CSC 410

AZADEH FARZAN

FALL 2019
Model Checking
First, a detour ...
LTL Expansion Laws

\[
\begin{align*}
\varphi \mathbin{U} \psi & \equiv \psi \lor (\varphi \land \Box (\varphi \mathbin{U} \psi)) \\
\Diamond \psi & \equiv \psi \lor \Box \Diamond \psi \\
\Box \psi & \equiv \psi \land \Box \Box \psi
\end{align*}
\]

Lemma. Until is the least solution to the expansion law.

The following equation has many solutions:

\[
X = \psi \lor (\phi \land \varnothing X)
\]

Until is the smallest set that satisfies this equation.

Note that we are using the notions of sets (of paths) and formulas interchangeably, by referring to the set of paths that satisfy a given formula.
For LTL, you would need to know/learn about automata on infinite words ....
For CTL, we can learn about algorithms directly ...
Last time ...
Computational Tree Logic (CTL)

\[\begin{array}{ccc}
{s_0} & {s_1} & {s_2} \\
\{x \neq 0\} & {x = 0}\{x = 1, x \neq 0\}
\end{array}\]

\[\begin{array}{c}
(s_0, 0) \\
(s_1, 1) \\
(s_2, 2) \quad (s_3, 2) \\
(s_3, 3) \\
(s_2, 4) \quad (s_3, 4) \quad (s_3, 4) \quad (s_2, 4) \quad (s_3, 4)
\end{array}\]
### CTL Syntax

#### State Formula

\[ \Phi ::= \text{true} \mid a \mid \Phi_1 \land \Phi_2 \mid \neg \Phi \mid \exists \varphi \mid \forall \varphi \]

#### Path Formula

\[ \varphi ::= \bigcirc \Phi \mid \Phi_1 \lor \Phi_2 \]
More Examples

$\exists \Diamond \text{black}$

$\exists \square \text{black}$

$\forall \Diamond \text{black}$

$\forall \square \text{black}$

$\exists (\text{gray} \cup \text{black})$

$\forall (\text{gray} \cup \text{black})$
CTL for LTSs

\[ \exists \Box a \]

\[ \exists \Diamond a \]

\[ \exists (\exists \Box a) \]

\[ \exists (a \cup (\neg a \land \forall (\neg a \cup b))) \]
And, now ...
**CTL Expansion Laws**

\[
\begin{align*}
\exists (\Phi \cup \Psi) &\equiv \Psi \lor (\Phi \land \exists \circ \exists (\Phi \cup \Psi)) \\
\exists \diamond \Phi &\equiv \Phi \lor \exists \circ \exists \diamond \Phi \\
\exists \square \Phi &\equiv \Phi \land \exists \circ \exists \square \Phi
\end{align*}
\]

**Lemma.** Until is the least solution to the expansion law.

The following equation has many solutions:

\[
F \equiv \Psi \lor (\Phi \land \exists \circ F)
\]

Until is the smallest set that satisfies this equation.
CTL Model Checking
Idea

\[(98737 \odot 593) \oplus (\ominus 120)\]
Let \( \Phi \) be a fresh atomic proposition. The indicated formulae are fresh atomic propositions, i.e.,

\[
\Phi = \exists \bigcirc a \land \exists (b \lor \exists \Box \neg c)
\]

Consider the following state formula over a transition system without terminal states. For all \( \Psi' \) and \( \Psi'' \), the set \( \exists \circ \Psi \) is treated, they can be replaced by the atomic proposition \( a \). The formula that is to be treated for the root node simply thus is:

\[
\begin{aligned}
&\exists \bigcirc S, \text{respectively, such that} \\
&\exists \circ \exists \bullet \wedge \exists \bigcirc \neg \Psi
\end{aligned}
\]

The above procedure thus is completed. Using \( \exists \circ \Psi \), \( \exists \bullet S \), and \( \exists \bigcirc \neg \Psi \), while the computation now continues with determining \( \exists \circ \Psi \).
Let’s invent the algorithm together ....
You can use a normal form like ENF (existential normal form) to have a short description ...
Recursive Rules for ENF

(a) \( \text{Sat}(\text{true}) = S \),

(b) \( \text{Sat}(a) = \{ s \in S \mid a \in L(s) \} \), for any \( a \in \text{AP} \),

(c) \( \text{Sat}(\Phi \land \Psi) = \text{Sat}(\Phi) \cap \text{Sat}(\Psi) \),

(d) \( \text{Sat}(\neg \Phi) = S \setminus \text{Sat}(\Phi) \),

(e) \( \text{Sat}(\exists \Diamond \Phi) = \{ s \in S \mid \text{Post}(s) \cap \text{Sat}(\Phi) \neq \emptyset \} \),

(f) \( \text{Sat}(\exists(\Phi \cup \Psi)) \) is the smallest subset \( T \) of \( S \), such that

\[ 1 \) \( \text{Sat}(\Psi) \subseteq T \) and \( 2 \) \( s \in \text{Sat}(\Phi) \) and \( \text{Post}(s) \cap T \neq \emptyset \) implies \( s \in T \),

(g) \( \text{Sat}(\exists \Box \Phi) \) is the largest subset \( T \) of \( S \), such that

\[ 3 \) \( T \subseteq \text{Sat}(\Phi) \) and \( 4 \) \( s \in T \) implies \( \text{Post}(s) \cap T \neq \emptyset \).
Final words on Model Checking ...
Back to LTL
Fairness

Ticket Lock from Unix Kernel

```
global nat s, t
local nat m
while(true):
    m=t++ // Acquire a ticket
while(m>s):
    // Busy wait
    skip
s++ // Exit critical
```

![Diagram of the ticket lock protocol](image)

The overall contribution of this paper is a formal foundation of proving liveness properties in infinite-state programs with a parameterized number of threads. The protocol, pictured in Figure 1, is an idealized version of ticket mutual exclusion and operates as follows: First, the thread acquires a ticket and increments the ticket number. Second, the thread waits for the ticket number to reach its ticket value, and then enters its critical section. After exiting the critical section, the next thread in the sequence enters. However, it is surprisingly difficult to come up with a formal correctness argument manually, let alone automatically.
unconditional LTL fairness:

\[ \square \diamond \psi \]

every process gets its turn infinitely often.

strong LTL fairness:

\[ \square \diamond \phi \rightarrow \square \diamond \psi \]

every process that is enabled infinitely often gets its turn infinitely often.

weak LTL fairness:

\[ \diamond \square \phi \rightarrow \square \diamond \psi \]

every process that is continuously enabled from a certain time instant on gets its turn infinitely often.

Fair satisfaction relation: \[ TS \models \mathcal{F} \phi \]
What is not covered ...
Scaling Model Checking

☐ Symbolic Model Checking

☐ Binary Decision Diagrams (BDDs)

☑ SAT-based techniques

☐ Bounded Model Checking

☑ Any forward reachability algorithm with a fixed depth.

☐ Partial Order Reduction

☐ Symmetry Reduction

☐ Abstraction