\mathbf{X} f, \mathbf{X} \neg f\}$

  - $\{\text{Heat}, \text{Close}\} \subseteq \mathrm{L}(4), \mathrm{L}(7)$

    $K_3' = \{\text{Heat}, \text{Close}, f, \mathbf{X} f\}$

    $K_3'' = \{\text{Heat}, \text{Close}, f, \neg\mathbf{X} f, \mathbf{X} \neg f\}$

Compute the graph G

Example of transitions:

$(1, K_1') \rightarrow (2, K_1')$ because $\mathbf{X} f \in K_1'$, $f \in K_1'$, and $(1,2) \in R$

$(1, K_1'') \rightarrow (2, K_1'')$ because $\mathbf{X} \neg f \in K_1'$, $\neg f \in K_1''$, and $(1,2) \in R$

There is no transition $(1, K_1') \rightarrow (2, K_1'')$ since $\mathbf{X} f \in K_1'$ but $f \notin K_1''$

Once the full graph is constructed, it is easy to see that there is no atom (s, K) from which there is a path into a self-fullfilling non trivial strong component of $G^{\mathcal{M}, f}$.

Therefore, no state s is such that $\mathcal{M}, s \vDash \mathbf{E} \neg f$ and hence all states satisfy $\mathcal{M}, s \vDash \mathbf{A}\, g$.

Lesson 3a:

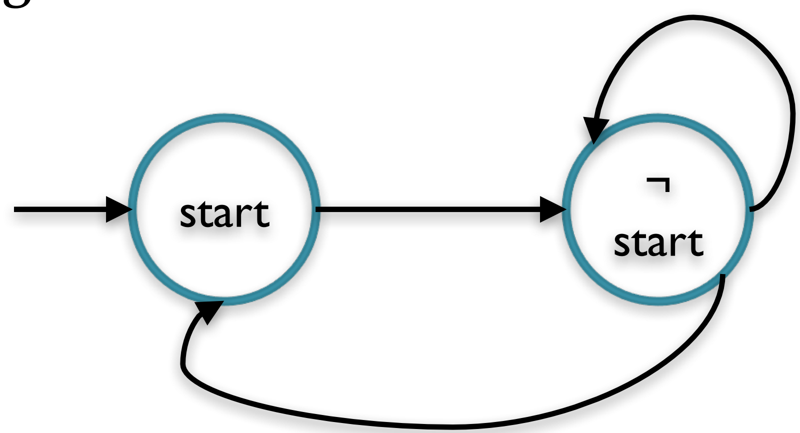
Computation Tree Logic CTL

Non Linear Time properties

"For every computation, it is always possible to return to the initial state"

A G F start

does not properly work.
It is **too strong**.



This system intuitively satisfies our intended property, but not the linear property **A G F** start

The solution is a **branching notion** of time, allowing nesting of path quantifiers **A** and **E**: in this case **A G E F** start.

CTL: syntax

State formulas are formulas that depend on a state of a transition system

- If $p \in AP$, then p is a state formula
- If f, g are state formulas, then so are $\neg f$, $f \wedge g$, $f \vee g$
- If f is a path formula, the **A** f and **E** f are state formulas

Path formulas are formulas that depend on a computation path

- If f, g are **state** formulas, then $\neg f$, $f \wedge g$, $f \vee g$, **X** f, **F** f, **G** f, f **U** g, and f **R** g are path formulas

Similar to CTL*, but **each temporal operator** (**X**, **F**, **G**, **U**, **R**) **must be preceded by a path quantifier** (**E** or **A**)

Examples: (il)legal CTL formulas

Let $AP = \{x = 1, x < 2, x \geq 3\}$ be the set of atomic propositions.

Legal CTL formulas are:

$\mathbf{EX}\ (x = 1)$, $\mathbf{AX}\ (x = 1)$, $x = 1 \vee x < 2$

Illegal CTL formulas are:

$\mathbf{E}\ (x = 1 \wedge \mathbf{AX}\ x \geq 3)$

because $\mathbf{AX}\ x \geq 3$ is not a path formula

$\mathbf{EX}\ (\text{true}\ \mathbf{U}\ x = 1)$

because **EX** nested with a path formula

By contrast, the following are legal CTL formulas:

$\mathbf{EX}\ (x = 1 \wedge \mathbf{AX}\ x \geq 3)$ $\mathbf{EX}\ \mathbf{A}\ (\text{true}\ \mathbf{U}\ x = 1)$

