Software Verification
Exercise class:
Model Checking

Carlo A. Furia
Recap of definitions and results
Finite State Automata: Syntax

Def. Nondeterministic Finite State Automaton (FSA): a tuple $[\Sigma, S, I, \rho, F]$:

- $\Sigma$: finite nonempty (input) alphabet
- $S$: finite nonempty set of states
- $I \subseteq S$: set of initial states
- $F \subseteq S$: set of accepting states
- $\rho: S \times \Sigma \rightarrow 2^S$: transition function
Finite State Automata: Semantics

Def. An accepting run of an FSA $A=\left[\Sigma, S, I, \rho, F \right]$ over input word $w = w(1) w(2) \ldots w(n) \in \Sigma^*$ is a sequence $r = r(0) r(1) r(2) \ldots r(n) \in S^*$ of states such that:

- it starts from an initial state: $r(0) \in I$
- it ends in an accepting state: $r(n) \in F$
- it respects the transition function:
  $$r(i+1) \in \rho(r(i), w(i)) \text{ for all } 0 \leq i < n$$
Def. Any FSA $A = [\Sigma, S, I, \rho, F]$ defines a set of input words $\langle A \rangle$:

$\langle A \rangle \triangleq \{ w \in \Sigma^* \mid \text{there is an accepting run of } A \text{ over } w \}$

$\langle A \rangle$ is called the language of $A$
**Propositional Linear Temporal Logic (LTL) formulae** are defined by the grammar:

\[ F ::= p \mid \neg F \mid F \land G \mid X F \mid F U G \]

with \( p \in P \) any atomic proposition from a fixed set \( P \).

**Temporal (modal) operators:**
- next: \( X F \)
- until: \( F U G \)
- release: \( F R G \triangleq \neg (\neg F U \neg G) \)
- eventually: \( \Diamond F \triangleq \text{True} U F \)
- always: \( \Box F \triangleq \neg \Diamond \neg F \)

**Propositional connectives:**
- not: \( \neg F \)
- and: \( F \land G \)
- or: \( F \lor G \triangleq \neg (\neg F \land \neg G) \)
- implies: \( F \Rightarrow G \triangleq \neg F \lor G \)
- iff: \( F \Leftrightarrow G \triangleq (F \Rightarrow G) \land (G \Rightarrow F) \)
Def. A word \( w = w(1) \ldots w(n) \in P^* \)
satisfies an LTL formula \( F \) at position \( 1 \leq i \leq n \), denoted \( w, i \models F \), under the following conditions:

- \( w, i \models p \iff p = w(i) \)
- \( w, i \models \neg F \iff w, i \models F \) does not hold
- \( w, i \models F \land G \iff \) both \( w, i \models F \) and \( w, i \models G \) hold
- \( w, i \models X F \iff i < n \) and \( w, i+1 \models F \)
  \(-i.e., \) \( F \) holds in the next step
- \( w, i \models F \cup G \iff \) for some \( i \leq j \leq n \) it is: \( w, j \models G \)
  \( \) and for all \( i \leq k < j \) it is \( w, k \models F \)
  \(-i.e., \) \( F \) holds until \( G \) will hold
For **derived operators**:

- \( w, i \models \Diamond F \) iff **for some** \( i \leq j \leq n \) it is: \( w, j \models F \)
  - i.e., \( F \) holds **eventually** (in the future)

- \( w, i \models \Box F \) iff **for all** \( i \leq j \leq n \) it is: \( w, j \models F \)
  - i.e., \( F \) holds **always** (in the future)
Def. Satisfaction:

\[ w \models F \iff w, 1 \models F \]

i.e., word \( w \) satisfies formula \( F \) initially

Def. Any LTL formula \( F \) defines a set of words \( \langle F \rangle \):

\[ \langle F \rangle \triangleq \{ w \in P^* \mid w \models F \} \]

\( \langle F \rangle \) is called the language of \( F \)
Automata-theoretic Model Checking

An semantic view of the Model Checking problem:

- **Given**: a finite-state automaton $A$ and a temporal-logic formula $F$
- if $\langle A \rangle \cap \langle \neg F \rangle$ is empty then any run of $A$ satisfies $F$
- if $\langle A \rangle \cap \langle \neg F \rangle$ is not empty then some run of $A$ does not satisfy $F$
  - any member of the nonempty intersection $\langle A \rangle \cap \langle \neg F \rangle$ is a counterexample
Automata-theoretic Model Checking

How to check $\langle A \rangle \cap \langle \neg F \rangle = \emptyset$ algorithmically (given $A$, $F$)?

