

Symbolic Model Checking

(Based on [Clarke et al. 1999] and [Kesten et al. 1995])

Yih-Kuen Tsay

(with help from Ming-Hsien Tsai and Jinn-Shu Chang)

Dept. of Information Management
National Taiwan University

Introduction

- 🌐 We have studied
 - ☀️ the operations on OBDDs and
 - ☀️ the encoding of a transition system in OBDDs.
- 🌐 How does one use OBDDs in model checking?
 - ☀️ Symbolic CTL model checking
 - ☀️ Symbolic LTL model checking
- 🌐 The model checking algorithms are **symbolic**, because they are based on the manipulation of Boolean functions (rather than state transition graphs).
- 🌐 Boolean functions (OBDDs) represent sets of states and transitions.
- 🌐 We can operate on **entire sets** rather than on individual states and transitions.

Fixpoints

- Let S be the set of all states of a system.
- A set $Z \in \mathcal{P}(S)$ is called a **fixpoint** of a function $\tau : \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ if $\tau(Z) = Z$.
- A temporal formula f can be viewed as a set Z of states such that
 - $Z \in \mathcal{P}(S)$ and
 - f is true exactly on the states in Z .
- Each temporal logic operator can be characterized by a fixpoint.

Complete Lattices

- Recall that a **complete lattice** is a partially ordered set in which every subset of elements has a **least upper bound** (supremum) and a **greatest lower bound** (infimum).
- For a given set S , $\langle \mathcal{P}(S), \subseteq \rangle$ forms a complete lattice.
- Let $S' \subseteq \mathcal{P}(S)$, then
 - the supremum of S' , usually denoted $\text{sup}(S')$, equals $\bigcup S'$ and
 - the infimum of S' , denoted $\text{inf}(S')$, equals $\bigcap S'$.
- The least element in $\mathcal{P}(S)$ is the empty set \emptyset , which we refer to as *False*.
- The greatest element in $\mathcal{P}(S)$ is the set S , which we refer to as *True*.

Predicate Transformer

- 🌐 A **predicate transformer** on $\mathcal{P}(S)$ is a function $\tau : \mathcal{P}(S) \rightarrow \mathcal{P}(S)$.
- 🌐 $\tau^i(Z)$ is used to denote i applications of τ to Z :
 - ☀️ $\tau^0(Z) = Z$
 - ☀️ $\tau^{i+1}(Z) = \tau(\tau^i(Z))$

Predicate Transformer (cont.)

Let τ be a predicate transformer.

τ is **monotonic** (order-preserving) provided that

$$P \subseteq Q \text{ implies } \tau(P) \subseteq \tau(Q).$$

τ is **\cup -continuous** provided that

$$P_1 \subseteq P_2 \subseteq \dots \text{ implies } \tau(\cup_i P_i) = \cup_i \tau(P_i).$$

τ is **\cap -continuous** provided that

$$P_1 \supseteq P_2 \supseteq \dots \text{ implies } \tau(\cap_i P_i) = \cap_i \tau(P_i).$$

LFP and GFP

- 🌐 We have seen the following results in a separate lecture.
- 🌐 $\mathcal{P}(S)$ is a complete lattice and hence also a CPO.
- 🌐 Consequently, a monotonic predicate transformer τ on $\mathcal{P}(S)$ always has
 - ☀️ a least fixpoint, denoted $\mu Z . \tau(Z)$, and
 - ☀️ a greatest fixpoint, denoted $\nu Z . \tau(Z)$.
- 🌐 More precisely,

$$\mu Z . \tau(Z) = \begin{cases} \bigcap \{Z \mid \tau(Z) \subseteq Z\} & \text{whenever } \tau \text{ is monotonic} \\ \bigcup_i \tau^i(\text{False}) & \text{whenever } \tau \text{ is also } \cup\text{-continuous} \end{cases}$$

$$\nu Z . \tau(Z) = \begin{cases} \bigcup \{Z \mid \tau(Z) \supseteq Z\} & \text{whenever } \tau \text{ is monotonic} \\ \bigcap_i \tau^i(\text{True}) & \text{whenever } \tau \text{ is also } \cap\text{-continuous} \end{cases}$$

Continuity of Predicate Transformers

Lemma (Lemma 5)

If S is finite and τ is monotonic, then τ is also \cup -continuous and \cap -continuous.

Proof:

🌐 Because S is finite, there is j_0 such that

- ☀ for every $j \geq j_0$, $P_j = P_{j_0}$, and
- ☀ for every $j < j_0$, $P_j \subseteq P_{j_0}$.

🌐 Thus, $\cup_i P_i = P_{j_0}$ and $\tau(\cup_i P_i) = \tau(P_{j_0})$.

🌐 Because τ is monotonic,

- ☀ $\tau(P_1) \subseteq \tau(P_2) \subseteq \dots$, and thus
- ☀ for every $j \geq j_0$, $\tau(P_j) = \tau(P_{j_0})$ and
- ☀ for every $j < j_0$, $\tau(P_j) \subseteq \tau(P_{j_0})$.


🌐 As a result, $\cup_i \tau(P_i) = \tau(P_{j_0}) = \tau(\cup_i P_i)$.


🌐 The proof that τ is \cap -continuous is similar.

Iterative Approximation


Lemma (Lemma 6)


If τ is monotonic, then for every $i (\geq 0)$


 $\tau^i(\text{False}) \subseteq \tau^{i+1}(\text{False})$, and

 $\tau^i(\text{True}) \supseteq \tau^{i+1}(\text{True})$.

Proof:

 By induction on i .

 Base case: $\tau^0(\text{False}) = \text{False} \subseteq \tau(\text{False})$.



 Inductive step: since τ is monotonic, $\tau^k(\text{False}) \subseteq \tau^{k+1}(\text{False})$ implies $\tau(\tau^k(\text{False})) \subseteq \tau(\tau^{k+1}(\text{False}))$ and hence $\tau^{(k+1)}(\text{False}) \subseteq \tau^{(k+1)+1}(\text{False})$, for $k \geq 0$.

 The other case is similar.

Convergence of Iterative Approximation

Lemma (Lemma 7)

If τ is monotonic and S is finite, then


-  there is an integer i_0 such that for every $j \geq i_0$,
 $\tau^j(\text{False}) = \tau^{i_0}(\text{False})$, and
-  similarly, there is some j_0 such that for every $j \geq j_0$,
 $\tau^j(\text{True}) = \tau^{j_0}(\text{True})$.

