

Temporal Logic Model Checking

(Based on [Clarke et al. 1999])

Yih-Kuen Tsay

Dept. of Information Management
National Taiwan University

About Temporal Logic

- 🌐 Temporal logic is a formalism for describing **temporal ordering** (or dependency) between occurrences of “events” (represented by propositions).
- 🌐 It provides such expressive features by introducing **temporal/modal operators** into classic logic.
- 🌐 These temporal operators usually do not explicitly mention time points.
- 🌐 There are two principal views of the structure of time:
 - ☀️ **linear-time**: occurrences of events form a **sequence**
 - ☀️ **branching-time**: occurrences of events form a **tree**

Outline

- 🌐 Temporal Logics
 - ☀ CTL* (generalized Computation Tree Logic)
 - ☀ CTL (Computation Tree Logic; subset of CTL*)
 - ☀ LTL (Linear Temporal Logic; subset of CTL*)
- 🌐 Fairness
- 🌐 Algorithmic Temporal Logic Verification
 - ☀ CTL Model Checking
 - ☀ LTL Model Checking
 - ☀ CTL* Model Checking

- 🌐 CTL* formulae describe properties of a **computation tree** (generated from a Kripke structure).
- 🌐 They are composed of *path quantifiers* and *temporal operators*.
- 🌐 Path quantifiers:
 - ☀ **E** (for some path)
 - ☀ **A** (for all paths)
- 🌐 Temporal operators:
 - ☀ **X** (next)
 - ☀ **F** (eventually or sometime in the future)
 - ☀ **G** (always or globally)
 - ☀ **U** (until)
 - ☀ **R** (release)

Syntax of CTL*

- Let AP be a set of atomic propositions.
- The syntax of **state formulae**:
 - If $p \in AP$, then p is a state formula.
 - If f_1 and f_2 are state formulae, then so are $\neg f_1$, $f_1 \vee f_2$ and $f_1 \wedge f_2$.
 - If g is a path formula, then **E** g and **A** g are state formulae.
- The syntax of **path formulae**:
 - If f is a state formula, then f is also a path formula.
 - If g_1 and g_2 are path formulae, then so are $\neg g_1$, $g_1 \vee g_2$, $g_1 \wedge g_2$, **X** g_1 , **F** g_1 , **G** g_1 , g_1 **U** g_2 , and g_1 **R** g_2 .
- CTL* is the set of state formulae generated by the above rules.

Example CTL* Formulae

- 🌐 Formula: **AG**(*Req* \rightarrow **AF***Ack*).
Intended meaning: every request will eventually be granted.
- 🌐 Formula: **AG**(**EF***Restart*).
Intended meaning: it is always possible at any time to get to the *Restart* state.

Kripke Structures

- Let AP be a set of atomic propositions.
- A *Kripke structure* M over AP is a tuple $\langle S, S_0, R, L \rangle$:
 - S is a finite set of states,
 - $S_0 \subseteq S$ is the set of initial states,
 - $R \subseteq S \times S$ is a total transition relation, and
 - $L : S \rightarrow 2^{AP}$ is a function labeling each state with a subset of propositions (which are true in that state).
- A *computation* or *path* π of M from a state s is an infinite sequence s_0, s_1, s_2, \dots of states such that $s_0 = s$ and $(s_i, s_{i+1}) \in R$, for all $i \geq 0$.
- In the sequel, π^i denotes the *suffix* of π starting at s_i .

Semantics of CTL*

- 🌐 When f is a **state** formula, $M, s \models f$ means that f holds at state s in the Kripke structure M .
- 🌐 When f is a **path** formula, $M, \pi \models f$ means that f holds along the path π in the Kripke structure M .
- 🌐 Assuming that f_1 and f_2 are state formulae and g_1 and g_2 are path formulae, the semantics of CTL* is as follows:
 - ☀️ $M, s \models p \iff p \in L(s)$
 - ☀️ $M, s \models \neg f_1 \iff M, s \not\models f_1$
 - ☀️ $M, s \models f_1 \vee f_2 \iff M, s \models f_1 \text{ or } M, s \models f_2$
 - ☀️ $M, s \models f_1 \wedge f_2 \iff M, s \models f_1 \text{ and } M, s \models f_2$
 - ☀️ $M, s \models \mathbf{E}g_1 \iff \text{for some path } \pi \text{ from } s, M, \pi \models g_1$
 - ☀️ $M, s \models \mathbf{A}g_1 \iff \text{for every path } \pi \text{ from } s, M, \pi \models g_1$

Semantics of CTL* (cont.)