Combination of three different algorithms:

- **LTL2FSA**: given LTL formula $F$ build automaton $a(F)$ such that $\langle F \rangle = \langle a(F) \rangle$

- **FSA-Intersection**: given automata $A$, $B$ build automaton $C$ such that $\langle A \rangle \cap \langle B \rangle = \langle C \rangle$

- **FSA-Emptiness**: given automaton $A$ check whether $\langle A \rangle = \emptyset$ is the case
Exercises:
Semantics of derived operators
LTL derived operators: eventually

Prove that the satisfaction relation

\[ w, i \Vdash \lozenge F \]

for eventually, defined as:

\[ \lozenge F \triangleq \text{True} \cup F \]

is equivalent to:

for some \( i \leq j \leq n \) it is: \( w, j \Vdash F \)
LTL derived operators: eventually

\[ w, i \models \Diamond F \]

iff

\[ w, i \models \text{True} \cup F \quad \text{(definition of eventually)} \]

iff

for some \( i \leq j \leq n \) it is: \( w, j \models F \)
and for all \( i \leq k < j \) it is \( w, k \models \text{True} \)

(simplification of until)

iff

for some \( i \leq j \leq n \) it is: \( w, j \models F \)

(simplification of \( A \) and \( \text{True} \))
LTL derived operators: always

Prove that the satisfaction relation

\[ w, i \models \Box F \]

for *always*, defined as:

\[ \Box F \equiv \neg \Diamond \neg F \]

is equivalent to:

for all \( i \leq j \leq n \) it is: \( w, j \models F \)
LTL derived operators: always

\[ w, i \models □ F \]

iff

\[ w, i \models \neg \Diamond \neg F \]  \hspace{1em} \text{(definition of always)}

iff

\[ w, i \models \Diamond \neg F \]  \hspace{1em} \text{(definition of not)}

iff

it is not the case that: for some \( i \leq j \leq n \) it is: \( w, j \models \neg F \)

\hspace{1em} \text{(semantics of eventually)}

iff

for all \( i \leq j \leq n \) it is not the case that \( w, j \models \neg F \)

\hspace{1em} \text{(semantics of quantifiers: pushing negation inward)}

iff

for all \( i \leq j \leq n \): it is not the case that it is not the case that \( w, j \models F \)

\hspace{1em} \text{(semantics of negation)}

iff

for all \( i \leq j \leq n \) it is: \( w, j \models F \)

\hspace{1em} \text{(simplification of double negation)}
Exercises:
Evaluate LTL formulas on automata
Does the property hold?

☐ (start ⇒ ◇ stop)
Does the property hold?

□ (start ⇒ ◇ stop)

Yes:
- whenever start occurs we reach state closed-cooking
- we must eventually exit state closed-cooking to reach the only accepting state closed-off
- state closed-cooking can be exited only if stop occurs
Does the property hold?

□ ◊ turn_off
Does the property hold?

No:
- counterexample: pull push
Does the property hold?

\[ \square \Diamond (\text{turn\_off} \lor \text{push}) \]
Does the property hold?

□ ◊ (turn_off ∨ push)

Yes:

- every accepting run eventually goes back to state closed-off
- state closed-off can be reached only if either turn_off or push occurs
- the empty word is also compliant with the semantics of the always operator
Does the property hold?

◊ (turn_off ∨ push)
Does the property hold?

◊ (turn_off ∨ push)

No:

- counterexample: the empty word
  (compare the semantics of existential quantification against universal quantification)
Does the property hold?

☐ False

◊ (turn_off v push)
Does the property hold?

Yes:

- “always False” means that False holds at every step in the word: it is satisfied precisely by the empty word.
- if the word is not empty, then it must end with turn_off or push, thus it satisfies the other disjunct.

□ False

◊ (turn_off ∨ push)
Does the property hold?

\[
\text{turn\_on} \cup \text{start} \quad \lor \\
\text{pull} \cup \text{push}
\]
Does the property hold?