Lemma (Lemma 8)

If τ is monotonic and S is finite, then

- 🌐 there is an integer i_0 such that $\mu Z . \tau(Z) = \tau^{i_0}(\text{False})$, and
- 🌐 similarly, there is an integer j_0 such that $\nu Z . \tau(Z) = \tau^{j_0}(\text{True})$.

LFP Procedure

-  In a Kripke structure, if τ is monotonic, its least fixpoint can be computed by the following program.

```
function Lfp( $\tau$  : PredicateTransformer) : Predicate  
     $Q := \text{False}$ ;  
     $Q' := \tau(Q)$ ;  
    while ( $Q \neq Q'$ ) do  
         $Q := Q'$ ;  
         $Q' := \tau(Q)$ ;  
    end while;  
    return( $Q$ );  
end function
```

Correctness of LFP Procedure

- 🌐 The invariant of the while loop is

$$(Q' = \tau(Q)) \wedge (Q \subseteq \mu Z . \tau(Z))$$

(cf. $(Q' = \tau(Q)) \wedge (Q' \subseteq \mu Z . \tau(Z))$)

- 🌐 The number of iterations before the while loop terminates is bounded by $|S|$.
- 🌐 When the loop does terminate, we will have
 - ☀ $Q = \tau(Q)$ (Q is a fixpoint) and
 - ☀ $Q \subseteq \mu Z . \tau(Z)$.
- 🌐 Since Q is also a fixpoint, $\mu Z . \tau(Z) \subseteq Q$.
- 🌐 Hence $Q = \mu Z . \tau(Z)$.

GFP Procedure

- 🌐 We can also see that, if τ is monotonic, its greatest fixpoint can be computed by the following program.

```
function Gfp( $\tau$  : PredicateTransformer) : Predicate
   $Q := True$ ;
   $Q' := \tau(Q)$ ;
  while ( $Q \neq Q'$ ) do
     $Q := Q'$ ;
     $Q' := \tau(Q)$ ;
  end while;
  return( $Q$ );
end function
```

- 🌐 An analogous argument can be used to show that the procedure terminates and the value returns is $\nu Z . \tau(Z)$.

Characterization of CTL Operators

- Each CTL formula f is identified with the predicate $\{s \mid M, s \models f\}$ in $\mathcal{P}(S)$.
- It turns out that each of the basic CTL operators may be characterized as the least or greatest fixpoint of an appropriate predicate transformer.
- Least fixpoints correspond to eventualities.
- Greatest fixpoints correspond to properties that should hold forever.
- We will take a closer look at two cases:
 - $\mathbf{EG} f = \nu Z . f \wedge \mathbf{EX} Z$
 - $\mathbf{E}[f_1 \mathbf{U} f_2] = \mu Z . f_2 \vee (f_1 \wedge \mathbf{EX} Z)$

Characterization of EG

- 🌐 To see why $\mathbf{EG} f = \nu Z . f \wedge \mathbf{EX} Z$ intuitively ...
- 🌐 Let $\tau(Z) = f \wedge \mathbf{EX} Z$.
- 🌐 $\tau(\text{True}) = f \wedge \mathbf{EX} \text{True} = f$.
- 🌐 $\tau^2(\text{True}) = f \wedge \mathbf{EX} f$.
- 🌐 $\tau^3(\text{True}) = f \wedge \mathbf{EX} (f \wedge \mathbf{EX} f)$.
- 🌐 ...
- 🌐 $\tau^i(\text{True}) = f \wedge \mathbf{EX} (f \wedge \mathbf{EX} (\dots (f \wedge \mathbf{EX} f) \dots))$
(\mathbf{EX} is applied $i - 1$ times to the inner most f).
- 🌐 So, states in the limit of $\tau^i(\text{True})$ satisfy $\mathbf{EG} f$.

About $\tau(Z) = f \wedge \mathbf{EX} Z$

Lemma (Lemma 9)

$\tau(Z) = f \wedge \mathbf{EX} Z$ is monotonic.

Proof:

- 🌍 Let $P_1 \subseteq P_2$. We need to show that $\tau(P_1) \subseteq \tau(P_2)$.
- 🌍 Consider an arbitrary state $s \in \tau(P_1)$.
- 🌍 To show that $s \in \tau(P_2)$, it is sufficient to show that
 - ☀️ $s \models f$ and
 - ☀️ there is a successor of s which is in P_2 .
- 🌍 Because $s \in \tau(P_1)$,
 - ☀️ $s \models f$ and
 - ☀️ there exists a state s' such that $(s, s') \in R$ and $s' \in P_1$, which implies $s' \in P_2$ (since $P_1 \subseteq P_2$).
- 🌍 Thus $s \in \tau(P_2)$.

About $\tau(Z) = f \wedge \mathbf{EX} Z$ (cont.)

Lemma (Lemma 10)

Let $\tau(Z) = f \wedge \mathbf{EX} Z$ and let $\tau^{i_0}(\text{True})$ be the limit of the sequence $\text{True} \supseteq \tau(\text{True}) \supseteq \dots$. For every $s \in S$, if $s \in \tau^{i_0}(\text{True})$ then $s \models f$, and there is a state s' such that $(s, s') \in R$ and $s' \in \tau^{i_0}(\text{True})$.

Proof:

- Let $s \in \tau^{i_0}(\text{True})$.
- Because $\tau^{i_0}(\text{True})$ is a fixpoint of τ , $\tau^{i_0}(\text{True}) = \tau(\tau^{i_0}(\text{True}))$.
- Thus $s \in \tau(\tau^{i_0}(\text{True}))$.
- By definition of τ we get that $s \models f$ and there is a state s' , such that $(s, s') \in R$ and $s' \in \tau^{i_0}(\text{True})$.

About $\tau(Z) = f \wedge \mathbf{EX} Z$ (cont.)

Lemma (Lemma 11)

EG f is a fixpoint of the function $\tau(Z) = f \wedge \mathbf{EX} Z$.

Proof:

- 1. We first show $\mathbf{EG} f \subseteq f \wedge \mathbf{EX} \mathbf{EG} f$ and then $f \wedge \mathbf{EX} \mathbf{EG} f \subseteq \mathbf{EG} f$.
- 2. Suppose $s_0 \models \mathbf{EG} f$.
- 3. By the definition of \models , there is a path s_0, s_1, \dots in M such that for all k , $s_k \models f$.
- 4. This implies that $s_0 \models f$ and $s_1 \models \mathbf{EG} f$.
- 5. In other words, $s_0 \models f$ and $s_0 \models \mathbf{EX} \mathbf{EG} f$.
- 6. Thus, $\mathbf{EG} f \subseteq f \wedge \mathbf{EX} \mathbf{EG} f$.
- 7. Similarly, if $s_0 \models f \wedge \mathbf{EX} \mathbf{EG} f$, then $s_0 \models \mathbf{EG} f$.
- 8. Thus, $f \wedge \mathbf{EX} \mathbf{EG} f \subseteq \mathbf{EG} f$.