🌐 The semantics of CTL* (cont.):

- ☀ $M, \pi \models f_1 \iff s$ is the first state of π and $M, s \models f_1$
- ☀ $M, \pi \models \neg g_1 \iff M, \pi \not\models g_1$
- ☀ $M, \pi \models g_1 \vee g_2 \iff M, \pi \models g_1$ or $M, \pi \models g_2$
- ☀ $M, \pi \models g_1 \wedge g_2 \iff M, \pi \models g_1$ and $M, \pi \models g_2$
- ☀ $M, \pi \models \mathbf{X}g_1 \iff M, \pi^1 \models g_1$
- ☀ $M, \pi \models \mathbf{F}g_1 \iff$ for some $k \geq 0$, $M, \pi^k \models g_1$
- ☀ $M, \pi \models \mathbf{G}g_1 \iff$ for all $i \geq 0$, $M, \pi^i \models g_1$
- ☀ $M, \pi \models g_1 \mathbf{U} g_2 \iff$ for some $k \geq 0$, $M, \pi^k \models g_2$ and, for all $0 \leq j < k$, $M, \pi^j \models g_1$
(g_1 remains *true* until g_2 becomes *true*, which eventually happens.)
- ☀ $M, \pi \models g_1 \mathbf{R} g_2 \iff$ for all $j \geq 0$, if for every $i < j$, $M, \pi^i \not\models g_1$, then $M, \pi^j \models g_2$
(Only after g_1 becomes *true*, g_2 may become *false*.)

Minimalistic CTL*

- 🌍 The operators \vee , \neg , **X**, **U**, and **E** are sufficient to express any other CTL* formula (in an equivalent way).
- 🌍 In particular,
 - ☀ $\mathbf{F}f = \text{true } \mathbf{U} f$
 - ☀ $\mathbf{G}f = \neg \mathbf{F} \neg f$
 - ☀ $f \mathbf{R} g = \neg(\neg f \mathbf{U} \neg g)$
 - ☀ $\mathbf{A}f = \neg \mathbf{E} \neg f$
- 🌍 $\neg(\neg f \mathbf{U} \neg g)$ says that it is not the case that in some state g becomes *false* and until then f has never been *true*.
- 🌍 This is the same as saying that *only after* f becomes *true*, g may become *false* (or f “releases” g), namely $f \mathbf{R} g$.

CTL and LTL

- 🌐 CTL and LTL are restricted subsets of CTL*.
- 🌐 CTL is a branching-time logic, while LTL is linear-time.
- 🌐 In CTL, each temporal operator **X**, **F**, **G**, **U**, or **R** must be immediately preceded by a path quantifier **E** or **A**.
- 🌐 The syntax of path formulae in CTL is more restricted:
 - ☀ If f_1 and f_2 are state formulae, then $\mathbf{X}f_1$, $\mathbf{F}f_1$, $\mathbf{G}f_1$, $f_1 \mathbf{U} f_2$, and $f_1 \mathbf{R} f_2$ are path formulae.
- 🌐 The syntax of state formulae remains the same:
 - ☀ If $p \in AP$, then p is a state formula.
 - ☀ If f_1 and f_2 are state formulae, then so are $\neg f_1$, $f_1 \vee f_2$ and $f_1 \wedge f_2$.
 - ☀ If g is a path formula, then $\mathbf{E}g$ and $\mathbf{A}g$ are state formulae.

CTL and LTL (cont.)

- 🌐 LTL consists of formulae that have the form $\mathbf{A}f$, where f is a path formula in which **atomic propositions are the only permitted state formulae**.
- 🌐 The syntax of path formulae in LTL is as follows:
 - ☀ If $p \in AP$, then p is a path formula.
 - ☀ If g_1 and g_2 are path formulae, then so are $\neg g_1$, $g_1 \vee g_2$, $g_1 \wedge g_2$, $\mathbf{X}g_1$, $\mathbf{F}g_1$, $\mathbf{G}g_1$, $g_1 \mathbf{U} g_2$, and $g_1 \mathbf{R} g_2$.

Expressive Powers

- 🌐 CTL, LTL, and CTL* have distinct expressive powers.
- 🌐 Some discriminating examples:
 - ☀ **A(FGp)** in LTL, not expressible in CTL.
 - ☀ **AG(EFp)** in CTL, not expressible in LTL.
 - ☀ Both **A(FGp)** and **AG(EFp)** are expressible in CTL*.
- 🌐 So, CTL* is strictly more expressive than CTL and LTL, the two of which are incomparable.

Fair Kripke Structures

- 🌐 A fair Kripke structure is a 4-tuple $M = (S, R, L, F)$, where S , L , and R are defined as before and $F \subseteq 2^S$ is a set of fairness constraints. (Generalized Büchi acceptance conditions)
- 🌐 Let $\pi = s_0, s_1, \dots$ be a path in M .
- 🌐 Define $\text{inf}(\pi) = \{s \mid s = s_i \text{ for infinitely many } i\}$.
- 🌐 We say that π is *fair* iff, for every $P \in F$, $\text{inf}(\pi) \cap P \neq \emptyset$.










Fair Semantics

- 🌐 We write $M, s \models_F f$ to indicate that the state formula f is true in state s of the fair Kripke structure M .
- 🌐 $M, \pi \models_F g$ indicates that the path formula g is true along the path π in M .
- 🌐 Only the following semantic rules are different from the original ones:
 - ☀️ $M, s \models_F p \iff$ there exists a fair path starting from s and $p \in L(s)$.
 - ☀️ $M, s \models_F \mathbf{E}g_1 \iff$ there exists a fair path π starting from s s.t. $M, \pi \models_F g_1$.
 - ☀️ $M, s \models_F \mathbf{A}g_1 \iff$ for every fair path π starting from s , $M, \pi \models_F g_1$.