Common operators: $\mathbf{EF}\ \varphi \equiv$ "φ holds potentially"

$\mathbf{AF}\ \varphi \equiv$ "φ is inevitable"

$\mathbf{EG}\ \varphi \equiv$ "φ holds potentially always"

$\mathbf{AG}\ \varphi \equiv$ "invariantly φ"

Minimal Fragment of CTL

From a theoretical point of view, only 3 operators are really needed: **EX**, **EG**, and **EU**:

$$\mathbf{AX}\, f \equiv \neg\, \mathbf{EX}\, \neg f$$

$$\mathbf{EF}\, f \equiv \neg\, \mathbf{E}\,(\text{true}\; \mathbf{U}\, f\,)$$

$$\mathbf{AG}\, f \equiv \neg\, \mathbf{EF}\, \neg f$$

$$\mathbf{AF}\, f \equiv \neg\, \mathbf{EG}\, \neg f$$

$$\mathbf{A}(f\, \mathbf{U}\, g\,) \equiv \neg\, \mathbf{E}\,(\neg g\, \mathbf{U}\, \neg f \wedge \neg g) \wedge \neg\, \mathbf{EG}\, \neg g$$

$$\mathbf{A}(\,f\, \mathbf{R}\, g\,) \equiv \neg\, \mathbf{E}\,(\neg f\, \mathbf{U}\, \neg g)$$

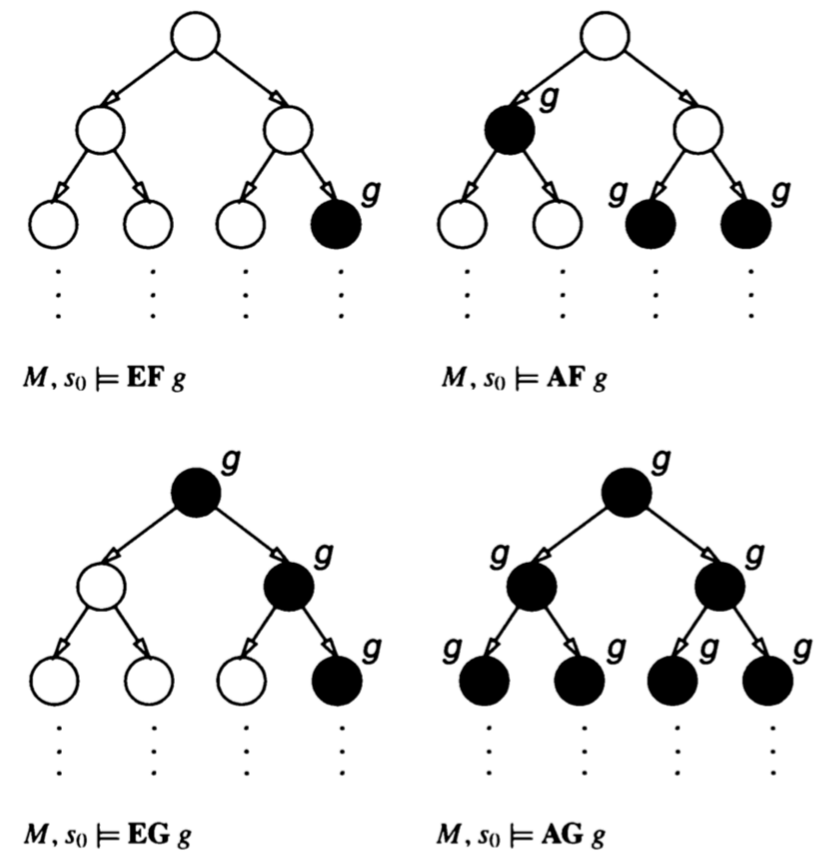
$$\mathbf{E}\,(\,f\, \mathbf{R}\, g\,) \equiv \neg\, \mathbf{A}\,(\neg f\, \mathbf{U}\, \neg g)$$

Attention! that propositional operators ($\wedge$, $\vee$, $\neg$, etc.) **cannot be applied to path formula**, so it is not true that **EG** f $\equiv$ **E** $\neg$ **F** $\neg f$ simply because the latter **is not** a CTL formula.