About $\tau(Z) = f \wedge \mathbf{EX} Z$ (cont.)

Lemma (Lemma 12)

EG f is the greatest fixpoint of the function $\tau(Z) = f \wedge \mathbf{EX} Z$.

Proof:

- 🌐 Because τ is monotonic (Lemma 9), by Lemma 5 it is also \cap -continuous.
- 🌐 In order to show that **EG** f is the greatest fixpoint of τ , it is sufficient to prove that **EG** $f = \bigcap_i \tau^i(\text{True})$, i.e.,
 - ☀️ **EG** $f \subseteq \bigcap_i \tau^i(\text{True})$ and
 - ☀️ $\bigcap_i \tau^i(\text{True}) \subseteq \mathbf{EG} f$.

About $\tau(Z) = f \wedge \mathbf{EX} Z$ (cont.)

Proof of $\mathbf{EG} f \subseteq \bigcap_i \tau^i(\text{True})$:

- 🌐 It suffices to show that $\mathbf{EG} f \subseteq \tau^i(\text{True})$, for all i .
- 🌐 The proof is by induction on i .
- 🌐 Base case: clearly, $\mathbf{EG} f \subseteq \text{True} = \tau^0(\text{True})$.
- 🌐 Inductive step:
 - ☀ Assume that $\mathbf{EG} f \subseteq \tau^k(\text{True})$, for an arbitrary k .
 - ☀ Because τ is monotonic, $\tau(\mathbf{EG} f) \subseteq \tau(\tau^k(\text{True})) = \tau^{k+1}(\text{True})$.
 - ☀ By Lemma 11 ($\mathbf{EG} f$ is a fixpoint of τ), $\tau(\mathbf{EG} f) = \mathbf{EG} f$.
 - ☀ Hence, $\mathbf{EG} f \subseteq \tau^{k+1}(\text{True})$.

About $\tau(Z) = f \wedge \mathbf{EX} Z$ (cont.)

Proof of $\bigcap_i \tau^i(\text{True}) \subseteq \mathbf{EG} f$:

- Consider some state $s \in \bigcap_i \tau^i(\text{True})$.
- The state s is included in every $\tau^i(\text{True})$.
- Hence, it is also in the fixpoint $\tau^{i_0}(\text{True})$.
- By Lemma 10, s is the start of an infinite sequence of states in which each state is related to the previous one by the relation R .
- Furthermore, each state in the sequence satisfies f .
- Thus $s \models \mathbf{EG} f$.

Characterization of EU

- To see why $\mathbf{E}[f_1 \mathbf{U} f_2] = \mu Z . f_2 \vee (f_1 \wedge \mathbf{E} X Z)$ intuitively ...
- Let $\tau(Z) = f_2 \vee (f_1 \wedge \mathbf{E} X Z)$.
- $\tau(\text{False}) = f_2 \vee (f_1 \wedge \mathbf{E} X \text{False}) = f_2$.
- $\tau^2(\text{False}) = f_2 \vee (f_1 \wedge \mathbf{E} X f_2)$.
- $\tau^3(\text{False}) = f_2 \vee (f_1 \wedge \mathbf{E} X (f_2 \vee (f_1 \wedge \mathbf{E} X f_2)))$.
- ...
- $\tau^i(\text{False}) = f_2 \vee (f_1 \wedge \mathbf{E} X (f_2 \vee (f_1 \wedge \mathbf{E} X (\dots (f_2 \vee (f_1 \wedge \mathbf{E} X f_2)) \dots))))$
($\mathbf{E} X$ is applied $i - 1$ times to the inner most f_2).
- f_2 will eventually become true on some path; Before then, f_1 remains true.
- So, states in the limit of $\tau^i(\text{False})$ satisfy $\mathbf{E}[f_1 \mathbf{U} f_2]$.

About $\tau(Z) = f_2 \vee (f_1 \wedge \mathbf{EX} Z)$

Lemma (Lemma 13)

$\mathbf{E}[f_1 \mathbf{U} f_2]$ is the least fixpoint function of the function $\tau(Z) = f_2 \vee (f_1 \wedge \mathbf{EX} Z)$.

Proof:

- 🌍 $\tau(Z) = f_2 \vee (f_1 \wedge \mathbf{EX} Z)$ is monotonic, hence τ is \mathbf{U} -continuous.
- 🌍 $\mathbf{E}[f_1 \mathbf{U} f_2]$ is a fixpoint of $\tau(Z)$.
- 🌍 We still need to prove that $\mathbf{E}[f_1 \mathbf{U} f_2]$ is the least fixpoint of $\tau(Z)$.
- 🌍 It is sufficient to show that $\mathbf{E}[f_1 \mathbf{U} f_2] = \bigcup_i \tau^i(\text{False})$, i.e.,
 - ☀️ $\bigcup_i \tau^i(\text{False}) \subseteq \mathbf{E}[f_1 \mathbf{U} f_2]$ and
 - ☀️ $\mathbf{E}[f_1 \mathbf{U} f_2] \subseteq \bigcup_i \tau^i(\text{False})$.

About $\tau(Z) = f_2 \vee (f_1 \wedge \mathbf{EX} Z)$ (cont.)

Proof of $\cup_i \tau^i(\text{False}) \subseteq \mathbf{E}[f_1 \mathbf{U} f_2]$:

- 🌐 It suffices to show that $\tau^i(\text{False}) \subseteq \mathbf{E}[f_1 \mathbf{U} f_2]$ for all i .
- 🌐 We prove this by induction on i .
- 🌐 Base case: $\tau^0(\text{False}) = \text{False} \subseteq \mathbf{E}[f_1 \mathbf{U} f_2]$.
- 🌐 Inductive step:
 - ☀ We assume $\tau^k(\text{False}) \subseteq \mathbf{E}[f_1 \mathbf{U} f_2]$ for an arbitrary k .
 - ☀ By the monotonicity of τ , $\tau(\tau^k(\text{False})) \subseteq \tau(\mathbf{E}[f_1 \mathbf{U} f_2])$.
 - ☀ Since $\mathbf{E}[f_1 \mathbf{U} f_2]$ is a fixpoint of $\tau(Z)$, $\tau(\mathbf{E}[f_1 \mathbf{U} f_2]) = \mathbf{E}[f_1 \mathbf{U} f_2]$.
 - ☀ It follows that $\tau^{k+1}(\text{False}) \subseteq \mathbf{E}[f_1 \mathbf{U} f_2]$.