CTL Model Checking

- Let $M = (S, R, L)$ be a Kripke structure.
- We want to determine which states in S satisfy the CTL formula f .
- The algorithm will operate by labelling each state s with the set $label(s)$ of sub-formulae of f which are true in s .
 - Initially, $label(s)$ is just $L(s)$.
 - During the i -th stage, sub-formulae with $i - 1$ nested CTL operators are processed.
 - When a sub-formula is processed, it is added to the labelling of each state in which it is true.
 - Once the algorithm terminates, we will have that $M, s \models f$ iff $f \in label(s)$.

Handling CTL Operators

-  There are ten basic CTL temporal operators: **AX** and **EX**, **AF** and **EF**, **AG** and **EG**, **AU** and **EU**, and **AR** and **ER**.
-  All these operators can be expressed in terms of **EX**, **EU**, and **EG**:
 -  $\mathbf{AX}f = \neg\mathbf{EX}\neg f$
 -  $\mathbf{EF}f = \mathbf{E}[true \mathbf{U} f]$
 -  $\mathbf{AF}f = \neg\mathbf{EG}\neg f$
 -  $\mathbf{AG}f = \neg\mathbf{EF}\neg f$
 -  $\mathbf{A}[f \mathbf{U} g] = \neg\mathbf{E}[\neg g \mathbf{U} (\neg f \wedge \neg g)] \wedge \neg\mathbf{EG}\neg g$
 (This case is less obvious and will be proven next.)
 -  $\mathbf{A}[f \mathbf{R} g] = \neg\mathbf{E}[\neg f \mathbf{U} \neg g]$ (from $f \mathbf{R} g = \neg(\neg f \mathbf{U} \neg g)$)
 -  $\mathbf{E}[f \mathbf{R} g] = \neg\mathbf{A}[\neg f \mathbf{U} \neg g]$

The Case of AU

🌐 To see why $\mathbf{A}[f \mathbf{U} g] = \neg \mathbf{E}[\neg g \mathbf{U} (\neg f \wedge \neg g)] \wedge \neg \mathbf{EG}\neg g$, let us introduce yet another temporal operator \mathbf{W} (wait-for).

🌐 Let $f \mathbf{W} g = f \mathbf{U} g \vee \mathbf{G}f$.

🌐 It can be shown that

$$\odot f \mathbf{U} g = (f \mathbf{W} g) \wedge \mathbf{F}g$$

$$\odot \neg(f \mathbf{W} g) = \neg g \mathbf{U} (\neg f \wedge \neg g).$$

🌐 Proof of the rewriting for $\mathbf{A}[f \mathbf{U} g]$:

$$\begin{aligned} & \mathbf{A}[f \mathbf{U} g] \\ &= \neg \mathbf{E}\neg(f \mathbf{U} g) \\ &= \neg \mathbf{E}\neg((f \mathbf{W} g) \wedge \mathbf{F}g) \\ &= \neg \mathbf{E}(\neg(f \mathbf{W} g) \vee \neg \mathbf{F}g) \\ &= \neg(\mathbf{E}\neg(f \mathbf{W} g) \vee \mathbf{E}\neg \mathbf{F}g) \\ &= \neg(\mathbf{E}[\neg g \mathbf{U} (\neg f \wedge \neg g)] \vee \mathbf{EG}\neg g) \\ &= \neg \mathbf{E}[\neg g \mathbf{U} (\neg f \wedge \neg g)] \wedge \neg \mathbf{EG}\neg g \end{aligned}$$

CTL Model Checking: AP, \neg , \vee , EX

So, for CTL model checking, it suffices to handle the following six cases: *atomic proposition*, \neg , \vee , **EX**, **EU**, and **EG**.

- 🌐 Atomic propositions are handled at the beginning of the algorithm (by the initial setting $label(s) = L(s)$).
- 🌐 For $\neg f$, we label those states that are not labelled by f .
- 🌐 For $f_1 \vee f_2$, we label any state that is labelled either by f_1 or by f_2 .
- 🌐 For **EX** f , we label every state that has some successor labelled by f .

CTL Model Checking: EU

- 🌐 To handle formulae of the form $\mathbf{E}[f_1 \mathbf{U} f_2]$, we follow these steps:
 - ☀️ Find all states that are labelled with f_2 .
 - ☀️ Work backward using the converse of the transition relation R and find all states that can be reached by a path in which *each state* is labelled with f_1 .
 - ☀️ Label all such states by $\mathbf{E}[f_1 \mathbf{U} f_2]$.
- 🌐 This requires time $O(|S| + |R|)$.

CTL Model Checking: EU (cont.)

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

- 🌐 To handle formulae of the form **EGf**, we need the following lemma:

Let $M' = (S', R', L')$, where $S' = \{s \in S \mid M, s \models f\}$.

