Behavior models and verification

*Lecture 5*

Jan Kofroň, František Plášil
State –based Temporal Logics

- State-based temporal logics
  - Kripke structure
  - LTL, CTL, CTL*, μ-calculus, ...

- Interpretation of “temporal”
  - Temporal ~ Time
    - Future on a path is reasoned about
  - Time: Linear vs. Branching
    - Linear (LTL, Linear Time Logic)
      - a single path is considered in isolation
    - Branching (CTL, Computation Tree Logic)
      - “forking” in a state is also considered – multiple paths

Jan Kofroň, František Plášil, Lecture 5
Model checking

- Model checking
  - general interpretation for state–based temporal logics:

- For a Kripke structure $M = (S, I, R, L)$ over $AP$ and a (state based) temporal logic formula $\varphi$ find the set of all states in $S$ that satisfy $\varphi$:
  - $X = \{ s \in S : M, s \models \varphi \}$
  - $M$ satisfies $\varphi$ if $I \subseteq X$
System satisfies the specification $\varphi$
System does not satisfy the specification \( \phi \)
Will see: Explicit vs. symbolic model checking

• Explicit model checking
  ▪ Each state of M is **explicitly** represented in memory as a labeled, directed graph, and checked

• Symbolic model checking
  ▪ Based on manipulation with **Boolean formulas**
  ▪ The algorithm operates on entire sets of states rather than on individual states
  ▪ Reduction of time and memory consumption
Did you know...?

Explicit model checking

- Each state of M is explicitly represented, labeled, directed graph, and checked

Symbolic model checking

- Based on manipulation with Boolean formulas
- The algorithm operates on entire model, not individual states
- Reduction of time and memory complexity

George Boole (1815 – 1864)

*English mathematician, philosopher and logician*
We will learn in this lecture

- CTL logic
- Expressiveness LTL vs. CTL
- Algorithm to check validity of CTL formula in Kripke structure $M$
  - Explicitly traversing the state space of $M$
  - i.e. an algorithm for CTL explicit model checking
- Complexity estimates
A CTL formula has one of the following forms:

- 0, 1, \( p \), \( \neg \varphi \), \( \varphi \land \psi \), \( \varphi \rightarrow \psi \), \( \varphi \lor \psi \)
  - \( p \) is an atomic formula, \( p \in AP \)
- \( AX \varphi \), \( EX \varphi \)
- \( AG \varphi \), \( EG \varphi \)
- \( AF \varphi \), \( EF \varphi \)
- \( A[\varphi U \psi ] \), \( E[\varphi U \psi ] \)
  - where \( \varphi \), \( \psi \) are CTL formulas
CTL semantics

- $M, s \models \varphi$ stands for “a state $s$ from Kripke structure $M$ satisfies a CTL formula $\varphi$”
  - $\models$ is defined by induction on the size of $\varphi$
    - Next slide(s)

- Interpretation of symbols
  - $A$ all infinite paths from $s$
  - $E$ exist an infinite paths from $s$
  - $G, F, X, U$ similar to LTL
Definition

- $M, s \models p \iff p \in L(s)$
- $M, s \models \neg \varphi_1 \iff \neg M, s \models \varphi_1$
- $M, s \models \varphi_1 \lor \varphi_2 \iff M, s \models \varphi_1 \text{ or } M, s \models \varphi_2$
- $M, s \models \varphi_1 \land \varphi_2 \iff M, s \models \varphi_1 \text{ and } M, s \models \varphi_2$
Definition

- $M, s \models \text{EX } \varphi_1 \iff$ there is a state $t$ and a transition $s \rightarrow t$ in $M$ s.t. $M, t \models \varphi_1$

- $M, s \models \text{AX } \varphi_1 \iff$ for every state $t$ in $M$ s.t. $s \rightarrow t$, $M, t \models \varphi_1$ holds
CTL semantics (cont.)

