Agenda

- CTL logic
- CTL model checking

- Computation Tree Logic
  - A formula can reason about many executions in a computation tree at once
CTL Model checking (explicit)

- Markov chains
- Timed automata
- Labelled transition system
- Kripke structure

Model

- open
- close
- heat
- start
- empty

Model checker

Property specification

AG(start → AF heat)

Property satisfied

Property violated

Error report

Jan Kofroň, František Plášil, Lecture 4
The task of explicit CTL model checking reads

- 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 $\varphi$:

  $$X = \{ s \in S : M, s \models \varphi \}$$

- “Explicit” : Each state of $M$ is explicitly represented in memory in a labeled, directed graph, and checked
KS satisfies the specification $\varphi$ in green states

KS in the initial states $I$ satisfies $\varphi$
KS does not satisfy the specification (formula) $\phi$ in the initial states $I$. 

Jan Kofroň, František Plášil, Lecture 4
CTL basic idea

- **Computational Tree Logic**
  - Considers property of the states along the paths in a computational (sub)tree
  - As opposed to LTL which, given a state s, considers property of each path (starting in s) separately
Consider

- Kripke str. with initial state, property $\varphi$

Assume

- Syntax note: $A$ – All paths  $E$ – Exists a path

- $\varphi = AG \ EF \ p$
  - In all (A) paths starting in and for any (G) state $s$ on them it holds that there exists (E) a path starting with $s$ such that it contains (F) a state where $p$ holds
  - true
- $\varphi = AG \ EX \ p$
  - true
- $\varphi = AG \ AX \ p$
  - false
- $\varphi = AG \ EX \ AGp$
  - True
Consider

- Kripke str. with initial state \( \odot \), property \( \varphi \)

LTL

- Assume \( \varphi = GFp \)
  - In all infinite paths starting in \( \odot \), \( p \) globally holds in the future
    - Not true, since
    - Implies computational tree with the path \( \uparrow \)
A CTL formula has one of the following forms:

- A, 1, p, ¬φ, φ & ψ, φ ⇒ ψ, φ ∨ ψ
  - p is an atomic formula, p ∈ AP
- AX φ, EX φ
- AG φ, EG φ
- AF φ, EF φ
- A[φ U ψ ], E[φ U ψ ]
  - where φ, ψ are CTL formulas

A – All paths  E – Exists a path    (two quantifiers)
X – neXt     G – Globally     F – Future     U - Until
CTL semantics

- \( M, s \models \varphi \) stands for “a state \( s \) from the Kripke structure \( M \) satisfies a CTL formula \( \varphi \)”

- \( \models \) is defined by induction on the size of \( \varphi \)

**Definition**

- \( M, s \models p \) \iff p \in L(s)
- \( M, s \models \neg \varphi_1 \) \iff not \( M, s \models \varphi_1 \)
- \( M, s \models \varphi_1 \lor \varphi_2 \) \iff \( M, s \models \varphi_1 \) or \( M, s \models \varphi_2 \)
- \( M, s \models \varphi_1 \land \varphi_2 \) \iff \( M, s \models \varphi_1 \) and \( M, s \models \varphi_2 \)
Definition (cont.)

- $M, s \models \text{EX } \varphi_1 \iff$ 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
Definition (cont.)

- $M, s \models EF \varphi_1 \iff$ 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
CTL semantics

Definition (cont.)

- **M, s ⊨ E [ψ₁ U ψ₂]**
  
  there exists an infinite path $π = π₀ → π₁ → π₂ ...$ (in M) s.t. $π₀ = s$ and there exists $i \geq 0$ s.t.:
  - $M, πᵢ ⊨ ψ₂$
  - for all $j$, $0 \leq j < i$, $M, πⱼ ⊨ ψ₁$ holds

- **M, s ⊨ A [ψ₁ U ψ₂]**
  
  for all infinite paths $π = π₀ → π₁ → π₂ ...$ (in M) s.t. $π₀ = s$, there exists $i \geq 0$ s.t.:
  - $M, πᵢ ⊨ ψ₂$
  - for all $j$, $0 \leq j < i$, $M, πⱼ ⊨ ψ₁$ holds
Kripke structure