$M, s \models \mathbf{EG}f$ iff the following two conditions hold:

1. $s \in S'$.
 2. There exists a path in M' that leads from s to some node t in a *nontrivial* strongly connected component (SCC) C of the graph (S', R') .
- 🌐 Note: an SCC is nontrivial if either it contains at least two nodes or it contains only one node with a self loop.

- 🌐 With the lemma, we can handle **EG** f by the following steps:
1. Construct the restricted Kripke structure M' .
 2. Partition the (S', R') into SCCs. (Complexity: $O(|S'| + |R'|)$).
 3. Find those states that belong to nontrivial components.
 4. Work backward using the converse of R' and find all of those states that can be reached by a path in which each state is labelled with f . (Complexity: $O(|S| + |R|)$)

CTL Model Checking: EG (cont.)

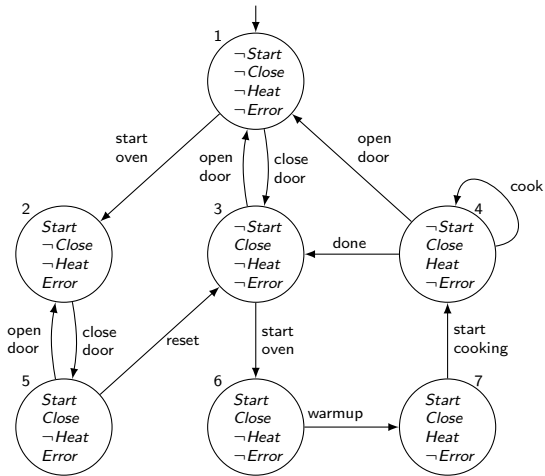
```
procedure CheckEG(f)  
   $S' := \{s \mid f \in \text{label}(s)\};$   
   $SCC := \{C \mid C \text{ is a nontrivial SCC of } S'\};$   
   $T := \bigcup_{C \in SCC} \{s \mid s \in C\};$   
  for all  $s \in T$  do  $\text{label}(s) := \text{label}(s) \cup \{\mathbf{EG}f\};$   
  while  $T \neq \emptyset$  do  
    choose  $s \in T;$   
     $T := T \setminus \{s\};$   
    for all  $t$  s.t.  $t \in S'$  and  $R(t, s)$  do  
      if  $\mathbf{EG}f \notin \text{label}(t)$  and  $f \in \text{label}(t)$  then  
         $\text{label}(t) := \text{label}(t) \cup \{\mathbf{EG}f\};$   
         $T := T \cup \{t\};$   
      end if;  
    end for all;  
  end while; end procedure;
```


CTL Model Checking (cont.)

- 🌐 We successively apply the state-labelling algorithm to the sub-formulae of f , starting with the shortest, most deeply nested, and work outward to include the whole formula.
- 🌐 By proceeding in this manner, we guarantee that whenever we process a sub-formula of f all its sub-formulae have already been processed.
- 🌐 There are at most $|f|$ sub-formulae, and each formula takes at most $O(|S| + |R|)$ time.
- 🌐 The complexity of this algorithm is $O(|f| \cdot (|S| + |R|))$.

An Example

We want to check CTL formula
 $\mathbf{AG}(Start \rightarrow \mathbf{AF}Heat)$, or
 $\neg \mathbf{E}[true \mathbf{U} (Start \wedge \mathbf{EG}\neg Heat)]$.



An Example (cont.)

Spec.: $\neg E[true \ U (Start \wedge EG\neg Heat)]$

$S(Start) = \{2, 5, 6, 7\}$

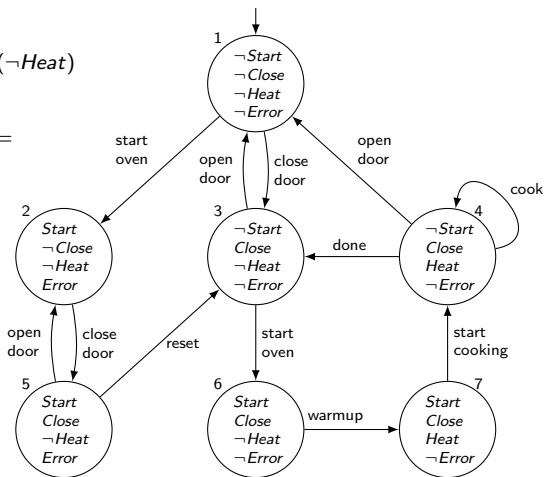
$S(\neg Heat) = \{1, 2, 3, 5, 6\}$

Find SCC $\{\{1, 2, 3, 5\}\}$ in $S' = S(\neg Heat)$

$S(EG\neg Heat) = \{1, 2, 3, 5\}$

$S(Start \wedge EG\neg Heat) = \{2, 5\}$

$S(E[true \ U (Start \wedge EG\neg Heat)]) = \{1, 2, 3, 4, 5, 6, 7\}$



Fairness Constraints

- Let $M = (S, R, L, F)$ be a fair Kripke structure.
- Let $F = \{P_1, \dots, P_k\}$.
- We say that a SCC C is *fair* w.r.t F iff for each $P_i \in F$, there is a state $t_i \in (C \cap P_i)$.
- To handle formulae of the form **EGf** in a fair kripke structure, we need the following lemma:

Let $M' = (S', R', L', F')$, where $S' = \{s \in S \mid M, s \models_F f\}$.
 $M, s \models_F EGf$ iff the following two conditions holds:

- $s \in S'$.
- There exists a path in S' that leads from s to some node t in a nontrivial *fair* strongly connected component of the graph (S', R') .