About $\tau(Z) = f_2 \vee (f_1 \wedge \mathbf{EX} Z)$ (cont.)

Proof of $\mathbf{E}[f_1 \mathbf{U} f_2] \subseteq \cup_i \tau^i(\text{False})$:

- 🌍 We prove this direction by induction on the length of the prefix of the path along which $f_1 \mathbf{U} f_2$ is satisfied.
- 🌍 If $s \in \mathbf{E}[f_1 \mathbf{U} f_2]$ (i.e., $s \models \mathbf{E}[f_1 \mathbf{U} f_2]$), then there exists a path $\pi = s_1, s_2, \dots$ with $s = s_1$ such that, for some $j \geq 1$, $s_j \models f_2$ and, for all $l < j$, $s_l \models f_1$.
- 🌍 We claim the following:
For every $\pi = s_1, s_2, \dots$, if $\pi \models f_1 \mathbf{U} f_2$, then for every j such that $s_j \models f_2$ and, for all $l < j$, $s_l \models f_1$, $s_1 \in \tau^j(\text{False})$ holds.
- 🌍 From the claim, it follows that $s \in \mathbf{E}[f_1 \mathbf{U} f_2]$ implies $s \in \tau^j(\text{False})$ for some j .
- 🌍 Therefore, $\mathbf{E}[f_1 \mathbf{U} f_2] \subseteq \cup_i \tau^i(\text{False})$.

About $\tau(Z) = f_2 \vee (f_1 \wedge \mathbf{EX} Z)$ (cont.)

Proof of $\mathbf{E}[f_1 \mathbf{U} f_2] \subseteq \cup_i \tau^i(\text{False})$ (continued):

🌐 We now prove the claim by induction on j .

🌐 Base case ($j = 1$):

☀️ $s_1 \models f_2$ and therefore $s_1 \in f_2 \vee (f_1 \wedge \mathbf{EX} \text{False}) = \tau(\text{False})$.

🌐 Inductive step:

☀️ Let π be a path $s_1, s_2, \dots, s_k, \dots$ with $k > 1$ such that $s_k \models f_2$ and for all $l < k$, $s_l \models f_1$ (so, $\pi \models f_1 \mathbf{U} f_2$).

☀️ Since $k > 1$, s_2, s_3, \dots also satisfies $f_1 \mathbf{U} f_2$. More precisely, s_2 is the start of a sequence $\pi' = s'_1, s'_2, \dots$ ($= s_2, s_3, \dots$) such that $s'_{k-1} (= s_k) \models f_2$ and for all $l < k - 1$, $s'_l \models f_1$.

☀️ From the induction hypothesis, $s'_1 \in \tau^{k-1}(\text{False})$, i.e., $s_2 \in \tau^{k-1}(\text{False})$.

☀️ With $s_1 \models f_1$, $(s_1, s_2) \in R$, and $s_2 \in \tau^{k-1}(\text{False})$, we have $s_1 \in f_1 \wedge \mathbf{EX} (\tau^{k-1}(\text{False})) \subseteq f_2 \vee (f_1 \wedge \mathbf{EX} (\tau^{k-1}(\text{False}))) = \tau^k(\text{False})$.

An Example

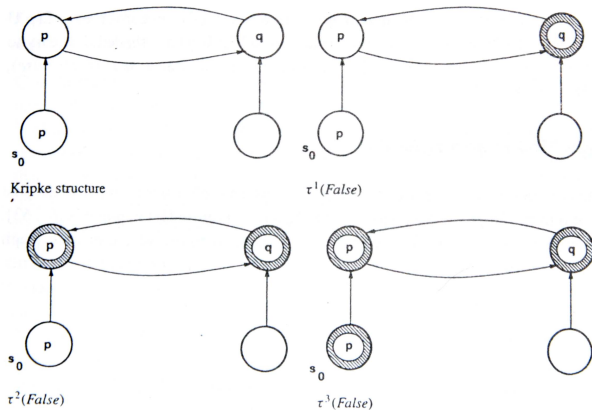


Figure 6.3
Sequence of approximations for $E[p \text{ U } q]$.









Source: [Clarke *et al.* 1999]. Names of states (clockwise): s_0, s_1, s_2, s_3 .

An Example (cont.)

Sequence of approximations for $\mathbf{E}[p \mathbf{U} q] = \mu Z . q \vee (p \wedge \mathbf{E} X Z)$:

$$\begin{aligned}\tau^1(\text{False}) &= q \vee (p \wedge \mathbf{E} X \text{False}) \\ &= q \\ \tau^2(\text{False}) &= q \vee (p \wedge \mathbf{E} X \tau(\text{False})) \\ &= q \vee (p \wedge \mathbf{E} X q) \\ &= q \vee (p \wedge \{s_1, s_3\}) \\ &= q \vee \{s_1\} \\ \tau^3(\text{False}) &= q \vee (p \wedge \mathbf{E} X \tau^2(\text{False})) \\ &= q \vee (p \wedge \mathbf{E} X (q \vee \{s_1\})) \\ &= q \vee (p \wedge \{s_0, s_1, s_2, s_3\}) \\ &= q \vee p\end{aligned}$$

Characterization of CTL Operators (cont.)

-  $\mathbf{AF} f = \mu Z . f \vee \mathbf{AX} Z$
-  $\mathbf{EF} f = \mu Z . f \vee \mathbf{EX} Z$
-  $\mathbf{AG} f = \nu Z . f \wedge \mathbf{AX} Z$
-  $\mathbf{EG} f = \nu Z . f \wedge \mathbf{EX} Z$
-  $\mathbf{A}[f \mathbf{U} g] = \mu Z . g \vee (f \wedge \mathbf{AX} Z)$
-  $\mathbf{E}[f \mathbf{U} g] = \mu Z . g \vee (f \wedge \mathbf{EX} Z)$
-  $\mathbf{A}[f \mathbf{R} g] = \nu Z . g \wedge (f \vee \mathbf{AX} Z)$
-  $\mathbf{E}[f \mathbf{R} g] = \nu Z . g \wedge (f \vee \mathbf{EX} Z)$

Symbolic Model Checking for CTL

- 🌐 There is a quite fast explicit state model checking algorithm for CTL, but a state explosion problem may occur.
- 🌐 In the following, we will present a **Symbolic Model Checking** (SMC) algorithm for CTL which operates on Kripke structures represented symbolically using OBDDs.
- 🌐 For this, the logic of **Quantified Boolean Formulae** (QBF) will be used.
 - ☀️ QBF formulae are as expressive as the usual Boolean formulae.
 - ☀️ However, they allow a more succinct notation for complex operations on Boolean formulae.

