SPIN

An explicit state model checker

Properties

- Safety properties
  - Something bad never happens
  - Properties of states

Reachability is sufficient

- Liveness properties
  - Something good eventually happens
  - Properties of paths

We need something more complex to check liveness properties

LTL Model Checking

- Liveness properties are expressed in LTL
  - Subset of CTL* of the form:
    - A f

    where f is a path formula which does not contain any quantifiers
- The quantifier A is usually omitted.
- G is substituted by □ (always)
- F is substituted by ◊ (eventually)
- X is (sometimes) substituted by ° (next)

LTL Formulae

- Always eventually p: $\Box \Diamond p$

AGFp in CTL*

AG AF p in CTL

- Always after p there is eventually q:

$$\Box (p \rightarrow (\Diamond q))$$

AG(p→Fq) in CTL*

AG(p →AFq) in CTL

- Fairness:

$$(\Box \Diamond p) \rightarrow \varphi$$

A((GF p) → φ) in CTL*

Can’t express it in CTL

LTL Semantics

- The semantics is the one defined by CTL*
- Given an infinite execution trace σ = *s*$_0$*s*$_1$…

$$\sigma \models p \Leftrightarrow p(s_0)$$

$$\sigma \models \phi_1 \wedge \phi_2 \Leftrightarrow (\sigma \models \phi_1) \wedge (\sigma \models \phi_2)$$

$$\sigma \models \phi_1 \vee \phi_2 \Leftrightarrow (\sigma \models \phi_1) \vee (\sigma \models \phi_2)$$

$$\sigma \models \neg\phi \Leftrightarrow \sigma \not\models \phi$$

$$\sigma \models []\phi \Leftrightarrow \forall i \geq 0.(\sigma)_i \models \phi$$

$$\sigma \models <> \phi \Leftrightarrow \exists i \geq 0.(\sigma)_i \models \phi$$

$$\sigma \models \phi_1 U \phi_2 \Leftrightarrow \exists i \geq 0.\left((\sigma)_i \models \phi_2 \wedge \left(\forall 0 \leq j < i.(\sigma)_j \models \phi_1\right)\right)$$

LTL Model Checking

- An LTL formula defines a set of traces
- Check trace containment
  - Traces of the program must be a subset of the traces defined by the LTL formula
  - If a trace of the program is not in such set
    - It violates the property
    - It is a counterexample
  - LTL formulas are universally quantified

LTL Model Checking

- Trace containment can be turned into emptiness checking
  - Negate the formula corresponds to complement the defined set:

$$set(\phi) = \overline{set(\neg\phi)}$$

  - Subset corresponds to empty intersection:

$$A \subseteq B \Leftrightarrow A \cap \overline{B} = 0$$

Buchi Automata

- An LTL formula defines a set of infinite traces
- Define an automaton which accepts those traces
- Buchi automata are automata which accept sets of infinite traces

Buchi Automata

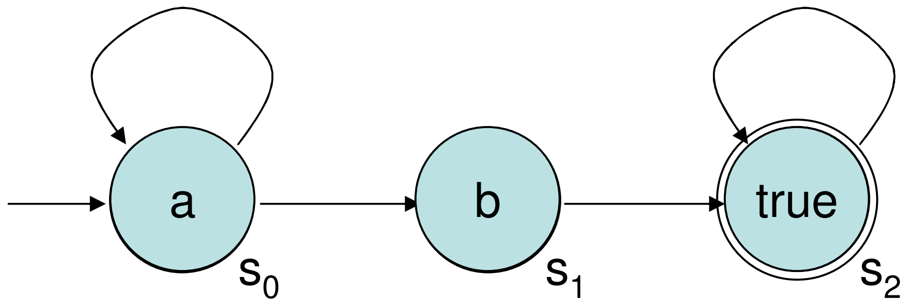
- A Buchi automaton is 4-tuple $<S,I,\delta,F>$:
  - S is a set of states
  - $I \subseteq S$ is a set of initial states
  - $\delta: S \rightarrow 2^S$ is a transition relation
  - $F \subseteq S$ is a set of accepting states
- We can define a labeling of the states:
  - $\lambda: S \rightarrow 2^L$ is a labeling function

  where L is the set of literals.