Fairness Constraints

- 🌐 We can create a *CheckFairEG* algorithm which is very similar to the *CheckEG* algorithm based on this lemma.
- 🌐 The complexity of *CheckFairEG* is $O((|S| + |R|) \cdot |F|)$, since we have to check which SCC is fair.
- 🌐 To check other CTL formulae, we introduce another proposition *fair* and stipulate that

$$M, s \models \textit{fair} \text{ iff } M, s \models_F \mathbf{EG} \textit{true}.$$

- 🌐 $M, s \models_F p$, for some $p \in AP$, we check $M, s \models p \wedge \textit{fair}$.
- 🌐 $M, s \models_F \mathbf{EX}f$, we check $M, s \models \mathbf{EX}(f \wedge \textit{fair})$.
- 🌐 $M, s \models_F \mathbf{E}[f_1 \mathbf{U} f_2]$, we check $M, s \models \mathbf{E}[f_1 \mathbf{U} (f_2 \wedge \textit{fair})]$.
- 🌐 Overall complexity: $O(|f| \cdot (|S| + |R|) \cdot |F|)$.

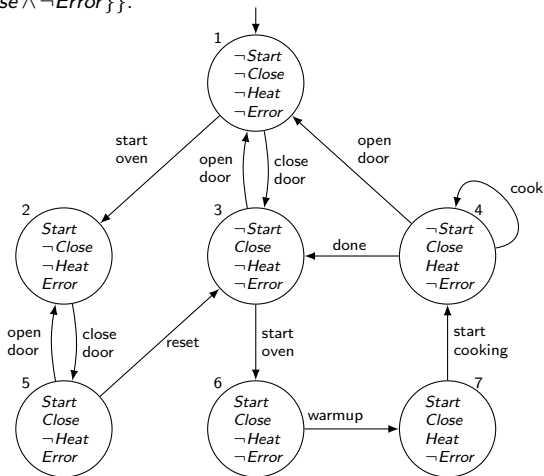
An Example

Assume $F = \{ \{s \mid s \models \text{Start} \wedge \text{Close} \wedge \neg \text{Error} \} \}$.

We want to check CTL formula

$\mathbf{AG}(\text{Start} \rightarrow \mathbf{AF}\text{Heat})$, or

$\neg \mathbf{E}[\text{true} \mathbf{U} (\text{Start} \wedge \mathbf{EG} \neg \text{Heat})]$.



An Example (cont.)

Spec.: $\neg E[true \ U (Start \wedge EG\neg Heat)]$

$S(Start) = \{2, 5, 6, 7\}$

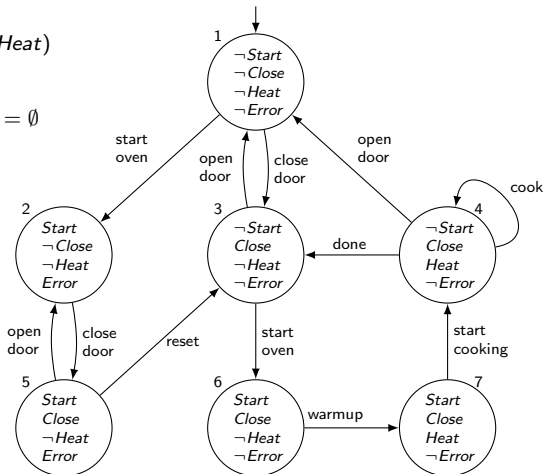
$S(\neg Heat) = \{1, 2, 3, 5, 6\}$

There is no fair SCC in $S' = S(\neg Heat)$

$S(EG\neg Heat) = \emptyset$

$S(Start \wedge EG\neg Heat) = \emptyset$

$S(E[true \ U (Start \wedge EG\neg Heat)]) = \emptyset$



The LTL Model Checking Problem

- Let $M = (S, R, L)$ be a Kripke structure with $s \in S$.
- Let $\mathbf{A}g$ be an LTL formula (so, g is a restricted path formula).
- We want to check if $M, s \models \mathbf{A}g$.
- $M, s \models \mathbf{A}g$ iff $M, s \models \neg \mathbf{E} \neg g$.
- Therefore, it suffices to be able to check $M, s \models \mathbf{E}f$, where f is a restricted path formula.