- **Definition**
  - $M, s \models EF \varphi_1 \iff$ there exists a state $t$ and a path from $s$ to $t$ (in $M$) s.t. $M, t \models \varphi_1$
  - $M, s \models AF \varphi_1 \iff$ on every infinite path (in $M$) beginning in $s$ there is a state $t$ s.t. $M, t \models \varphi_1$
  - $M, s \models EG \varphi_1 \iff$ there exists an infinite path $\pi = \pi_0 \rightarrow \pi_1 \rightarrow \pi_2 \ldots$ (in $M$) s.t. $s = \pi_0$ and for all $i \geq 0 \pi_i \models \varphi_1$ holds
  - $M, s \models AG \varphi_1 \iff$ for every infinite path $\pi = \pi_0 \rightarrow \pi_1 \rightarrow \pi_2 \ldots$ (in $M$) s.t. $s = \pi_0$, for all $i \geq 0 \pi_i \models \varphi_1$ holds
**Definition**

- $M, s \models E [\varphi_1 U \varphi_2] \iff$
  
  there exists an infinite path $\pi = \pi_0 \rightarrow \pi_1 \rightarrow \pi_2 \ldots$ (in $M$) s.t.
  $\pi_0 = s$ and there exists $i \geq 0$ s.t.:
  - $M, \pi_i \models \varphi_2$
  - for all $j$, $0 \leq j < i$, $M, \pi_j \models \varphi_1$ holds

- $M, s \models A [\varphi_1 U \varphi_2] \iff$

  for all infinite paths $\pi = \pi_0 \rightarrow \pi_1 \rightarrow \pi_2 \ldots$ (in $M$)
  s.t. $p_0 = s$, there exists $i \geq 0$ s.t.:
  - $M, \pi_i \models \varphi_2$
  - for all $j$, $0 \leq j < i$, $M, \pi_j \models \varphi_1$ holds
Kripke structure

Computational tree
$$AX \varphi_1$$

![Diagram of AX \varphi_1]

$\bullet = \varphi_1 \text{ holds}$

$$EX \varphi_1$$

![Diagram of EX \varphi_1]

$\bigcirc = \varphi_1 \text{ either holds or not}$
\[ AG \varphi_1 \]

\[ EG \varphi_1 \]

\[ AF \varphi_1 \]

\[ EF \varphi_1 \]

$\bullet = \varphi_1 \text{ holds}$

$\circ = \varphi_1 \text{ does not hold}$
\[ E[\varphi_1 \cup \varphi_2] = \varphi_1, \varphi_2 \text{ undefined} \]

\[ A[\varphi_1 \cup \varphi_2] = \varphi_1 \text{ holds (} \varphi_2 \text{ undefined)} \]

\[ = \varphi_2 \text{ holds (} \varphi_1 \text{ undefined)} \]

\[ = \varphi_1, \varphi_2 \text{ undefined} \]
Difference between CTL and LTL

- Relating CTL formulas to LTL and vice versa
  - $\text{AG EF } p$ is weaker than $\text{G F } p$
    
    ![Diagram 1](image1.png)
    
    $\neg \text{G F } p$
    $\sqrt{\text{AG EF } p}$
    Good for finding bugs...

  - $\text{AF AG } p$ is stronger than $\text{F G } p$
    
    ![Diagram 2](image2.png)
    
    $\sqrt{\text{FG } p}$
    $\neg \text{AF AG } p$
    Good for verifying invariants...

- As an aside: CTL formulas easier to verify
LTL vs. CTL

\[ M, s \models_{\text{LTL}} \text{FG } p \]

\[ \text{not } M, s \models_{\text{CTL}} \text{AF (AG } p) \]

- \( \circ \) = \( p \) holds
- \( \bigcirc \) = \( p \) does not hold
A property expressible in CTL but not in LTL

- **Fact:** There is no LTL formula $\varphi$ equivalent to the CTL formula $AG(EF p)$
- Note that $AG(EF p)$ is *not* the same as $G(F p)$ in LTL
  - Recall:

- Suppose that there is such a $\varphi$ (in LTL). Consider the following K.S.
• AG(EF p) is true in $s_0$
  $\implies \phi$ is true in $s_0$ as well
  i.e. $\phi$ is true on all paths that start in $s_0$
  $\implies$ therefore $\phi$ is true on the path that loops in $s_0$
  $\implies$ thus $\phi$ is true in $s'$ of the following Kripke structure

$\implies$ thus AG(EF p) would have to be true in $s'$
$\implies$ contradiction!
The LTL formula $FG\ p$ is not equivalent to any CTL formula.