Quantified Boolean Formulae (QBF)

- Let V be a set $\{v_0, \dots, v_{n-1}\}$ of propositional variables.
- $QBF(V)$ is the smallest set of formulae such that
 - every variable in V is a formula,
 - if f and g are formulae, then $\neg f$, $f \vee g$, and $f \wedge g$ are formulae, and
 - if f is a formula and $v \in V$, then $\exists v f$ and $\forall v f$ are formulae.

Truth Assignment

- A *truth assignment* for $QBF(V)$ is a function $\sigma : V \rightarrow \{0, 1\}$.
- If $a \in \{0, 1\}$, then the notation $\sigma\langle v \leftarrow a \rangle$ is used for the truth assignment defined by

$$\sigma\langle v \leftarrow a \rangle(w) = \begin{cases} a & \text{if } v = w \\ \sigma(w) & \text{otherwise} \end{cases}$$

Models of QBF

🌐 $\sigma \models f$ denotes that the QBF formula f is true under the assignment σ .

🌐 The \models (satisfaction) relation is defined inductively as follows:

$$\sigma \models v \quad \text{iff} \quad \sigma(v) = 1$$

$$\sigma \models \neg f \quad \text{iff} \quad \sigma \not\models f$$

$$\sigma \models f \vee g \quad \text{iff} \quad \sigma \models f \text{ or } \sigma \models g$$

$$\sigma \models f \wedge g \quad \text{iff} \quad \sigma \models f \text{ and } \sigma \models g$$

$$\sigma \models \exists v f \quad \text{iff} \quad \sigma \langle v \leftarrow 0 \rangle \models f \text{ or } \sigma \langle v \leftarrow 1 \rangle \models f$$

$$\sigma \models \forall v f \quad \text{iff} \quad \sigma \langle v \leftarrow 0 \rangle \models f \text{ and } \sigma \langle v \leftarrow 1 \rangle \models f$$

- 🌐 The quantifiers in QBF can be implemented as combinations of the restrict and apply operators.


$$\begin{aligned}\exists x f &= f|_{x \leftarrow 0} \vee f|_{x \leftarrow 1} \\ \forall x f &= f|_{x \leftarrow 0} \wedge f|_{x \leftarrow 1}\end{aligned}$$

- 🌐 So, like Boolean formulae, QBF formulae can be represented by OBDDs.

- 🌐 The SMC algorithm is implemented by a procedure *Check*.
 - ☀️ Argument: a CTL formula
 - ☀️ Return: an OBDD that represents exactly those states of the system that satisfy the formula

SMC Algorithm (cont.)

| | | |
|-------------------------------------|---|---|
| $Check(a)$ | = | the OBDD representing the set of states satisfying the atomic proposition a |
| $Check(f \wedge g)$ | = | $Check(f) \wedge Check(g)$ |
| $Check(\neg f)$ | = | $\neg Check(f)$ |
| $Check(\mathbf{EX} f)$ | = | $CheckEX(Check(f))$ |
| $Check(\mathbf{E}[f \mathbf{U} g])$ | = | $CheckEU(Check(f), Check(g))$ |
| $Check(\mathbf{EG} f)$ | = | $CheckEG(Check(f))$ |

-  The formula **EX** f is true in a state if the state has a successor in which f is true.

$$\text{CheckEX}(f(\bar{v})) = \exists \bar{v}' [f(\bar{v}') \wedge R(\bar{v}, \bar{v}')],$$

where $R(\bar{v}, \bar{v}')$ is the OBDD representation of the transition relation.

- 🌐 *CheckEU* is based on the least fixpoint characterization for the CTL operator **EU**.

$$\mathbf{E}[f \mathbf{U} g] = \mu Z . g \vee (f \wedge \mathbf{E} X Z)$$

- 🌐 The function Lfp is used to compute a sequence of approximations

$$Q_0, Q_1, \dots, Q_i, \dots$$

that converges to $\mathbf{E}[f \mathbf{U} g]$ in a finite number of steps.

- 🌐 If we have OBDDs for f , g , and the current approximation Q_i , then we can compute an OBDD for the next approximation Q_{i+1} .
- 🌐 When $Q_i = Q_{i+1}$ (it is easy to test because OBDDs provide a canonical form of Boolean functions), the function Lfp terminates.

- 🌐 *CheckEG* is based on the greatest fixpoint characterization for the CTL operator **EG**.

$$\mathbf{EG} f = \nu Z . f \wedge \mathbf{EX} Z$$

Fairness in SMC

- Assume the fairness constraints are given by a set of CTL formulae $F = \{P_1, \dots, P_n\}$.
- A fair path is a path on which each formula in F holds infinitely often.
- We define a new procedure *CheckFair* for checking CTL formulae relative to the fairness constructions in F .
- We do this by defining new intermediate procedures *CheckFairEX*, *CheckFairEU*, and *CheckFairEG*, which correspond to the intermediate procedures used to define *Check*.

EG f with Fairness

- 🌐 Consider the formula **EG** f given fairness constraints F .
- 🌐 The formula means that there exists a fair path beginning with the current state on which f holds globally.
- 🌐 The set of such states Z is the largest set with the following two properties:
 - ☀ all of the states in Z satisfy f , and
 - ☀ for all $P_k \in F$ and all $s \in Z$, there is a sequence of states of **length one or greater** from s to a state in Z satisfying P_k such that all states on the path satisfy f .
(cf. There exists a path in S' , where f holds, that leads from s to some node t in a **nontrivial fair strongly connected component** of the graph (S', R') .)

EG f with Fairness (cont.)

- The characterization can be expressed by means of a fixpoint as follows:

$$\mathbf{EG} f = \nu Z . f \wedge \bigwedge_{k=1}^n \mathbf{EXE}[f \mathbf{U} (Z \wedge P_k)]$$

- Note that the formula is not directly expressible in CTL.
- We are going to prove the correctness of this equation.
- We split it into two lemmas.