Complexity of LTL Model Checking

- 🌐 The problem is PSPACE-complete.
- 🌐 We can more easily show this problem to be NP-hard by a reduction from the Hamiltonian path problem.
- 🌐 Consider a directed graph $G = (V, A)$ where $V = \{v_1, v_2, \dots, v_n\}$.
- 🌐 Determining whether G has a directed Hamiltonian path is reducible to the problem of determining whether $M, s \models f$, where
 - ☀ M is a finite Kripke structure (constructed from G),
 - ☀ s is a state in M , and
 - ☀ f is the formula (using atomic propositions p_1, \dots, p_n):

$$\mathbf{E}[\mathbf{F}p_1 \wedge \dots \wedge \mathbf{F}p_n \wedge \mathbf{G}(p_1 \rightarrow \mathbf{XG}\neg p_1) \wedge \dots \wedge \mathbf{G}(p_n \rightarrow \mathbf{XG}\neg p_n)].$$

Complexity of LTL Model Checking (cont.)

- 🌐 The Kripke structure $M = (U, B, L)$ is obtained from $G = (V, A)$ as follows:
 - ☀️ $U = V \cup \{u_1, u_2\}$ where $u_1, u_2 \notin V$.
 - ☀️ $B = A \cup \{(u_1, v_i) \mid v_i \in V\} \cup \{(v_i, u_2) \mid v_i \in V\} \cup \{(u_2, u_2)\}$.
 - ☀️ L is an assignment of propositions to states s.t.:
 - 👤 p_i is true in v_i for $1 \leq i \leq n$,
 - 👤 p_i is false in v_j for $1 \leq i, j \leq n, i \neq j$, and
 - 👤 p_i is false in u_1, u_2 for $1 \leq i \leq n$.
- 🌐 Let s be u_1 .
- 🌐 $M, u_1 \models f$ iff there is a directed infinite path in M starting at u_1 that goes through every $v_i \in V$ exactly once and ends in the self loop at u_2 .

Here we introduce an algorithm by Lichtenstein and Pnueli.







- 🌐 The algorithm is exponential in the length of the formula, but linear in the size of the state graph.
- 🌐 It involves an implicit **tableau construction**.
- 🌐 A tableau is a graph derived from the formula from which a model for the formula can be extracted iff the formula is satisfiable.
- 🌐 To check whether M satisfies f , the algorithm composes the tableau and the Kripke structure and determines whether there exists a computation of the structure that is a path in the tableau.

Closure






- Like before, we need only deal with **X** and **U**.
- The closure $CL(f)$ of f contains formulae whose truth values can influence the truth value of f .
- It is the smallest set containing f and satisfying:
 - $\neg f_1 \in CL(f)$ iff $f_1 \in CL(f)$,
 - if $f_1 \wedge f_2 \in CL(f)$, then $f_1, f_2 \in CL(f)$,
 - if $\mathbf{X}f_1 \in CL(f)$, then $f_1 \in CL(f)$,
 - if $\neg\mathbf{X}f_1 \in CL(f)$, then $\mathbf{X}\neg f_1 \in CL(f)$,
 - if $f_1 \mathbf{U} f_2 \in CL(f)$, then $f_1, f_2, \mathbf{X}[f_1 \mathbf{U} f_2] \in CL(f)$.

- 🌐 An *atom* is a pair $A = (s_A, K_A)$ with $s_A \in S$ and $K_A \subseteq CL(f) \cup AP$ s.t.:
- ☀ for each proposition $p \in AP$, $p \in K_A$ iff $p \in L(s_A)$,
 - ☀ for every $f_1 \in CL(f)$, $f_1 \in K_A$ iff $\neg f_1 \notin K_A$,
 - ☀ for every $f_1 \vee f_2 \in CL(f)$, $f_1 \vee f_2 \in K_A$ iff $f_1 \in K_A$ or $f_2 \in K_A$,
 - ☀ for every $\neg \mathbf{X}f_1 \in CL(f)$, $\neg \mathbf{X}f_1 \in K_A$ iff $\mathbf{X}\neg f_1 \in K_A$,
 - ☀ for every $f_1 \mathbf{U} f_2 \in CL(f)$, $f_1 \mathbf{U} f_2 \in K_A$ iff $f_2 \in K_A$ or $f_1, \mathbf{X}[f_1 \mathbf{U} f_2] \in K_A$.
- 🌐 Intuitively, an atom (s_A, K_A) is defined so that K_A is a maximal consistent set of formulae that are also consistent with the labelling of s_A .

Behavior Graph and Self-Fulfilling SCC

-  A graph G is constructed with the set of atoms as the set of vertices.
-  (A, B) is an edge of G iff
 -  $(s_A, s_B) \in R$ and
 -  for every formula $\mathbf{X}f_1 \in CL(f)$, $\mathbf{X}f_1 \in K_A$ iff $f_1 \in K_B$.
-  A nontrivial SCC C of the graph G is said to be *self-fulfilling* iff for every atom A in C and for every $f_1 \mathbf{U} f_2 \in K_A$ there exists an atom B in C s.t. $f_2 \in K_B$.
-  **Lemma:** $M, s \models \mathbf{E}f$ iff there exists an atom (s, K) in G s.t. $f \in K$ and there is a path in G from (s, K) to a self-fulfilling SCC.