Computational tree
\(AX \varphi_1\) holds

\[\bullet = \varphi_1 \text{ holds}\]

\(EX \varphi_1\) either holds or not

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

EG $\varphi_1$

AF $\varphi_1$

EF $\varphi_1$

\[ \bullet = \varphi_1 \text{ holds} \]

\[ \bigcirc = \varphi_1 \text{ undefined} \]
\[ E[\varphi_1 \cup \varphi_2] = \varphi_1, \text{ undefined} \]

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

- \( \bullet \) = \( \varphi_1 \) holds (\( \varphi_2 \) undefined)
- \( \circ \) = \( \varphi_2 \) holds (\( \varphi_1 \) undefined)
- \( \circ \circ \) = \( \varphi_1, \varphi_2 \) undefined
\( E[\varphi_1 \cup \varphi_2] \)

\[ \begin{array}{c}
\varnothing \quad = \quad \varphi_1 \text{ holds (} \varphi_2 \text{ undefined}) \\
\varnothing \quad = \quad \varphi_2 \text{ holds (} \varphi_1 \text{ undefined}) \\
\varnothing \quad = \quad \varphi_1, \varphi_2 \text{ undefined}
\end{array} \]

\( A[\varphi_1 \cup \varphi_2] \)
Think of CTL formulas as approximations to LTL formulas

- $\text{AG EF } p$ is weaker than $\text{G F } p$

- $\text{AF AG } p$ is stronger than $\text{F G } p$
Think of CTL formulas as approximations to LTL formulas

- $\text{AG EF } p$ is **weaker** than $\text{G F } p$

- $\text{AF AG } p$ is **stronger** than $\text{F G } p$
Practicality perspective

- AG EF \( p \) is weaker than \( G F \ p \)

![Diagram of AG EF p](image1)

Good for finding bugs... EF \( p \) “exists” by CTL

- AF AG \( p \) is stronger than \( F G \ p \)

![Diagram of AF AG p](image2)

Good for verifying... FG \( p \) “invariant” by LTL

- CTL formulas easier to verify
A property expressible in CTL but not in LTL

Fact: There is no LTL formula $\phi$ equivalent to the CTL formula $AG(EF p)$

Note that $AG(EF p)$ is not the same as $G(F p)$ in LTL

Suppose that there is such a $\phi$ (in LTL). Consider the following K.S.
CTL vers. LTL (cont.)

- \( \text{AG(EF } p) \) is true in \( s_0 \)
  \[ \rightarrow \varphi \text{ is true in } s_0 \text{ as well} \]
  - i.e. \( \varphi \) is true on all paths that start in \( s_0 \)
  \[ \rightarrow \text{therefore } \varphi \text{ is true on the path that loops in } s_0 \]
  \[ \rightarrow \text{thus } \varphi \text{ is true in } s' \text{ of the following Kripke structure} \]

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

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

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 $M, s \models_{CTL} AF(AF\ p)$

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

M, s ⊨_{LTL} FG p
not M, s ⊨_{CTL} AF (AG p)

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

• Model checking of $M = (S,R,L)$
  - Does $M$ satisfy $\phi$?
    - $|M| = |S| + |R|$
    - $|\Phi| = \text{number of subformulas of } \Phi$

• Time complexity
  - CTL: $O(|M| \cdot |\Phi|)$
  - LTL: $O(|M| \cdot 2^{|\Phi|})$ (PSPACE complete)

• Conclusion
  - Linear complexity in $|M|$
  - LTL exponential in $|\phi|$
    - However, typically $|\phi| \ll |M|$
Back to CTL m.c.: Formula parse tree

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

Diagram:

- \(\text{EG E}[p \lor q]\)
  - \(E[p \lor q]\)
    - \(p\)
    - \(q\)
  - EX \(r\)
    - \(r\)
Explicit CTL model checking algorithm

