CISC/CMPE 422, CISC 835:
Formal Methods in Software Engineering

Juergen Dingel

Fall 2019

- Computation Tree Logic (CTL)
  - Syntax, semantics (Chapter 13.1)
- The CTL model checking algorithm (Chapter 13.2)

CTL Syntax

CTL formulas are defined by the following BNF

$$\varphi ::= ff \mid tt \mid p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid \mathbf{AX}\,\varphi \mid \mathbf{EX}\,\varphi \mid \mathbf{AG}\,\varphi \mid \mathbf{EG}\,\varphi \mid \mathbf{AF}\,\varphi \mid \mathbf{EF}\,\varphi \mid \mathbf{A}[\varphi_1\,\mathbf{U}\,\varphi_2] \mid \mathbf{E}[\varphi_1\,\mathbf{U}\,\varphi_2]$$

where p is an atomic proposition, that is, $p \in AP$.

CTL Semantics

$\mathbf{AX}\,\varphi$	"Along all paths, in the next state, φ holds"
$\mathbf{EX}\,\varphi$	"Along at least one path, in the next state, φ holds"
$\mathbf{AG}\,\varphi$	"Along all paths, in all future states, φ holds" "Along all paths, φ holds globally"
$\mathbf{EG}\,\varphi$	"Along at least one path, in all future states, φ holds" "Along at least one path, φ holds globally"
$\mathbf{AF}\,\varphi$	"Along all paths, in some future state, φ holds", or "Along all paths, φ holds eventually"
$\mathbf{EF}\,\varphi$	"Along at least one path, in some future state, φ holds", or "Along at least one path, φ holds eventually"
$\mathbf{A}[\varphi_1\,\mathbf{U}\,\varphi_2]$	"Along all paths, φ_1 holds at least until φ_2 does"
$\mathbf{E}[\varphi_1\,\mathbf{U}\,\varphi_2]$	"Along at least one path, φ_1 holds at least until φ_2 does"

CTL Semantics (Cont'd)

Formulas are interpreted over Kripke structures. Given a Kripke structure M, a state s, and a CTL formula φ, the satisfaction relation $(M,s) \models \varphi$ is defined as follows:

$(M,s) \models tt$

$(M,s) \models p$ if $p \in L(s)$

$(M,s) \models \neg\varphi_1$ if not $(M,s) \models \varphi_1$

$(M,s) \models \varphi_1 \wedge \varphi_2$ if $(M,s) \models \varphi_1$ and $(M,s) \models \varphi_2$

$(M,s) \models \varphi_1 \vee \varphi_2$ if $(M,s) \models \varphi_1$ or $(M,s) \models \varphi_2$

$(M,s) \models \varphi_1 \rightarrow \varphi_2$ if not $(M,s) \models \varphi_1$ or $(M,s) \models \varphi_2$

$(M,s) \models \mathbf{AX}\,\varphi$ if for all s' such that $R(s,s')$ we have $(M,s') \models \varphi$

$(M,s) \models \mathbf{EX}\,\varphi$ if for some s' such that $R(s,s')$ we have $(M,s') \models \varphi$

$(M,s) \models \mathbf{AG}\,\varphi$ if for all paths $s_1 s_2 s_3 \ldots$ in M such that $s = s_1$ we have $(M,s_i) \models \varphi$ for all $i \geq 1$

$(M,s) \models \mathbf{EG}\,\varphi$ if for some path $s_1 s_2 s_3 \ldots$ in M such that $s = s_1$ we have $(M,s_i) \models \varphi$ for all $i \geq 1$

CTL Semantics (Cont'd)

$$
\begin{array}{lll}
(M,s) & \models & \mathbf{AF}\ \varphi \text{ if for all paths } s_1 s_2 s_3 \ldots \text{ in } M \text{ such that } s = s_1 \\
 & & \text{there exists } i \geq 1 \text{ such that } (M,s_i) \models \varphi \\
(M,s) & \models & \mathbf{EF}\ \varphi \text{ if for some path } s_1 s_2 s_3 \ldots \text{ in } M \text{ such that } s = s_1 \\
 & & \text{there exists } i \geq 1 \text{ such that } (M,s_i) \models \varphi \\
(M,s) & \models & \mathbf{A}[\varphi_1\ \mathbf{U}\ \varphi_2] \text{ if for all paths } s_1 s_2 s_3 \ldots \text{ in } M \text{ such that } s = s_1 \\
 & & \text{there exists some } i \geq 1 \text{ such that } (M,s_i) \models \varphi_2, \text{ and} \\
 & & \text{for all } 1 \leq j < i, \text{ we have } (M,s_j) \models \varphi_1 \\
(M,s) & \models & \mathbf{E}[\varphi_1\ \mathbf{U}\ \varphi_2] \text{ if for some path } s_1 s_2 s_3 \ldots \text{ in } M \text{ such that } s = s_1 \\
 & & \text{there exists some } i \geq 1 \text{ such that } (M,s_i) \models \varphi_2, \text{ and} \\
 & & \text{for all } 1 \leq j < i, \text{ we have } (M,s_j) \models \varphi_1
\end{array}
$$