Sketch of the Correctness Proof

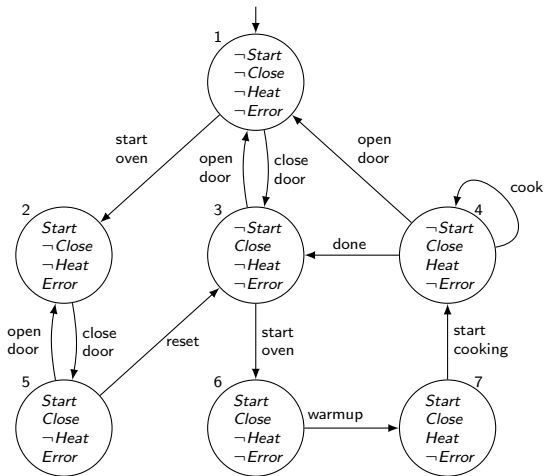
-  A path ρ in G (generated from M and f) is an *eventuality sequence* if $f_1 \mathbf{U} f_2 \in K_A$ for some atom A on ρ , then there exists an atom B , reachable from A along π , such that $f_2 \in K_B$.
-  Claim: $M, s \models \mathbf{E}f$ iff there exists an eventuality sequence starting from (s, K) such that $f \in K$.
 -  (\Leftarrow) If $\pi = s_0(=s), s_1, s_2, \dots$ corresponds to an eventuality sequence $(s, K) = (s_0, K_0), (s_1, K_1), \dots$, then for every $g \in CL(f)$ and every $i \geq 0$, $\pi^i \models g$ iff $g \in K_i$.
 -  (\Rightarrow) For a path $\pi = s_0(=s), s_1, s_2, \dots$ such that $M, \pi \models f$, define $K_i = \{g \mid g \in CL(f) \text{ and } \pi^i \models g\}$, then $(s_0, K_0), (s_1, K_1), \dots$ is an eventuality sequence.
-  Claim: there exists an eventuality sequence starting from (s, K) iff there is a path in G from (s, K) to a self-fulfilling SCC.

The LTL Model Checking Algorithm

- 🌐 Given a Kripke structure $M = (S, R, L)$, we want to check if $M, s \models \mathbf{E}f$, where f is a restricted path formula.
 - ☀ Construct the behavior graph $G = (V, E)$.
 - ☀ Find initial atom set $A = \{(s, K) \mid (s, K) \in V \wedge f \in K\}$.
 - ☀ Consider nontrivial self-fulfilling SCCs, traverse backward using the converse of E and mark all reachable states.
 - ☀ If any state in A is marked, $M, s \models \mathbf{E}f$ is true.
- 🌐 Time complexity: $O((|S| + |R|) \cdot 2^{O(|f|)})$.
- 🌐 For a fair Kripke structure $M' = (S', R', L', F')$, we should check if there exists any self-fulfilling and fair SCC.

An Example

We want to check LTL formula
 $\mathbf{A}[\neg \text{Heat } \mathbf{U} \text{ Close}]$, or
 $\neg \mathbf{E}\neg[\neg \text{Heat } \mathbf{U} \text{ Close}]$.



An Example (cont.)

- Let f denote $\neg Heat \mathbf{U} Close$.
- $CL(\neg f) = \{\neg f, f, \mathbf{X}f, \neg\mathbf{X}f, \mathbf{X}\neg f, Heat, \neg Heat, Close, \neg Close\}$.
- $\neg Close$ and $\neg Heat$ in states 1 and 2, so the possible “K” includes
 $\{\neg Close, \neg Heat, f, \mathbf{X}f\}, \{\neg Close, \neg Heat, \neg f, \mathbf{X}\neg f, \neg\mathbf{X}f\}$.
- $Close$ and $\neg Heat$ in states 3, 5 and 6, so the possible “K” includes $\{Close, \neg Heat, f, \mathbf{X}f\}, \{Close, \neg Heat, f, \mathbf{X}\neg f, \neg\mathbf{X}f\}$.
- $Close$ and $Heat$ in states 4 and 7, so the possible “K” includes $\{Close, Heat, f, \mathbf{X}f\}, \{Close, Heat, f, \mathbf{X}\neg f, \neg\mathbf{X}f\}$.

We can construct atoms using the states and the corresponding “K” and then build a graph based on those atoms.

Overview of CTL* Model Checking

- 🌐 We will study an algorithm developed by Clarke, Emerson, and Sistla.
- 🌐 The basic idea is to integrate the state labeling technique from CTL model checking into LTL model checking.
- 🌐 The algorithm for LTL handles formula of the form $\mathbf{E}f$ where f is a restricted path formula.
- 🌐 The algorithm can be extended to handle formulae in which f contains arbitrary state sub-formulae.