Buchi Automata

$S = \{ s_0, s_1, s_2 \}$

$I = \{ s_0 \}$



$\delta = \{ (s_0, \{s_0, s_1\}), (s_1, \{s_2\}), (s_2, \{s_2\}) \}$

$F = \{ s_2 \}$

$\lambda = \{ (s_0, \{a\}), (s_1, \{b\}), (s_2, \{\}) \}$

Buchi Automata

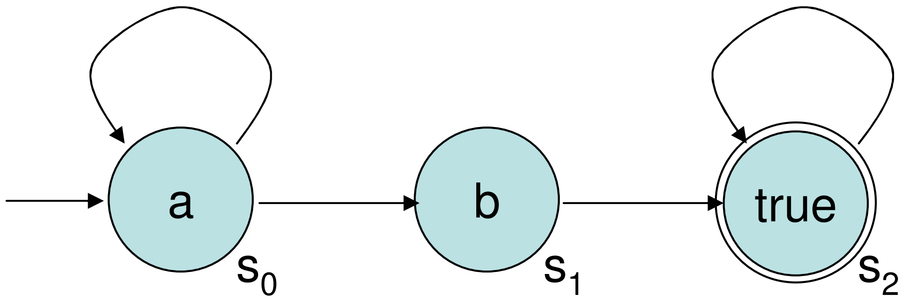
- An infinite trace $\sigma = s_0 s_1 \ldots$ is accepted by a Buchi automaton iff:
  - $s_0 \in I$
  - $\forall\, i \geq 0$: $s_{i+1} \in \delta(s_i)$
  - $\forall\, i \geq 0$: $\exists\, j > i$: $s_j \in F$

LTL Model Checking

- Let's assume each state is labeled with a complete set of literals
  - Each proposition or its negation is present
  - Labeling function Λ
- A Buchi automaton accepts a trace $\sigma = S_0S_1\ldots$
  - $\exists s_o \in I: \Lambda(S_0) \subseteq \lambda(s_o)$
  - $\forall\, i \geq 0: \exists\, s_{i+1} \in \delta(s_i).\ \Lambda(S_{i+1}) \subseteq \lambda(s_{i+1})$
  - $\forall\, i \geq 0: \exists\, j > i: s_j \in F$

Buchi Automata

σ = a a a a a b b b a c c c c …



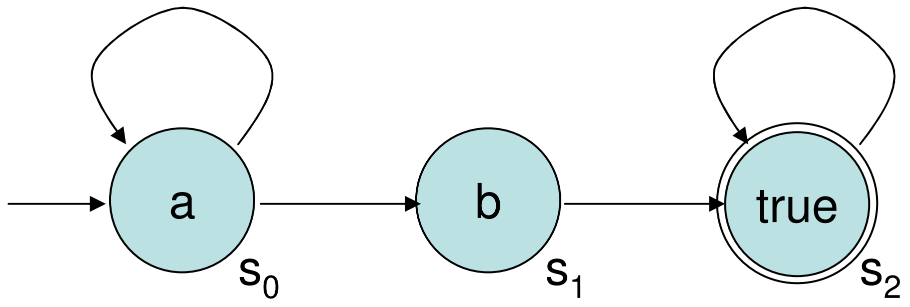
σ = a a a c a b b b a b a b b …

Buchi Automata

- Some properties:
  - Not all non-deterministic Buchi automata have an equivalent deterministic Buchi automata
  - Not all Buchi automata correspond to an LTL formula
  - Every LTL formula corresponds to a Buchi automaton
  - Set of Buchi automata closed until complement, union, intersection, and composition

Buchi Automata

What LTL formula does this Buchi automaton corresponds to (if any)?



a U b

LTL Model Checking

- Generate a Buchi automaton for the negation of the LTL formula to check
- Compose the Buchi automaton with the automaton corresponding to the system
- Check emptiness

LTL Model Checking

- Composition:
  - At each step alternate transitions from the system and the Buchi automaton
- Emptiness:
  - To have an accepted trace:
    - There must be a cycle
    - The cycle must contain an accepting state

LTL Model Checking

- Cycle detection
  - Nested DFS
    - Start a second DFS
    - Match the start state in the second DFS
      - Cycle!
    - Second DFS needs to be started at each state?
      - Accepting states only will suffice
    - Each second DFS is independent
      - If started in post-order states need to be visited at most once in the second DFS searches

LTL Model Checking

```
procedure DFS(s)
  visited = visited ∪ {s}
  for each successor s' of s
    if s' ∉ visited then
      DFS(s')
      if s' is accepting then
        DFS2(s', s')
      end if
    end if
  end for
end procedure
```

LTL Model Checking

```
procedure DFS2(s, seed)
  visited2 = visited2 $\cup$ {s}
  for each successor s' of s
    if s' = seed then
      return "Cycle Detect";
    end if
    if s' $\notin$ visited2 then
      DFS2(s', seed)
    end if
  end for
end procedure
```

References

- http://spinroot.com/
- ***Design and Validation of Computer Protocols*** by Gerard Holzmann
- ***The Spin Model Checker*** by Gerard Holzmann
- ***An automata-theoretic approach to automatic program verification***, by Moshe Y. Vardi, and Pierre Wolper
- ***An analysis of bitstate hashing***, by G.J. Holzmann
- ***An Improvement in Formal Verification***, by G.J. Holzmann and D. Peled
- ***Simple on-the-fly automatic verification of linear temporal logic***, by Rob Gerth, Doron Peled, Moshe Vardi, and Pierre Wolper
- ***A Minimized automaton representation of reachable states***, by A. Puri and G.J. Holzmann