In particular, it is not equivalent to the CTL formula $AF\ (AG\ p)$.
The LTL formula $FG \ p$ is not equivalent to CTL formula $AF(AG \ p)$

To prove this, we have to find a Kripke structure $M$ and a state $s$ in $M$ s.t.:

- either
  - $M, s \models_{LTL} FG \ p$
  - and not $M, s \models_{CTL} AF (AG \ p)$

- or
  - $M, s \models_{CTL} AF (AG \ p)$
  - and not $M, s \models_{LTL} FG \ p$
LTL ver. CTL

\[ M, s \models_{\text{LTL}} \text{FG} \ p \]

not \[ M, s \models_{\text{CTL}} \text{AF} \ (\text{AG} \ p) \]

- = p holds
- = p does not hold
Linear and Branching time logics are incomparable
LTL vers. CTL: Complexity

- Model checking
  - Does satisfy $\varphi$?
  - $|M| = n$, $|\varphi| = m$

- Time complexity
  - CTL: $O(nm)$
  - LTL: $O(n2^m)$
    - also PSPACE complete => polynomial space needed

- Conclusion
  - Linear complexity in $|M|$:
    - both CTL and LTL
  - LTL exponential in $|\varphi|$
    - However, typically $m << n$
(EG E[p U q]) & EX r

EG E[p U q]

E[p U q]

p

q

EX r

r
Explicit CTL model checking algorithm

For every state $s$ in $S$, the algorithm labels $s$ with all subformulas of $\phi$ which are true in $s$

- $label(s)$ – the set of labels associated with $s$
- initially, $label(s) = L(s)$
- then, the algorithm goes through a series of stages
  - during the $i$-th stage, the subformulas with $i$-1 nested operators are processed
  - when a subformula is processed, it is added to the labeling of each state $s$ in which it is true (i.e. $label(s)$ is updated)

- Once the algorithm terminates, we will have that

\[ M, s \models \phi \iff \phi \in label(s) \]
Explicit CTL model checking algorithm

\[(\text{EG } E[p \lor q]) \land \text{EX } r\]

\[E[p \lor q]\]

\[\text{EX } r\]\n
\[p\]

\[q\]

\[r\]
Explicit CTL model checking algorithm

\[(EG \ E[p \ U \ q]) \land EX \ r\]
Explicit CTL model checking algorithm

\[(\text{EG } E[p \cup q]) \land \text{EX } r\]

\[E[p \cup q]\]

\[E[p \cup q]\]

\[E[p \cup q]\]

\[p\]

\[q\]

\[r\]
Explicit CTL model checking algorithm

\( (EG \ E[p \ U \ q]) \ & \ EX \ r \)

\( EG \ E[p \ U \ q] \)

\( E[p \ U \ q] \)

\( p \quad q \quad r \)

\( EX \ r \)

\( p \)

\( E[p \ U \ q] \)

\( p \)

\( E[p \ U \ q] \)

\( q \)

\( E[p \ U \ q] \)

\( EX \ r \)

\( p \)

\( E[p \ U \ q] \)

\( r \)

\( EX \ r \)
Explicit CTL model checking algorithm

Jan Kofroň, František Plášil, Lecture 5
Explicit CTL model checking algorithm

\[(\text{EG E}[p \cup q]) \& \text{EX } r\]

\[\text{EG E}[p \cup q]\]

\[\text{E}[p \cup q]\]

\[\text{EX } r\]

\[p \quad q \quad r\]
Any CTL formula can be expressed in the terms of $\neg$, 
& , EX, EU, EG

Handling $\neg \phi_1$, $\phi_1 \& \phi_2$, EX $\phi_1$ during a stage of the algorithm is trivial
All operators in terms of EX, EG, EU

- $AX \varphi_1 = \neg EX (\neg \varphi_1)$
- $EF \varphi_1 = E[1 \ U \varphi_1]$
- $AG \varphi_1 = \neg EF (\neg \varphi_1)$
- $AF \varphi_1 = \neg EG (\neg \varphi_1)$
- $A[\varphi_1 \ U \varphi_2] = \neg EG (\neg \varphi_2) \ & \ \& \ \& \ \neg E[\neg \varphi_2 \ U (\neg \varphi_1 \ & \ \& \ \neg \varphi_2)]$