Handling CTL* Operators

Again, the operators \neg , \vee , **X**, **U**, and **E** are sufficient to express any other CTL* formula.

$$\text{🌐 } f \wedge g \equiv \neg(\neg f \vee \neg g)$$

$$\text{🌐 } \mathbf{F}f \equiv \text{true } \mathbf{U} f$$

$$\text{🌐 } \mathbf{G}f \equiv \neg \mathbf{F} \neg f$$

$$\text{🌐 } f \mathbf{R} g \equiv \neg(\neg f \mathbf{U} \neg g)$$

$$\text{🌐 } \mathbf{A}f \equiv \neg \mathbf{E} \neg f$$

One Stage in CTL* Model Checking

- Let $\mathbf{E}f'$ be an “inner most” formula with \mathbf{E} .
- Assuming that the state sub-formulae of f' have already been processed and that state labels have been updated accordingly, proceed as follows:
 - If $\mathbf{E}f'$ is in CTL, then apply the CTL algorithm.
 - Otherwise, f' is a LTL path formula, then apply the LTL model checking algorithm.
 - In both cases, the formula is added to the labels of all states that satisfy it.
- If $\mathbf{E}f'$ is a sub-formula of a more complex CTL* formula, then the procedure is repeated with $\mathbf{E}f'$ replaced by a fresh AP.
- Note: each state sub-formula will be replaced by a fresh AP in both the labeling of the model and the formula.

Levels of State Sub-formulae

- 🌐 The state sub-formulae of level i are defined inductively as follows:
 - ☀️ Level 0 contains all atomic propositions.
 - ☀️ Level $i + 1$ contains all state sub-formulae g s.t. all state sub-formulae of g are of level i or less and g is not contained in any lower level.
- 🌐 Let g be a CTL* formula, then a sub-formula $\mathbf{E}h_1$ of g is *maximal* iff $\mathbf{E}h_1$ is not a strict sub-formula of any strict sub-formula $\mathbf{E}h$ of g .

State Sub-formulae (Examples)

- Consider the formula
 $\neg \mathbf{EF}(\neg \textit{Close} \wedge \textit{Start} \wedge \mathbf{E}(\mathbf{F}\textit{Heat} \wedge \mathbf{G}\textit{Error})).$
- The levels of the state sub-formulae are:
 - Level 0: *Close*, *Start*, *Heat*, and *Error*
 - Level 1: $\mathbf{E}(\mathbf{F}\textit{Heat} \wedge \mathbf{G}\textit{Error})$ and $\neg \textit{Close}$
 - Level 2: $\neg \textit{Close} \wedge \textit{Start}$
 - Level 3: $\neg \textit{Close} \wedge \textit{Start} \wedge \mathbf{E}(\mathbf{F}\textit{Heat} \wedge \mathbf{G}\textit{Error})$
 - Level 4: $\mathbf{EF}(\neg \textit{Close} \wedge \textit{Start} \wedge \mathbf{E}(\mathbf{F}\textit{Heat} \wedge \mathbf{G}\textit{Error}))$
 - Level 5: $\neg \mathbf{EF}(\neg \textit{Close} \wedge \textit{Start} \wedge \mathbf{E}(\mathbf{F}\textit{Heat} \wedge \mathbf{G}\textit{Error}))$
- Note: this is slightly different from [Clarke *et al.*].

CTL* Model Checking

- Let $M = (S, R, L)$ be a Kripke structure, f a CTL* formula, and g a state sub-formula of f of level i .
- The states of M have already been labelled correctly with all state sub-formulae of level smaller than i .
- In stage i , each such g is added to the labels of all states that make it true.
- For g a CTL* state formula, we proceed as follows:
 - If $g \in AP$, then g is in $label(s)$ iff it is in $L(s)$.
 - If $g = \neg g_1$, then g is in $label(s)$ iff g_1 is not in $label(s)$.
 - If $g = g_1 \vee g_2$, then g is added to $label(s)$ iff either g_1 or g_2 are in $label(s)$. (To reduce the number of levels, do analogously for $g_1 \wedge g_2$.)
 - If $g = \mathbf{E}g_1$ call the *CheckE*(g) procedure.

CheckE(g) Procedure

```
procedure CheckE( $g$ )  
  if  $g$  is a CTL formula then  
    apply CTL model checking for  $g$ ;  
    return; // next formula or next stage  
  end if;  
   $g' := g[a_1/\mathbf{E}h_1, \dots, a_k/\mathbf{E}h_k]$ ; //  $\mathbf{E}h_i$ 's are maximal sub-formulae  
  for all  $s \in S$   
    for  $i = 1, \dots, k$  do  
      if  $\mathbf{E}h_i \in \text{label}(s)$  then  $\text{label}(s) := \text{label}(s) \cup \{a_i\}$ ;  
    end for all;  
  apply LTL model checking for  $g'$ ;  
  for all  $s \in S$  do  
    if  $g' \in \text{label}(s)$  then  $\text{label}(s) := \text{label}(s) \cup \{g\}$ ;  
  end for all;  
end procedure;
```

Complexity of the Algorithm for CTL*

- 🌐 The complexity depends on the complexity of the algorithm for CTL and that for LTL.
- 🌐 So, if the previous algorithms are used, the complexity is $O((|S| + |R|) \cdot 2^{O(|f|)})$.
- 🌐 In real implementation, state sub-formulae need not be replaced by, but just need to be treated as, atomic propositions.