Fair Version of EG f

Lemma (Lemma 14)

The fair version of **EG** f is a fixpoint of the equation

$$Z = f \wedge \bigwedge_{k=1}^n \mathbf{EXE}[f \mathbf{U} (Z \wedge P_k)].$$

Proof: It suffices to show that

$$\mathbf{EG} f \subseteq f \wedge \bigwedge_{k=1}^n \mathbf{EXE}[f \mathbf{U} (\mathbf{EG} f \wedge P_k)]$$

and

$$f \wedge \bigwedge_{k=1}^n \mathbf{EXE}[f \mathbf{U} (\mathbf{EG} f \wedge P_k)] \subseteq \mathbf{EG} f.$$

Fair Version of EG f (cont.)

🌐 Case 1: $\mathbf{EG} f \subseteq f \wedge \bigwedge_{k=1}^n \mathbf{EXE}[f \mathbf{U} (\mathbf{EG} f \wedge P_k)]$.

- ☀️ Let $s \models \mathbf{EG} f$, then s is the start of a fair path π , all of whose states satisfy f .
- ☀️ Let s_i be the first state on π such that $s_i \in P_i$ and $s_i \neq s$.
- ☀️ The state s_i is also a start of a fair path along which all states satisfy f .
- ☀️ Thus, $s_i \in \mathbf{EG} f$.
- ☀️ It follows that for every i , $s \models f \wedge \mathbf{EXE}[f \mathbf{U} (\mathbf{EG} f \wedge P_i)]$.
- ☀️ Therefore, $s \models f \wedge \bigwedge_{k=1}^n \mathbf{EXE}[f \mathbf{U} (\mathbf{EG} f \wedge P_k)]$.

Fair Version of EG f (cont.)

🌐 Case 2: $f \wedge \bigwedge_{k=1}^n \mathbf{EXE}[f \mathbf{U}(\mathbf{EG} f \wedge P_k)] \subseteq \mathbf{EG} f$.

- ☀️ If $s \models f \wedge \bigwedge_{k=1}^n \mathbf{EXE}[f \mathbf{U}(\mathbf{EG} f \wedge P_k)]$, then there is a finite path starting from s to a state s' such that $s' \models (\mathbf{EG} f \wedge P_k)$.
- ☀️ Every state on the path from s to s' satisfies f .
- ☀️ s' is the beginning of a fair path such that each state on the path satisfies f .
- ☀️ Thus, $s \models \mathbf{EG} f$.

Fair Version of EG f (cont.)

Lemma (Lemma 15)

The greatest fixpoint of the following equation is included in **EG** f .

$$Z = f \wedge \bigwedge_{k=1}^n \mathbf{EXE}[f \mathbf{U} (Z \wedge P_k)]$$

Fair Version of EG f (cont.)


Proof of Lemma 15:

- 🌐 Let Z be an arbitrary fixpoint of the formula.
- 🌐 Assume that $s \in Z$. Then $s \models f$.
- 🌐 s has a successor s' that is a start of a path to a state s_1 such that
 - ☀ all states on this path satisfy f and
 - ☀ s_1 satisfies $Z \wedge P_1$.
- 🌐 Because $s_1 \in Z$ we can conclude by the same argument that there is a path from s_1 to a state s_2 in P_2 .

Fair Version of EG f (cont.)

Proof of Lemma 15 (continued):

- Using this argument n times we conclude that s is the start of a path along which all states satisfy f and which passes through P_1, \dots, P_k .
- The last state on the path is in Z , and thus there is a path from this state back to some state in P_1 .
- Induction can be used to show that there exists a fair path starting at s such that f is satisfied along the path, i.e., $s \models \mathbf{EG} f$.

-  $CheckFairEG(f(\bar{v}))$ is based on the following fixpoint characterization:

$$\nu Z(\bar{v}) . f(\bar{v}) \wedge \bigwedge_{k=1}^n \mathbf{EXE}[f(\bar{v}) \mathbf{U} (Z(\bar{v}) \wedge P_k)].$$

- 🌐 The set of all states which are the start of some fair computation is

$$\mathit{fair}(\bar{v}) = \mathit{CheckFair}(\mathbf{EG} \text{ True}).$$

- 🌐 The formula $\mathbf{EX} f$ under fairness constraints is equivalent to the formula $\mathbf{EX} f \wedge \mathit{fair}$ without fairness constraints.

$$\mathit{CheckFairEX}(f(\vec{v})) = \mathit{CheckEX}(f(\vec{v}) \wedge \mathit{fair}(\vec{v}))$$

- 🌐 The formula $\mathbf{E}[f \mathbf{U} g]$ under fairness constraints is equivalent to the formula $\mathbf{E}[f \mathbf{U} g \wedge \mathit{fair}]$ without fairness constraints.

$$\mathit{CheckFairEU}(f(\bar{v}), g(\bar{v})) = \mathit{CheckEU}(f(\bar{v}), g(\bar{v}) \wedge \mathit{fair}(\bar{v}))$$

LTL Model Checking

- Let $\mathbf{A} f$ be a linear temporal logic formula where f is a restricted path formula.
- A formula f is a **restricted path formula** if all state subformulae in f are atomic propositions.
- The problem is to determine all of those states $s \in S$ such that $M, s \models \mathbf{A} f$.
- Since $M, s \models \mathbf{A} f$ iff $M, s \models \neg \mathbf{E} \neg f$, it is sufficient to check the truth of formulae of the form $\mathbf{E} f$.

LTL Model Checking (cont.)

- Given a formula $\mathbf{E} f$ and a Kripke structure M , the procedure of LTL model checking is:
 - Construct a tableau T for the path formula f .
 - Compose T with M .
 - Find a path in the composition.
- The tableau can be represented by OBDDs.

States of the Tableau

- Each state in the tableau is a set of elementary formulae obtained from f .
- The set of elementary subformulae of f is denoted by $el(f)$ and is defined recursively as follows.

$$\begin{aligned}el(p) &= \{p\} \text{ if } p \in AP_f \\el(\neg g) &= el(g) \\el(g \vee h) &= el(g) \cup el(h) \\el(\mathbf{X}g) &= \{\mathbf{X}g\} \cup el(g) \\el(g \mathbf{U} h) &= \{\mathbf{X}(g \mathbf{U} h)\} \cup el(g) \cup el(h)\end{aligned}$$

- The set of states S_T of T is $\mathcal{P}(el(f))$.