$$
\begin{array}{lll}
(M,s) & \models & tt \\
(M,s) & \models & p \text{ if } p \in L(s) \\
(M,s) & \models & \neg\varphi_1 \text{ if not } (M,s) \models \varphi_1 \\
(M,s) & \models & \varphi_1 \wedge \varphi_2 \text{ if } (M,s) \models \varphi_1 \text{ and } (M,s) \models \varphi_2 \\
(M,s) & \models & \varphi_1 \vee \varphi_2 \text{ if } (M,s) \models \varphi_1 \text{ or } (M,s) \models \varphi_2 \\
(M,s) & \models & \varphi_1 \rightarrow \varphi_2 \text{ if not } (M,s) \models \varphi_1 \text{ or } (M,s) \models \varphi_2 \\
(M,s) & \models & \mathbf{AX}\ \varphi \text{ if for all } s' \text{ such that } R(s,s') \text{ we have } (M,s') \models \varphi \\
(M,s) & \models & \mathbf{EX}\ \varphi \text{ if for some } s' \text{ such that } R(s,s') \text{ we have } (M,s') \models \varphi \\
(M,s) & \models & \mathbf{AG}\ \varphi \text{ if for all paths } s_1 s_2 s_3 \ldots \text{ in } M \text{ such that } s = s_1 \text{ we have} \\
 & & (M,s_i) \models \varphi \text{ for all } i \geq 1 \\
(M,s) & \models & \mathbf{EG}\ \varphi \text{ if for some path } s_1 s_2 s_3 \ldots \text{ in } M \text{ such that } s = s_1 \text{ we have} \\
 & & (M,s_i) \models \varphi \text{ for all } i \geq 1 \\
(M,s) & \models & \mathbf{AF}\ \varphi \text{ if for all paths } s_1 s_2 s_3 \ldots \text{ in } M \text{ such that } s = s_1 \\
 & & \text{there exists } i \geq 1 \text{ such that } (M,s_i) \models \varphi \\
(M,s) & \models & \mathbf{EF}\ \varphi \text{ if for some path } s_1 s_2 s_3 \ldots \text{ in } M \text{ such that } s = s_1 \\
 & & \text{there exists } i \geq 1 \text{ such that } (M,s_i) \models \varphi \\
(M,s) & \models & \mathbf{A}[\varphi_1\ \mathbf{U}\ \varphi_2] \text{ if for all paths } s_1 s_2 s_3 \ldots \text{ in } M \text{ such that } s = s_1 \\
 & & \text{there exists some } i \geq 1 \text{ such that } (M,s_i) \models \varphi_2, \text{ and} \\
 & & \text{for all } 1 \leq j < i, \text{ we have } (M,s_j) \models \varphi_1 \\
(M,s) & \models & \mathbf{E}[\varphi_1\ \mathbf{U}\ \varphi_2] \text{ if for some path } s_1 s_2 s_3 \ldots \text{ in } M \text{ such that } s = s_1 \\
 & & \text{there exists some } i \geq 1 \text{ such that } (M,s_i) \models \varphi_2, \text{ and} \\
 & & \text{for all } 1 \leq j < i, \text{ we have } (M,s_j) \models \varphi_1
\end{array}
$$

Adequacy (Chapter 13.1)

Theorem: The following set of connectives and operators is adequate for CTL:

$$\{\neg, \wedge, \mathbf{EX}, \mathbf{AF}, \mathbf{EU}\}$$

Proof:

$$
\begin{array}{rcl}
\varphi_1 \vee \varphi_2 & \leftrightarrow & \neg(\neg\varphi_1 \wedge \neg\varphi_2) \\
\varphi_1 \rightarrow \varphi_2 & \leftrightarrow & \neg\varphi_1 \vee \varphi_2) \\
\mathbf{EG}\ \varphi & \leftrightarrow & \neg\mathbf{AF}\ \neg\varphi \\
\mathbf{AX}\ \varphi & \leftrightarrow & \neg\mathbf{EX}\ \neg\varphi \\
\mathbf{EF}\ \varphi & \leftrightarrow & \mathbf{E}[tt\ \mathbf{U}\ \varphi] \\
\mathbf{AG}\ \varphi & \leftrightarrow & \neg\mathbf{EF}\ \neg\varphi \\
\mathbf{A}[\varphi_1\ \mathbf{U}\ \varphi_2] & \leftrightarrow & (\mathbf{AF}\ \varphi_2)\ \wedge\ \neg\mathbf{E}[\neg\varphi_2\ \mathbf{U}\ \neg\varphi_1 \wedge \varphi_2]
\end{array}
$$