- For every state $s$ in $S$, the algorithm labels $s$ with all subformulas of $\varphi$ which are true in $s$
  - $\text{label}(s)$ – the set of labels associated with $s$
  - initially, $\text{label}(s) = L(s)$
- 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. $\text{label}(s)$ is updated)
- Once the algorithm terminates, we will have

$$M, s \models \varphi \iff \varphi \in \text{label}(s)$$
Explicit CTL model checking algorithm

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

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

Jan Kofroň, František Plášil, Lecture 4
Explicit CTL model checking algorithm

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

\[EG E[p \cup q]\]

\[E[p \cup q]\]

\[EX r\]
Explicit CTL model checking algorithm

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

$\text{EG E}[p \text{ U q}]$

$\text{E}[p \text{ U q}]$

$\text{EX } r$

$p$

$q$

$r$

$p$

$q$

$r$

$p$

$q$

$r$
Explicit CTL model checking algorithm

\[(\text{EG } \text{E}[p \lor q]) \land \text{EX } r\]
Explicit CTL model checking algorithm

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

• Any CTL formula can be expressed in the terms of $\neg$, 
  $\&$, EX, EU, EG

• Handling $\neg \varphi_1$, $\varphi_1 \& \varphi_2$, EX $\varphi_1$ during a stage of the
  algorithm is trivial
All operators in terms of EX, EG, EU

- \( AX \varphi_1 = \neg EX (\neg \varphi_1) \)
- \( EF \varphi_1 = E[1 \cup \varphi_1] \)
- \( AG \varphi_1 = \neg EF (\neg \varphi_1) \)
- \( AF \varphi_1 = \neg EG (\neg \varphi_1) \)
- \( A[\varphi_1 \cup \varphi_2] = \neg EG (\neg \varphi_2) \land \neg E[\neg \varphi_2 \cup (\neg \varphi_1 \land \neg \varphi_2)] \)
procedure CheckEU(ϕ₁, ϕ₂)
    T := {s : ϕ₂ ∈ label(s)};
    for all s ∈ T do
        label(s) := label(s) ∪ {E[ϕ₁ U ϕ₂]};
    end for all
    while T != {}
        choose s ∈ T;
        T := T \ {s};
        for all t such that R(t,s) do
            if E[ϕ₁ U ϕ₂] ∉ label(t)
                and ϕ₁ ∈ label(t) then
                label(t) := label(t) ∪ {E[ϕ₁ U ϕ₂]};
                T := T ∪ {t};
            end if
        end for all
    end while
end procedure
Handling $E[\varphi_1 \cup \varphi_2]$ – example: fragment of $M$
Handling $E[\varphi_1 U \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 U \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
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 $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\phi_1 \cup \phi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
Handling $E[\varphi_1 \cup \varphi_2]$
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
- 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 $\varphi_1$

$M:$

- $\bullet = \varphi_1 \text{ holds}$
- $\circ = \varphi_1 \text{ does not hold}$
Handling $\varphi_1$

$M'$:

Construction of $(S', R')$
Identification of nontrivial strongly connected components SCC by Tarjan algorithm (not detailed here)
Handling $EG \varphi_1$
procedure CheckEG(φ₁)
    \[ S' = \{ s : \phi_1 \in \text{label}(s) \}; \]
    \[ \text{SCC} = \{ C : C \text{ is a nontrivial SCC of } S' \}; \]
    \[ T := \bigcup_{C \in \text{SCC}} \{ s : s \in C \}; \]
    for all \( s \in T \) do
        label(s) := label(s) \cup \{ \text{EG } \phi_1 \};
    end for all
    while \( T \neq \{ \} \)
        choose \( s \in T \);
        \( T := T \setminus \{ s \}; \)
        for all \( t \) such that \( t \in S' \) and \( R(t,s) \) do
            if \( \text{EG } \phi_1 \not\in \text{label}(t) \) then
                label(t) := label(t) \cup \{ \text{EG } \phi_1 \};
                \( T := T \cup \{ t \}; \)
            end if
        end for all
    end while
end procedure
Explicit CTL model checking algorithm

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

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