Transition Relation of the Tableau

🌐 An additional function sat is defined recursively as follows.

$$sat(g) = \{s \mid g \in s\} \text{ where } g \in el(f)$$

$$sat(\neg g) = \{s \mid s \notin sat(g)\}$$

$$sat(g \vee h) = sat(g) \cup sat(h)$$

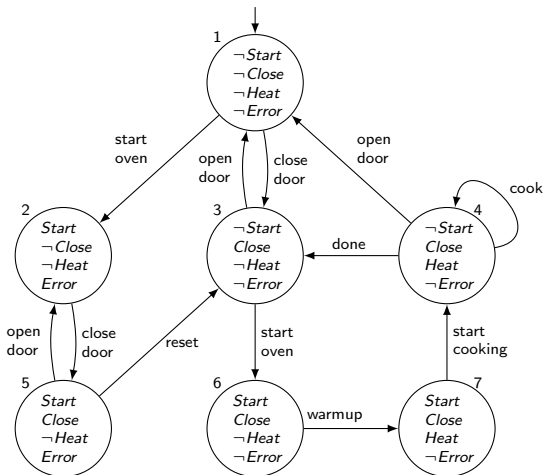
$$sat(g \mathbf{U} h) = sat(h) \cup (sat(g) \cap sat(\mathbf{X}(g \mathbf{U} h)))$$

🌐 The transition relation R_T of T is defined as

$$R_T(s, s') = \bigwedge_{\mathbf{X}g \in el(f)} s \in sat(\mathbf{X}g) \Leftrightarrow s' \in sat(g)$$

- 🌐 An additional condition is necessary in order to identify those paths along which f holds.
- 🌐 A path π that starts from a state $s \in \text{sat}(f)$ will satisfy f iff
 - ☀️ for every subformula $g \mathbf{U} h$ and for every state s on π , if $s \in \text{sat}(g \mathbf{U} h)$ then either $s \in \text{sat}(h)$ or there is a later state t on π such that $t \in \text{sat}(h)$.

The Microwave Oven Example



Source: redrawn from [Clarke et al. 1999, Fig. 4.3].

The Microwave Oven Example (cont.)

Tableau for $\neg g = \neg(\neg \text{heat } \mathbf{U} \text{ close})$:

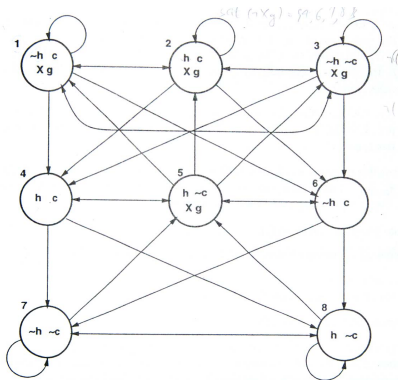


Figure 6.9
Tableau for $(\neg \text{heat}) \mathbf{U} \text{ close}$.

Source: [Clarke *et al.* 1999].

Eventuality

- 🌐 The definition of R_T does not guarantee that eventuality properties are fulfilled.
- 🌐 A path π that starts from a state $s \in \text{sat}(f)$ will satisfy f if and only if
 - ☀️ for every subformulae $g \mathbf{U} h$ and for every state s on π , if $s \in \text{sat}(g \mathbf{U} h)$ then either $s \in \text{sat}(h)$ or there is a later state t on π such that $t \in \text{sat}(h)$.

- 🌐 $\pi' = s'_0, s'_1, \dots$ represents a path in M .
- 🌐 For the suffix $\pi'_i = s'_i, s'_{i+1}, \dots$ of π , we define

$$s_i = \{\psi \mid \psi \in el(f) \text{ and } M, \pi' \models \psi\}$$

Correctness

Lemma (Lemma 16)

Let $sub(f)$ be the set of all subformulae of f . For all $g \in sub(f) \cup el(f)$, $M, \pi'_i \models g$ if and only if $s_i \in sat(g)$.

Proof:

- 🌐 Case 1: Let $g \in el(f)$.
 - ☀️ $M, \pi'_i \models g$ iff $g \in s_i$.
 - ☀️ $g \in s_i$ iff $s_i \in sat(g)$.
- 🌐 Case 2: Let $g = \neg g_1$ or $g = g_1 \vee g_2$.
- 🌐 Case 3: Let $g = g_1 \mathbf{U} g_2$.
 - ☀️ $M, \pi'_i \models g_1 \mathbf{U} g_2$ iff $M, \pi'_i \models g_2$ or $(M, \pi'_i \models g_1$ and $M, \pi'_i \models \mathbf{X}(g_1 \mathbf{U} g_2))$.
 - ☀️ $M, \pi'_i \models g_2$ or $(M, \pi'_i \models g_1$ and $M, \pi'_i \models \mathbf{X}(g_1 \mathbf{U} g_2))$ iff $s_i \in sat(g_2) \vee (s_i \in sat(g_1) \wedge s_i \in sat(\mathbf{X}(g_1 \mathbf{U} g_2)))$.
 - ☀️ $s_i \in sat(g_2) \vee (s_i \in sat(g_1) \wedge s_i \in sat(\mathbf{X}(g_1 \mathbf{U} g_2)))$ iff $s_i \in sat(g_1 \mathbf{U} g_2)$.

Correctness (cont.)

Lemma (Lemma 17)

Let $\pi' = s'_0 s'_1 \dots$ be a path in M . For all $i \geq 0$, let s_i be the tableau state. Then $\pi = s_0 s_1 \dots$ is a path in T .

Correctness (cont.)

Theorem (Theorem 4)

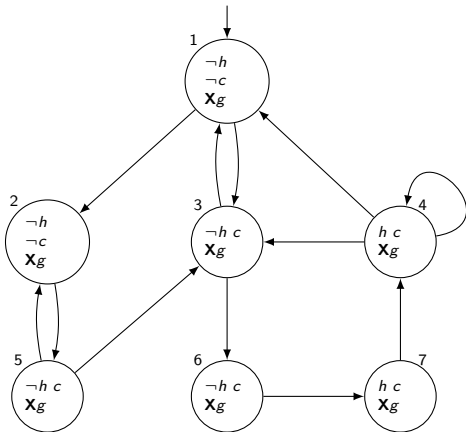
Let T be the tableau for the path formula f . Then, for every Kripke structure M and every path π' of M , if $M, \pi' \models f$ then there is a path π in T that starts in a state in $\text{sat}(f)$, such that $\text{label}(\pi') \upharpoonright_{AP_f} = \text{label}(\pi)$.