- Thus we need just to show how to traverse the state space for EU and EG
  - EX is trivial
Handling $E[\varphi_1 U \varphi_2]$

procedure CheckEU($\varphi_1$, $\varphi_2$)

\[ T := \{s : \varphi_2 \in \text{label}(s)\}; \]

for all $s \in T$
do

\[ \text{label}(s) := \text{label}(s) \cup \{E[\varphi_1 U \varphi_2]\}; \]

end for all

while $T \neq \{\}$
do

\[ \text{choose } s \in T; \]

\[ T := T \setminus \{s\}; \]

for all $t$ such that $R(t,s)$
do

\[ \text{if } E[\varphi_1 U \varphi_2] \notin \text{label}(t) \]

\[ \text{and } \varphi_1 \in \text{label}(t) \text{ then} \]

\[ \text{label}(t) := \text{label}(t) \cup \{E[\varphi_1 U \varphi_2]\}; \]

\[ T := T \cup \{t\}; \]

end if

end for all

end while

end procedure
Handling $E[\phi_1 \cup \phi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling \( E[\varphi_1 \cup \varphi_2] \)
Handling $E[\varphi_1 U \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 U \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 U \varphi_2]$
Handling $E[\varphi_1 U \varphi_2]$
Handling $E[\varphi_1 U \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \lor \varphi_2]$
Handling $E[\varphi_1 U \varphi_2]$
Handling $E[\varphi_1 U \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling EG $\varphi_1$

- Based on decomposition of the graph into nontrivial strongly connected components
- A *strongly connected component* (SCC) $C$ is a *maximal* subgraph such that every node in $C$ is reachable from every other node in $C$ along a directed path entirely contained within $C$
- $C$ is *nontrivial* iff either it has more than one node or it contains one node with a self-loop
  - infinite path
Handling EG $\varphi_1$

- $M' = (S', R', L')$
- $S' = \{ s \in S : M, s \models \varphi_1 \}$
- $R' = R \mid_{S' \times S'}$
- $L' = L \mid_{S'}$
**Lemma:** $M, s \models EG \varphi_1$ iff both the following conditions are satisfied:

- $s \in S'$
- There exists a path in $M'$ that leads from $s$ to some node $t$ in a nontrivial strongly connected component $C$ of the graph $(S', R')$
Handling $\text{EG } \varphi_1$

- Construct the restricted Kripke structure $M' = (S', R', L')$
- Partition the graph $(S', R')$ into strongly connected components (*Tarjan algorithm*)
- Find those states that belong to a nontrivial component
- Work backward (using converse of $R'$)
  - find all the states that can be reached by a path (converse of $R'$ !) in which each state is labeled with $\varphi_1$
Handling EG $\varphi_1$
Handling EG $\varphi_1$

Construction of $(S', R')$
Identification of nontrivial strongly connected components
Handling $\text{EG } \varphi_1$

$\text{EG } \varphi_1$ holds

\[
\text{ = EG } \varphi_1\text{ holds}
\]
procedure CheckEG($\varphi_1$)
S' = \{s : $\varphi_1 \in \text{label}(s)\};
SCC = \{C : C \text{ is a nontrivial SCC of } S'\};
T := \bigcup_{C \in \text{SCC}} \{s : s \in C\};
for all s \in T do
    \text{label}(s) := \text{label}(s) \cup \{\text{EG } \varphi_1\};
end for all
while T != {} do
    choose s \in T;
    T := T \setminus \{s\};
    for all t such that t \in S' and R(t,s) do
        if EG $\varphi_1 \notin \text{label}(t)$ then
            \text{label}(t) := \text{label}(t) \cup \{\text{EG } \varphi_1\};
            T := T \cup \{t\};
        end if
    end for all
end while
end procedure
Explicit CTL model checking algorithm

- Check EU
  - $O(|S| + |R|)$
- Check EG
  - $O(|S| + |R|)$
  - Partitioning using Tarjan algorithm: $O(|S'| + |R'|)$
- $\varphi$ has at most $|\varphi|$ different subformulas

- Time complexity: $O(|\varphi| \times (|S| + |R|))$