Non Linear Time (LT) examples

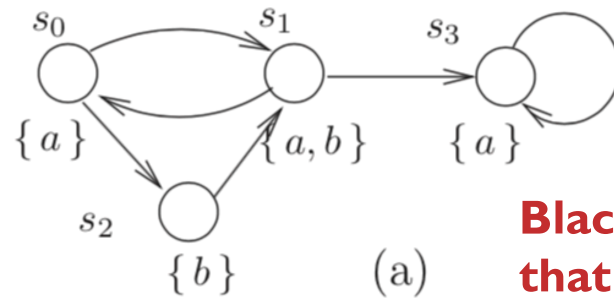
The semantics of CTL* formulas are relative to a computation Tree.

Here some example of computation trees and CTL* formulas valid in such computation trees.



Other Examples

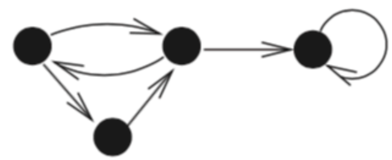
TS



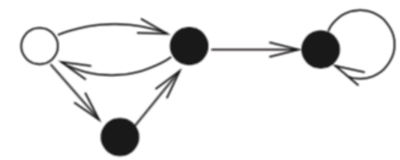
(a)

Black states are those that satisfy the formula

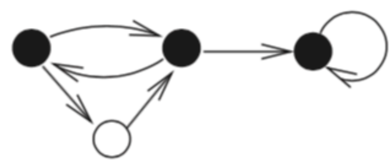
$\exists \bigcirc a$



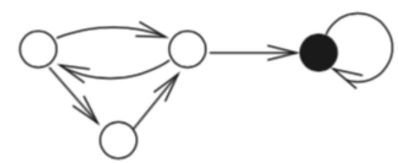
$\forall \bigcirc a$



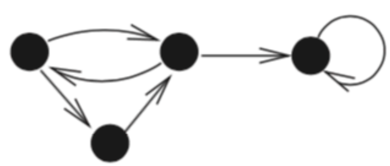
$\exists \Box a$



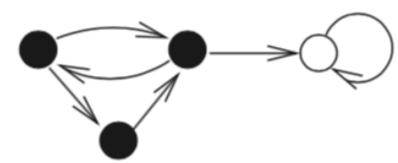
$\forall \Box a$



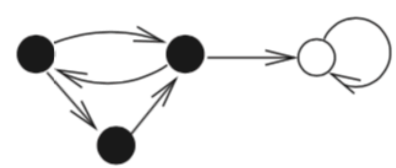
$\exists \Diamond (\exists \Box a)$



$\forall (a \mathsf{U} b)$



$\exists (a \mathsf{U} (\neg a \wedge \forall (\neg a \mathsf{U} b)))$



A remark on negation

A transition system $\mathcal{M}$ satisfies a CTL formula φ, notation $\mathcal{M} \models \varphi$ if and only if $\mathcal{M}, s \models \varphi$ for all $s \in S_0$, where S_0 is the set of initial states of $\mathcal{M}$.

Be careful that $\mathcal{M}, s \not\models \varphi$ implies $\mathcal{M}, s \models \neg\varphi$, but **it is not true** that $\mathcal{M} \not\models \varphi$ implies $\mathcal{M} \models \neg\varphi$ (**The same holds for LTL!**).

The problem is the universal quantification over initial states!

Example: Both a and $\neg a$ does not hold here:

Equivalent CTL formulas

A CTL formula f is equivalent to g if and only if for all transition system $\mathcal{M}$, $\mathcal{M} \vDash f$ iff $\mathcal{M} \vDash g$

Expansion Laws for CTL:

$$\mathbf{A}\,(f\,\mathbf{U}\,g) \equiv g \vee (f \wedge \mathbf{AX}\ \mathbf{A}(f\,\mathbf{U}\,g))$$

$$\mathbf{AG}\,f \equiv f \wedge \mathbf{AX}\ \mathbf{AG}\,f$$

$$\mathbf{AF}\,f \equiv f \vee \mathbf{AX}\ \mathbf{AF}\,f$$

$$\mathbf{E}\,(f\,\mathbf{U}\,g) \equiv g \vee (f \wedge \mathbf{EX}\ \mathbf{E}(f\,\mathbf{U}\,g))$$

$$\mathbf{EG}\,f \equiv f \wedge \mathbf{EX}\ \mathbf{EG}\,f$$

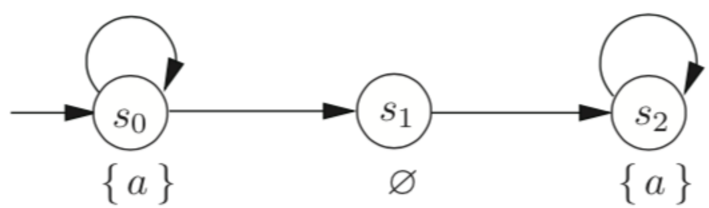
$$\mathbf{EF}\,f \equiv f \vee \mathbf{EX}\ \mathbf{EF}\,f$$