Unwindings (Chapter 13.1)

The model checking algorithm will use the following equivalences:

$$
\begin{array}{rcl}
\mathbf{AG}\ \varphi & \leftrightarrow & \varphi \wedge \mathbf{AX}\ \mathbf{AG}\ \varphi \\
\mathbf{EG}\ \varphi & \leftrightarrow & \varphi \wedge \mathbf{EX}\ \mathbf{EG}\ \varphi \\
\mathbf{AF}\ \varphi & \leftrightarrow & \varphi \vee \mathbf{AX}\ \mathbf{AF}\ \varphi \\
\mathbf{EF}\ \varphi & \leftrightarrow & \varphi \vee \mathbf{EX}\ \mathbf{EF}\ \varphi \\
\mathbf{A}[\varphi\ \mathbf{U}\ \psi] & \leftrightarrow & \psi \vee (\varphi \wedge \mathbf{AX}\ \mathbf{A}[\varphi\ \mathbf{U}\ \psi]) \\
\mathbf{E}[\varphi\ \mathbf{U}\ \psi] & \leftrightarrow & \psi \vee (\varphi \wedge \mathbf{EX}\ \mathbf{E}[\varphi\ \mathbf{U}\ \psi])
\end{array}
$$

CTL Model Checking Algorithm (Chapter 13.2)

Input: Kripke structure $M = (S, S_0, R, L)$ and CTL formula φ

Output: "Yes", if $(M, s_0) \models \varphi$ for all initial states $s_0 \in S_0$. "No", otherwise.

Step 1: Preprocessing
Translate φ into an equivalent formula φ' that contains only the adequate connectives.

Step 2: Labeling
Label all states s in M with the subformulas φ'' of φ' (including φ') s.t. $(M, s) \models \varphi''$.

Step 3: Check that initial states are labeled with φ'
If all initial states of M are labeled with φ', then output "Yes". Otherwise, output "No".

We use the recursive procedure $SAT(\varphi)$ to implement Step 2.

Input: Kripke structure $M = (S, S_0, R, L)$ and CTL formula φ'

Output: For every state s in M and every subformula φ'' of φ', s is labeled with φ'' if and only if $(M, s) \models \varphi''$.

$SAT(\varphi')$ is defined as follows:

case φ' **of**

- p:
  if $p \in L(s)$, then label s with p
- $\neg\psi_1$:
  $SAT(\psi_1)$;
  if s not labeled with ψ_1, then label s with $\neg\psi_1$
- $\psi_1 \wedge \psi_2$:
  $SAT(\psi_1)$;
  $SAT(\psi_2)$;
  if s labeled with ψ_1 and psi_2, then label s with $\psi_1 \wedge \psi_2$
- **EX** ψ_1:
  $SAT(\psi_1)$;
  if s has at least on successor labeled with ψ_1, then label s with **EX** ψ_1
- **AF** ψ_1:
  $SAT(\psi_1)$;
  (a) If state s is labeled with ψ_1, then label s with **AF** ψ_1
  (b) If all successors of s are labeled with **AF** ψ_1, then label s with **AF** ψ_1
  (c) If step (b) changed the labeling, then go back to (b). Otherwise, stop
- **E**[ψ_1 **U** ψ_2]:
  $SAT(\psi_1)$;
  $SAT(\psi_2)$;
  (a) If state s is labeled with ψ_2, then label s with **E**[ψ_1 **U** ψ_2]
  (b) If state s is labeled with ψ_1 and has at least one successor labeled **E**[ψ_1 **U** ψ_2], then label s with **E**[ψ_1 **U** ψ_2],
  (c) If step (b) changed the labeling, then go back to (b). Otherwise, stop.

O(n * |S| *(|S|+|R|) where
n is #connectives in ϕ
With optimizations:
O(n * (|S| + |R|)

State Space Explosion Problem

- Factors influencing state space size
  - Number of variables
  - Number of different values variables can take on
  - Number of processes

```
-- Program P4.n: n processes counting up
MODULE main
VAR
          p1 : process P(1000);
          ...
          pn : process P(1000);

MODULE P(TO)
VAR
          x : 1..TO;
ASSIGN
          init(x):=1;
          next(x):= case
                  x<TO: x+1;
                  TRUE: x;
                esac;
```

Program	P4.1	P4.2	P4.3	P4.4	P4.5	P4.6
Number of reachable states	10^3	10^6	10^9	10^{12}	10^{15}	10^{18}
Time taken to compute (in secs)	5	8	22	63	281	13,115

Wrapping up

- Course summary
- Final exam

Queen's is Full of Surprises

Queen's is Full of Surprises

You → CISC/CMPE 422/835 → You'

The End