Temporal logics

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

- **Action-based temporal logics**
  - Labeled transition system
  - Hennessy-Milner logic, Action-based EF logic, ...
Recall: Model checking

- 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 $\varphi$:
  
  $X = \{ s \in S : M, s \models \varphi \}$
System satisfies the specification
System does not satisfy the specification
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 by labeled, directed graph, and checked

Symbolic model checking
- Based on manipulation with Boolean functions
- The algorithm operates on entire set of states at once
- Reduction of time and memory occupation

George Boole (1815 – 1864)
- English mathematician, philosopher and logician
A CTL formula has one of the following forms:

- 0, 1, p, \( \neg \phi \), \( \phi \land \psi \), \( \phi \Rightarrow \psi \), \( \phi \lor \psi \)
  - \( p \) is an atomic formula, \( p \in AP \)
- \( AX \phi \), \( EX \phi \)
- \( AG \phi \), \( EG \phi \)
- \( AF \phi \), \( EF \phi \)
- \( A[\phi U \psi ] \), \( E[\phi U \psi ] \)
  - where \( \phi \), \( \psi \) 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 $\varphi$

Definition

- $M, s \models p \iff p \in L(s)$
- $M, s \models \neg \varphi_1 \iff \text{not } M, s \models \varphi_1$
- $M, s \models \varphi_1 \lor \varphi_2 \iff M, s \models \varphi_1 \text{ or } M, s \models \varphi_2$
- $M, s \models \varphi_1 \land \varphi_2 \iff M, s \models \varphi_1 \text{ and } M, s \models \varphi_2$
Definition (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
**CTL semantics**

- **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 $M$) starting 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 $\pi = \pi_0 \to \pi_1 \to \pi_2 \ldots$ (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 $\pi = \pi_0 \to \pi_1 \to \pi_2 \ldots$ (in $M$) s.t. $s = \pi_0$, for all $i \geq 0$ $\pi_i \models \varphi_1$ holds
CTL semantics

Definition (cont.)

- \( M, s \models E [\varphi_1 U \varphi_2] \iff \)
  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. \( \pi_0 = s \), 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
$AX \varphi_1$

$\bullet = \varphi_1$ holds

$EX \varphi_1$

$\bigcirc = \varphi_1$ either holds or not
AG $\varphi_1 = \varphi_1$ holds

EG $\varphi_1$

AF $\varphi_1$

EF $\varphi_1$
\[ E[\varphi_1 \cup \varphi_2] \]

\[ A[\varphi_1 \cup \varphi_2] \]

- \( \bullet \) = \( \varphi_1 \) holds (\( \varphi_2 \) undefined)
- \( \circ \) = \( \varphi_2 \) holds (\( \varphi_1 \) undefined)
- \( \bigcirc \) = \( \varphi_1, \varphi_2 \) undefined
Difference between CTL and LTL

- Think of CTL formulas as approximations to LTL
  - \( \text{AG EF } p \) is **weaker** than \( \text{G F } p \)

![Diagram 1]

- \( \text{AF AG } p \) is **stronger** than \( \text{F G } p \)

![Diagram 2]
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
- Suppose that there is such a $\varphi$ (in LTL). Consider the following K.S.
• $\text{AG(EF } p\text{)}$ is true in $s_0$
  $\Rightarrow \varphi$ is true in $s_0$ as well
  - i.e. $\varphi$ is true on all paths that start in $s_0$
  $\Rightarrow$ therefore $\varphi$ is true on the path that loops in $s_0$
  $\Rightarrow$ thus $\varphi$ is true in $s'$ of the following Kripke structure

$\Rightarrow$ thus $\text{AG(EF } p\text{)}$ is true in $s'$
$\Rightarrow$ 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 $\text{FG } p$ is not equivalent to CTL formula $\text{AF(AG p)}$

- To prove this, we have to find a Kripke structure $M$ and a state $s$ in $M$ s.t.:
  - either
    - $M, s \models_{\text{LTL}} \text{FG } p$
    - and not $M, s \models_{\text{CTL}} \text{AF (AG p)}$
  - or
    - $M, s \models_{\text{CTL}} \text{AF (AG p)}$
    - and not $M, s \models_{\text{LTL}} \text{FG } p$
LTL ver. CTL

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

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

= \( p \) holds

= \( p \) does not hold
Linear and Branching time logics are incomparable
LTL vers. CTL: Complexity

• Model checking
  ▪ Does T satisfy $\phi$?
  ▪ $|T| = n$, $|\phi| = m$

• Time complexity
  ▪ CTL: $O(nm)$
  ▪ LTL: $O(n2^m)$ (PSPACE complete)

• Conclusion
  ▪ Linear complexity in $|T|$
  ▪ LTL exponential in $|\phi|$
    ▪ However, typically $m << n$
Back to CTL: formula derivation tree

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

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

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

\[p\]

\[q\]

\[\text{EX } r\]

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

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

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

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

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

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

\[E[p \cup q]\]

\[p \quad q\]

\[p \quad r\]

\[p \quad E[p \cup q]\]

\[q \quad E[p \cup q]\]

\[r\]
Explicit CTL model checking algorithm

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

\[E[p \lor q]\]

\[E[p \lor q]\]

\[E[p \lor q]\]

\[p \quad q\]

\[\text{EX } r\]

\[p \quad q\]

\[\text{EX } r\]

\[\text{EX } r\]
Explicit CTL model checking algorithm

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

EG E[p U q]

E[p U q]

EX r

p q r

EG E[p U q]

EG E[p U q]

EG E[p U q]

EG E[p U q]

EG E[p U q]
Explicit CTL model checking algorithm

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

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

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

- \( AX \varphi_1 = \neg \text{EX} (\neg \varphi_1) \)
- \( EF \varphi_1 = \text{E}[1 \cup \varphi_1] \)
- \( AG \varphi_1 = \neg \text{EF} (\neg \varphi_1) \)
- \( AF \varphi_1 = \neg \text{EG} (\neg \varphi_1) \)
- \( A[\varphi_1 \cup \varphi_2] = \neg \text{EG} (\neg \varphi_2) \) &
  \( \& \neg \text{E}[\neg \varphi_2 \cup (\neg \varphi_1 \& \neg \varphi_2)] \)
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 != {} do
        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 $E[\varphi_1 \cup \varphi_2]$
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 $E[\varphi_1 \cup \varphi_2]$
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 $E[\phi_1 \cup \phi_2]$
Handling $E[\varphi_1 U \varphi_2]$
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 $\text{EG } \varphi_1$

$\varphi_1$ holds

$\varphi_1$ does not hold
Handling EG $\varphi_1$

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

$= EG \varphi_1 \text{ holds}$
procedure CheckEG(φ₁)
    S' = {s : φ₁ ∈ label(s)};
    SCC = {C : C is a nontrivial SCC of S'};
    T := ∪_{C∈SCC} {s : s ∈ C};
    for all s ∈ T do
        label(s) := label(s) ∪ {EG φ₁};
    end for all
    while T != {}
        choose s ∈ T;
        T := T \ {s};
        for all t such that t ∈ S' and R(t,s) do
            if EG φ₁ ∉ label(t) then
                label(t) := label(t) ∪ {EG φ₁};
                T := T ∪ {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: \( O(|\varphi| \times (|S| + |R|)) \)