LTL versus CTL: eliminating A

Theorem. Let f be a CTL formula and let f^{LTL} be the LTL formula obtained by eliminating all path quantifiers in f. Then: $f \equiv f^{\text{LTL}}$ or there does not exist any LTL formula equivalent to f

Lemma. [PERSISTENCE] The CTL formula **A F A G** a and the LTL formula **F G** a are not equivalent.

Proof: Just consider the following Kripke structure.



We have $s_0 \vDash_{\text{LTL}}$ **F G** a, since all path starting in s_0 will remain forever in s_0 or in s_2 (that satisfy **G** a).

By contrast $s_0 \nvDash_{\text{CTL}}$ **A F A G** a, since $s_0^{\boldsymbol{\omega}} \nvDash_{\text{CTL}}$ **F A G** a because of the paths $s_0^{*}\, s_1\, s_2^{\boldsymbol{\omega}}$ which passes the $\neg a$-state s_1. ❑

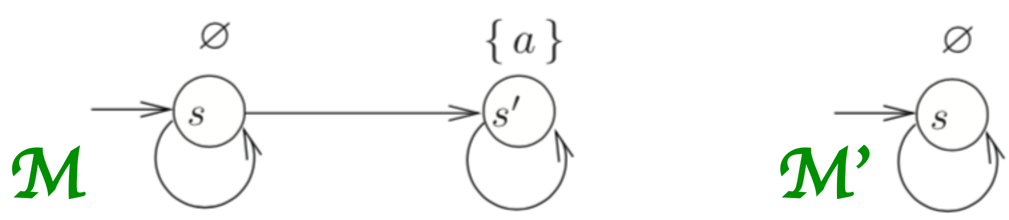
LTL and CTL are not comparable

Theorem.

1. There exist LTL formulas for which no equivalent CTL formula exist. For instance: $\mathbf{F}\,\mathbf{G}\,a$ or $\mathbf{F}\,(a \wedge \mathbf{X}\,a)$
2. There exist CTL formulas for which no equivalent LTL formula exist. For instance: $\mathbf{AF}\,\mathbf{AG}\,a$ or $\mathbf{AF}\,(a \wedge \mathbf{AX}\,a)$ or $\mathbf{AG}\,\mathbf{EF}\,a$

Proof (idea)**:** exhibit suitable transition systems $\mathcal{M}$ and $\mathcal{M}'$ such that $\mathcal{M} \models_{\text{LTL}} g$ and $\mathcal{M}' \not\models_{\text{LTL}} g$ but such that cannot be distinguished by any CTL formula, that is, for all CTL property g, $\mathcal{M} \models_{\text{CTL}} g$ if and only if f $\mathcal{M}' \models_{\text{CTL}} g$.

Let us consider $\mathbf{AG}\,\mathbf{EF}\,a$. This is satisfied by $\mathcal{M}$ above, but not by $\mathcal{M}'$. On the other hand since traces($\mathcal{M}'$)$\subseteq$traces($\mathcal{M}$), $\mathcal{M}'$ satisfies all LTL formulas satisfied by $\mathcal{M}$. ❑



Lesson 3b:

CTL Model Checking

CTL model checking 1

Idea: Compute a set *label*(s) in such a way that for each sub-formula g of f, $g \in label(s)$ whenever $\mathcal{M}, s \models g$ holds.

Observation: the number of sub-formulas are linear in the size $| f |$ of a CTL formula f.

Start with the original labeling of states with atomic propositions, i.e. $label(s) = L(s)$.