Composition of T and M

- 🌐 $P = (S, R, L)$ is the product of the tableau $T = (S_T, R_T, L_T)$ and the Kripke structure $M = (S_M, R_M, L_M)$.
 - ☀️ $S = \{(s, s') \mid s \in S_T, s' \in S_M \text{ and } L_M(s') \upharpoonright_{AP_f} = L_T(s)\}$.
 - ☀️ $R((s, s'), (t, t'))$ iff $R_T(s, t)$ and $R_M(s', t')$.
 - ☀️ $L((s, s')) = L_T(s)$.
- 🌐 The function sat is extended to be defined over S by $(s, s') \in sat(g)$ if and only if $s \in sat(g)$.

The Microwave Oven Example (cont.)

Product of the microwave and the tableau for $\neg(\neg\text{heat } \mathbf{U} \text{ close})$:



Source: adapted from [Clarke *et al.* 1999, Fig. 6.10].

Correctness

Lemma (Lemma 18)

$\pi'' = (s_0, s'_0), (s_1, s'_1), \dots$ is a path in P with $L_P((s_i, s'_i)) = L_T(s_i)$ for all $i \geq 0$ if and only if there exists a path $\pi = s_0, s_1, \dots$ in T , and a path $\pi' = s'_0, s'_1, \dots$ in M with $L_T(s_i) = L_M(s'_i) \upharpoonright_{AP_f}$ for all $i \geq 0$.

Correctness (cont.)

Theorem (Theorem 5)

$M, s' \models \mathbf{E} f$ if and only if there is a state s in T such that $(s, s') \in \text{sat}(f)$ and $P, (s, s') \models \mathbf{EG} \text{ True}$ under fairness constraints

$$\{\text{sat}(\neg(g \mathbf{U} h) \vee h) \mid g \mathbf{U} h \text{ occurs in } f\}.$$

Summary of LTL Model Checking

- Given a Kripke structure M , a state s' in M and a LTL formula f .
- Construct a symbolic representation of M .
- Construct a symbolic representation of $T_{\neg f}$.
- Construct the product P of M and $T_{\neg f}$.
- Use the symbolic CTL model checking algorithm to check if there is a state s in $T_{\neg f}$ such that
 - $(s, s') \in \text{sat}(\neg f)$ and
 - $P, (s, s') \models \mathbf{EG True}$ under fairness constraints

$$\{\text{sat}(\neg(g \mathbf{U} h) \vee h) \mid g \mathbf{U} h \text{ occurs in } f\}.$$

- 🌐 Here we slightly modify the definition of Kripke structures and the symbolic algorithm in [Kesten *et al.* 1995].
- 🌐 A Kripke structure M is a tuple (V, S_0, R) where
 - ☀ V is a set of system variables and thus the set of states S is the set of all valuations for V ,
 - ☀ S_0 is the initial condition defined upon V , and
 - ☀ $R \subseteq S \times S$ is the transition relation which is total.
- 🌐 The problem is to check, given a Kripke structure M and a formula f , whether $M \models f$ (all paths of M satisfy f).

- Let V_f be the set of all propositions in f . Without loss of generality, we assume $V_f = V$ (of the Kripke structure).
- For each elementary formula $p \in el(f)$, a Boolean variable (elementary variable) x_p is associated.
- The set of elementary variables are represented by a vector $\bar{x} = x_1, x_2, \dots, x_m$ where $m = |el(f)|$.
- Note that a valuation for \bar{x} constitutes a state in M and a state in T_f .

Formulae in Elementary Formulae

- Let $CL(f)$ denote the closure of the LTL formula f .
- For each formula $p \in CL(f)$, we define a Boolean function $\chi_p(\bar{x})$ which expresses p in terms of the elementary variables:

$$\text{For } p \in el(f), \chi_p(\bar{x}) = x_p$$

$$\text{For } p = \neg q, \chi_p = \neg \chi_q$$

$$\text{For } p = q \wedge r, \chi_p = \chi_q \wedge \chi_r$$

$$\text{For } p = q \mathbf{U} r, \chi_p = \chi_r \vee (\chi_q \wedge \mathbf{X}\chi_{(q \mathbf{U} r)})$$




$$\text{For } p = q \mathbf{S} r, \chi_p = \chi_r \vee (\chi_q \wedge \mathbf{X}\chi_{(q \mathbf{S} r)})$$

Note: \mathbf{Y} is the “previous” operator.

- There exists a computation in M satisfying f iff $sat_{M,f}$ as defined below is true.

$$sat_{M,f} : \exists \bar{x}, \bar{y} : init(\bar{x}) \wedge E^*(\bar{x}, \bar{y}) \wedge scf^E(\bar{y})$$

Initial Condition

-  The following formula identifies an initial state in the product of M and T_f .
-  It is an initial state in M .
 -  It is also an initial atom in T_f .

$$init(\bar{x}) : \chi_f(\bar{x}) \wedge \left(\bigwedge_{\mathbf{Y}p \in CL(f)} \neg x_{\mathbf{Y}p} \right) \wedge S_0(\bar{x})$$

Transition Relation

- The following formula identifies the set of transitions in the product:

$$E(\bar{x}, \bar{y}) : e(\bar{x}, \bar{y}) \wedge R(\bar{x}, \bar{y})$$

where

$$e(\bar{x}, \bar{y}) : \bigwedge_{\mathbf{x}_p \in el(f)} (\mathbf{x}_{\mathbf{x}_p} \leftrightarrow \chi_p(\bar{y})) \wedge \bigwedge_{\mathbf{y}_p \in el(f)} (\chi_p(\bar{x}) \leftrightarrow \mathbf{y}_{\mathbf{y}_p})$$

$$E^+(\bar{x}, \bar{y}) = E(\bar{x}, \bar{y}) \vee \exists \bar{z} : E^+(\bar{x}, \bar{z}) \wedge E(\bar{z}, \bar{y})$$

$$E^*(\bar{x}, \bar{y}) : (\bar{x} = \bar{y}) \vee E^+(\bar{x}, \bar{y})$$

- The definitions of $e^+(\bar{x}, \bar{y})$ and $e^*(\bar{x}, \bar{y})$ are similar to $E^+(\bar{x}, \bar{y})$ and $E^*(\bar{x}, \bar{y})$.

🌐 The following formula identifies fulfilling atoms.

$$\begin{aligned} scf^E(\bar{x}) : E^+(\bar{x}, \bar{x}) \wedge \bigwedge_{p \mathbf{U} q \in CL(f)} (\chi_p \mathbf{U} q(\bar{x}) \rightarrow \\ \exists \bar{z} : E^*(\bar{x}, \bar{z}) \wedge \chi_q(\bar{z}) \wedge E^*(\bar{z}, \bar{x})) \end{aligned}$$