No:
- counterexample: the empty word
- counterexample: \(\text{turn\_on} \cup \text{turn\_off}\)
- counterexample: \(\text{turn\_on} \cup \text{pull} \cup \text{push} \cup \text{turn\_off}\)
Does the property hold?

□ ( start ⇒ (cook U ◊ turn_off) )
Does the property hold?

□ ( start ⇒ (cook U ◊ turn_off) )

Yes:

• Once start occurs, turn_off must occur eventually
• Hence "eventually turn_off" is the case right after start occurs
• Cook can occur right after start occurs, one or more times
Exercises:
Equivalence of LTL formulas
Prove that $\lozenge$ is idempotent, that is:

$$\lozenge \lozenge q$$

is equivalent to:

$$\lozenge q$$
Equivalence of formulas

\[ w, i \models \lozenge \lozenge q \]

iff

\[ \text{for some } i \leq j \leq n \text{ it is: } w, j \models \lozenge q \]  
\hspace{1cm}  
\text{(semantics of eventually)}

iff

\[ \text{for some } i \leq j \leq n \text{ it is: for some } j \leq h \leq n \text{ it is: } w, h \models q \]  
\hspace{1cm}  
\text{(semantics of eventually)}

iff

\[ \text{for some } i \leq j \leq h \leq n \text{ it is: } w, h \models q \]  
\hspace{1cm}  
\text{(merging of intervals)}

iff

\[ \text{for some } i \leq h \leq n \text{ it is: } w, h \models q \]  
\hspace{1cm}  
\text{(dropping } j, \text{ a fortiori)}

iff

\[ w, i \models \lozenge q \]  
\hspace{1cm}  
\text{(semantics of eventually)}
Equivalence of formulas

Prove that:

\[ p \cup \diamond q \]

is equivalent to:

\[ \diamond q \]
Equivalence of formulas: \( \Rightarrow \) direction

\[
\begin{align*}
\text{iff} & \\
p U \Diamond q & \quad \text{iff} \\
\text{for some } i \leq j \leq n \text{ it is: } w, j \models \Diamond q & \quad \text{(semantics of until)} \\
\text{and for all } i \leq k < j \text{ it is } w, k \models p & \\
\text{(semantics of until)} \\
\text{implies} & \\
\text{for some } i \leq j \leq n \text{ it is: } w, j \models \Diamond q & \quad \text{(a fortiori)} \\
\text{iff} & \\
\text{for some } i \leq j \leq n \text{ it is: for some } j \leq h \leq n \text{ it is: } w, h \models q & \quad \text{(semantics of eventually)} \\
\text{iff} & \\
\text{for some } i \leq h \leq n \text{ it is: } w, h \models q & \quad \text{(simplification of range of quantification)} \\
\text{iff} & \\
w, i \models \Diamond q & \quad \text{(semantics of eventually)}
\end{align*}
\]
Equivalence of formulas: $\iff$ direction

$w, i \vDash \Diamond q$

iff

for some $i \leq j \leq i$: $w, j \vDash \Diamond q$

(singleton range of quantification)

iff

for some $i \leq j \leq i$: $w, j \vDash \Diamond q$ and True

(semantics of and)

iff

for some $i \leq j \leq i$: $w, j \vDash \Diamond q$

and for all $i \leq k < j$ it is $w, k \vDash p$

(semantics of universally quantified empty range)

implies

for some $i \leq j \leq n$: $w, j \vDash \Diamond q$

and for all $i \leq k < j$ it is $w, k \vDash p$

(a fortiori)

iff

$w, i \vDash p \cup \Diamond q$

(semantics of until)
Exercises:
Automata-theoretic model-checking
(on paper)
Automata-based model checking

Let us prove by model checking that it's not a property of the automaton
Build an automaton with the same language as:

$$\neg ( \Box \Diamond \text{turn\_off} )$$

Let us start from the unnegated formula:

$$\Box \Diamond \text{turn\_off}$$

and then complement the states of the automaton.
¬( □ ◊ turn_off )
FSA-Emptiness: node reachability

Any accepting run on the intersection automaton is a counterexample to the LTL formula being a property of the automaton

- pull
- push
- pull
- push
- pull
- push
- ...