$g \equiv \neg h \Rightarrow g \in label(s)$ if and only if $h \notin label(s)$

$g \equiv h_1 \vee h_2 \Rightarrow g \in label(s)$ if and only if $h_1 \in label(s)$ or $h_2 \in label(s)$

$g \equiv \mathbf{E\,X}\,h \Rightarrow g \in label(s)$ if and only if $h \in label(s')$ for some s', $s \rightarrow s'$

The interesting cases are $g \equiv \mathbf{E\,G}\,h$ and $g \equiv \mathbf{E}\,[\,h_1\,\mathbf{U}\,h_2\,]$

CTL model checking 2

When $g \equiv \mathbf{E}\,[\,h_1\ \mathbf{U}\ h_2\,]$ the idea is: start from the set of states such that $h_2 \in label(s)$ and then proceed backwards on states such that $h_1 \in label(s)$. Label all these states with g.

procedure $CheckEU(f_1, f_2)$
  $T := \{\, s \mid f_2 \in label(s) \,\}$;
  for all $s \in T$ **do** $label(s) := label(s) \cup \{\, \mathbf{E}[f_1\ \mathbf{U}\ f_2] \,\}$;
  while $T \neq \emptyset$ **do**
    choose $s \in T$;
    $T := T \setminus \{s\}$;
    for all t **such that** $R(t, s)$ **do**
      if $\mathbf{E}[f_1\ \mathbf{U}\ f_2] \notin label(t)$ **and** $f_1 \in label(t)$ **then**
        $label(t) := label(t) \cup \{\, \mathbf{E}[f_1\ \mathbf{U}\ f_2] \,\}$;
        $T := T \cup \{t\}$;
      end if;
    end for all;
  end while;
end procedure

It is essentially a (backward) visit of a graph. The complexity is $O(|S| + |R|)$

CTL model checking 3

When $g \equiv \mathbf{E}\,\mathbf{G}\,h$, we must find infinite paths labeled by h.

In a finite directed graph, such path must enter a strongly connected component where all states are labeled by h.

Roughly speaking:

1. Compute the set of states $S' = \{\, s \in S \mid h \in \mathit{label}(s)\}$.
2. Decompose (S', R') in strongly connected components.
3. Add all states s such that $h \in \mathit{label}(s)$ and from which one of such strongly connected components is reachable.

CTL model checking 4

Lemma. Let $S' = \{ s' \in S \mid \mathcal{M}, s' \vDash h \}$. Then Let $\mathcal{M}, s \vDash \mathbf{E}\,\mathbf{G}\,h$ if and only if the following conditions are satisfied:

1. $s \in S'$

2. There exists a path from s to a strongly connected component $C \subseteq S'$ and of $\mathcal{M}'$.

Proof: (**If**) Let π be an infinite path starting at s satisfying $\mathbf{G}\,h$. Clearly, $s \vDash h$. Since π is an infinite, it has the shape $\pi_0\pi_1$ and in π_1 each state occurs infinitely often. Both states in π_0 and π_1 belongs to $C \subseteq S'$. Since each state appears infinitely often, there is a path between any pairs of state in C that is C is a SCC.

(**Only If**) There is a finite path π_0 from s to $t \in C$ in S'. Then we can find a finite path π_1 from t back to t. The path $\pi_0\, \pi_1^{\boldsymbol{\omega}}$ satisfies $\mathbf{G}\,h$. ❑

Theorem. Given a Kripke structure $\mathcal{M}=(S, R, L)$ and a CTL formula f, determining if $\mathcal{M} \vDash_{\text{CTL}} g$ can be decided in time

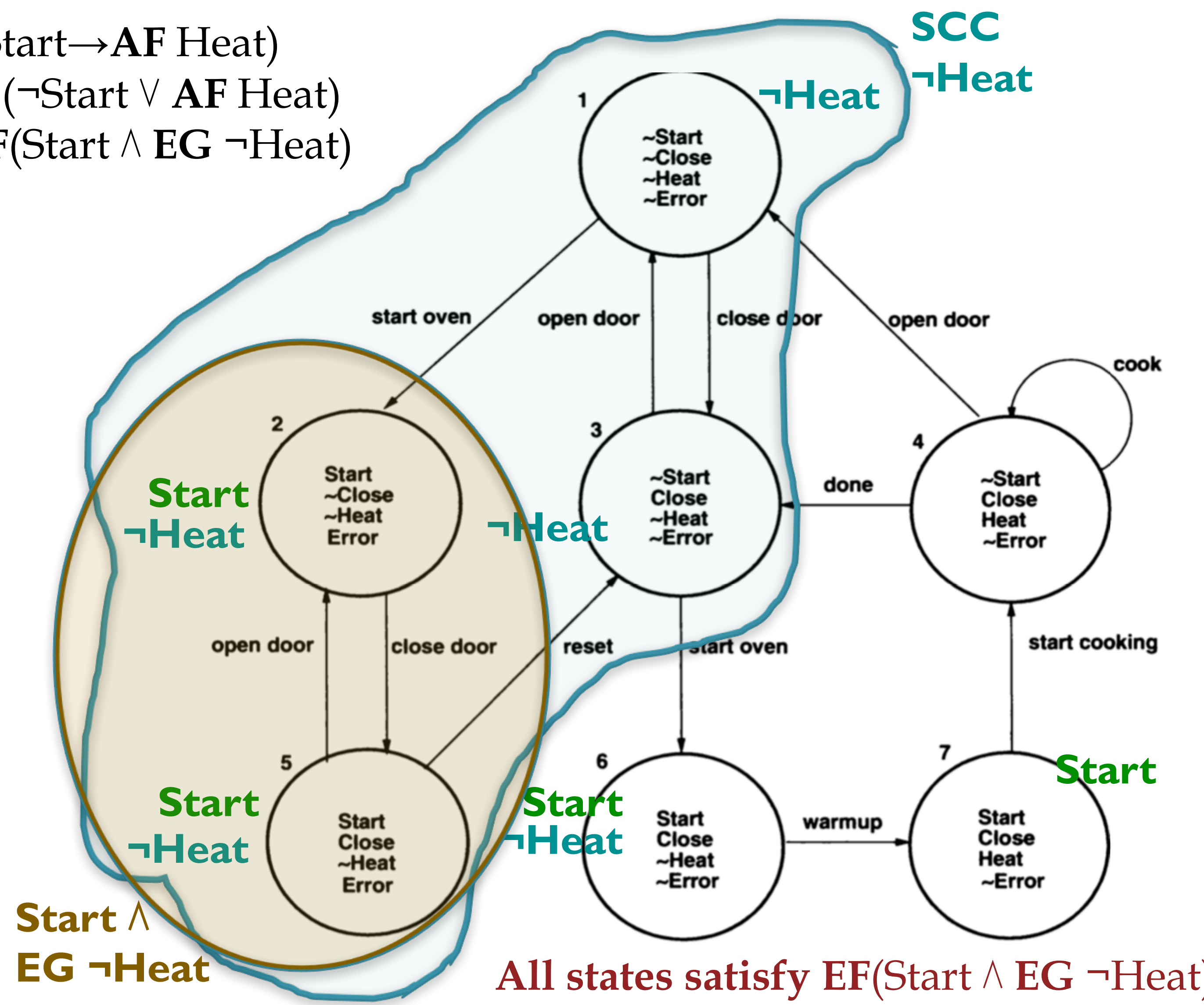
$$O(\,(|S|+|R|) \cdot\ |f|\,.$$

CTL model checking 5

```
procedure CheckEG(f₁)
    S' := { s | f₁ ∈ label(s) };
    SCC := { C | C is a nontrivial SCC of S' };
    T := ⋃_{C∈SCC} { s | s ∈ C };
    for all s ∈ T do label(s) := label(s) ∪ { EG f₁ };
    while T ≠ ∅ do
        choose s ∈ T;
        T := T \ {s};
        for all t such that t ∈ S' and R(t, s) do
            if EG f₁ ∉ label(t) then
                label(t) := label(t) ∪ { EG f₁ };
                T := T ∪ {t};
            end if;
        end for all;
    end while;
end procedure
```

Example: microwave oven

AG(Start→**AF** Heat)
≡ **AG**(¬Start ∨ **AF** Heat)
≡ ¬**EF**(Start ∧ **EG** ¬Heat)



All states satisfy EF(Start ∧ **EG** ¬Heat)

Detour: Hamiltonian path in CTL

Taken a graph G=(V, E), we define a Kripke structure $\mathcal{M}$ =(S, R, L) where:

- $S = E \cup \{b\}$ **b is needed to make R total**
- $R = E \cup \{v \rightarrow b \mid v \in E\}$
- $L(v) = \{v\}$

We define $f = \bigvee_{(i_1,\ldots,i_n) \text{ permutation of } (1,\ldots,n)} g(v_{i_1}, \ldots, v_{i_n})$
and g inductively as follows:

$$g(v_i) = v_i$$
$$g(v_{i_1}, \ldots, v_{i_n}) = v_{i_1} \wedge \mathbf{E}\,\mathbf{X}\, g(v_{i_2}, \ldots, v_{i_n}) \text{ if } n>1$$

It is easy to see that **$g(v_{i_1}, \ldots, v_{i_n})$ holds** if and only if **$v_{i_1}, \ldots, v_{i_n}$ is a Hamiltonian path** in G.

Therefore, $\mathcal{M} \vDash f$ if and only if G has a Hamiltonian path.

Obviously, **this reduction is not polynomial**!

Lesson 3c:

CTL Model Checking*

Idea of CTL Model Checking*

Idea: use CTL and LTL model checking procedures on sub-formulas.

Substitute any maximal state sub-formulas with fresh atomic propositions.Like CTL algorithm, the CTL* algorithm works in stages.

Level 0: atomic propositions
Level *i*+1: all state sub-formulas g such that all state sub-formulas of g are of level i or less and g is not contained in any lower level.

Example: $\mathbf{AG}((\neg \text{Close} \wedge \text{Start}) \rightarrow \mathbf{A}\ (\mathbf{G}\ \neg \text{Heat} \vee \mathbf{F}\ \neg \text{Error}))$
Only **E** quantifier: $\neg \mathbf{EF}((\neg \text{Close} \wedge \text{Start} \wedge \mathbf{E}\ (\mathbf{F}\ \text{Heat} \wedge \mathbf{G}\ \text{Error}))$

Level 0: Close, Start, Heat, Error
Level 1: $\neg$Close, $\mathbf{E}\ (\mathbf{F}\ \text{Heat} \wedge \mathbf{G}\ \text{Error})$
Level 2: $\mathbf{EF}((\neg \text{Close} \wedge \text{Start} \wedge \mathbf{E}\ (\mathbf{F}\ \text{Heat} \wedge \mathbf{G}\ \text{Error}))$
Level 3: $\neg \mathbf{EF}((\neg \text{Close} \wedge \text{Start} \wedge \mathbf{E}\ (\mathbf{F}\ \text{Heat} \wedge \mathbf{G}\ \text{Error}))$

CTL* Model Checking: algorithm

Algorithm 27 CTL* model checking algorithm (basic idea)

Input: finite transition system TS with initial states I, and CTL* formula Φ
Output: $I \subseteq Sat(\Phi)$

for all $i \leqslant |\,\Phi\,|$ **do**
  for all $\Psi \in Sub(\Phi)$ with $|\,\Psi\,| = i$ **do**
    switch(Ψ):

true	:	$Sat(\Psi) := S$;
a	:	$Sat(\Psi) := \{\, s \in S \mid a \in L(s) \,\}$;
$a_1 \wedge a_2$	:	$Sat(\Psi) := Sat(a_1) \cap Sat(a_2)$;
$\neg a$	:	$Sat(\Psi) := S \setminus Sat(a)$;
$\exists\varphi$	:	determine $Sat_{LTL}(\neg\varphi)$ by means of an LTL model-checker;
	:	$Sat(\Psi) := S \setminus Sat_{LTL}(\neg\varphi)$

    end switch
    $AP := AP \cup \{\, a_\Psi \,\}$;   (* introduce fresh atomic proposition *)
    replace Ψ with a_Ψ
    forall $s \in Sat(\Psi)$ **do** $L(s) := L(s) \cup \{\, a_\Psi \,\}$; **od**
  od
od
return $I \subseteq Sat(\Phi)$

CTL: example and complexity*

Example: ¬**EF**((¬Close ∧ Start ∧ **E** (**F** Heat ∧ **G** Error))
Level 1: The level 1 formula ¬Close is added to $L(1)$ and $L(2)$
E (**F** Heat ∧ **G** Error) is pure LTL, but there is no state satisfying this formula.

Level 2: **E** (**F** Heat ∧ **G** Error) is replaced by a fresh atomic proposition a. LTL-model checking is then applied to the formula **EF**((¬Close ∧ Start ∧ a), that is unsatisfiable, so all states are labeled with ¬**EF**((¬Close ∧ Start ∧ **E** (**F** Heat ∧ **G** Error)).

Theorem: There exists a CTL* model checking algorithm with complexity $O(|\mathcal{M}|2^{|f|})$

Theorem: CTL* model checking is PSPACE-complete.