



Automata Framework

Literature: Peled ch. 6

Mads Dam

2005 Mads Dam IMIT, KTH

1

2G1516 Formal Methods

The Reachability Problem

Many problems can be cast in terms of state reachability:

Does there exist a path from an initial state to a designated state?

Examples:

- Invariants: Is there a path to a state failing a given invariant?
- Deadlock detection: Is there a path to a deadlocked state
- Dead code: If a given program is reachable it will be executable under some circumstances, so it is not dead

Transition system (state space) is just a graph – maybe we can use some graph algorithm

2005 Mads Dam IMIT, KTH

2

2G1516 Formal Methods

Depth First Search

```

program DFS
for all  $q_0 \in Q_0$  do
  add-statespace( $q_0$ );
  push-stack( $q_0$ );
  dfs()
od
end DFS

procedure dfs()
 $q := \text{top-stack}$ ();
for all  $q'$  such that  $q \rightarrow q'$ 
do
  if not in-statespace( $q'$ )
  then
    begin
      add-statespace( $q'$ );
      push-stack( $q'$ );
      dfs()
    end
  end
od;
pop-stack()

```

2005 Mads Dam IMIT, KTH

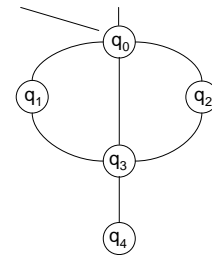
3

2G1516 Formal Methods

Initial Configuration

statespace: q_0

stack: q_0



2005 Mads Dam IMIT, KTH

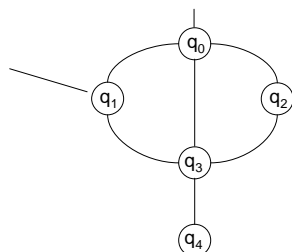
4

2G1516 Formal Methods

Next

statespace: q_0, q_1

stack: q_1, q_0



2005 Mads Dam IMIT, KTH

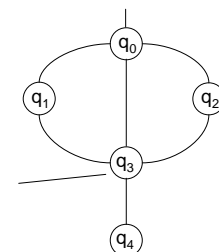
5

2G1516 Formal Methods

Next

statespace: q_0, q_1, q_3

stack: q_3, q_1, q_0



Observe: If q_3 is one of the "goal states", stack holds a path from the initial state = counterexample

2005 Mads Dam IMIT, KTH

6

2G1516 Formal Methods

Backtrack

statespace: q_0, q_1, q_3

stack: q_1, q_0

2005 Mads Dam IMIT, KTH 7 2G1516 Formal Methods

Backtrack

statespace: q_0, q_1, q_3

stack: q_0

2005 Mads Dam IMIT, KTH 8 2G1516 Formal Methods

Next

statespace: q_0, q_1, q_3, q_2

stack: q_2, q_0

2005 Mads Dam IMIT, KTH 9 2G1516 Formal Methods

Backtrack and Done

statespace: q_0, q_1, q_3, q_2

stack: q_0

2005 Mads Dam IMIT, KTH 10 2G1516 Formal Methods

Cost of Verification

Time and space is linear in number of reachable product states

Worst case: Exponential in number of components

Hope: Explore only small fraction of product space

Still that's often far too much

2005 Mads Dam IMIT, KTH 11 2G1516 Formal Methods

Space vs Time Trade-offs

Statespace routines only prevent double work

Can randomly erase states from statespace, but only if not on stack

Can replace *in-statespace* with *in-stack* and omit all calls to statespace

Algorithm still works and is guaranteed to terminate

Increase in run time between $O(R)$ and $O(R^2)$ where R is number of reachable states

2005 Mads Dam IMIT, KTH 12 2G1516 Formal Methods

Breadth-First Search

DFS does not guarantee that shortest counterexample is found
 BFS could achieve this
 Reachable states generated one "generation" at a time
 Each generation contains all states reachable from any state in current generation, after traversing one edge
 BFS: Space consumed order of max width of search tree
 DFS: Space consumed order of max depth of search tree

BFS vs DFS

- | | |
|--|----------------|
| Example 1: | Example 2: |
| • Two successor states per state (binary tree) | • Single trace |
| • Width after n steps is 2^n | • Width is 1 |
| • Depth is n | • Depth is R |

Producing counterexample for BFS requires 2^{nd} DFS search

Neither method is optimal in all cases

DFS is generally preferred

Verification Using Automata

State reachability addresses only limited temporal properties
 Automata theoretic framework:
 - Model cast as Buchi automaton: $Model \Rightarrow A_{Model}$
 - Specification cast as Buchi automaton: $Spec \Rightarrow A_{Spec}$

Want: $L(A_{Model}) \subseteq L(A_{Spec})$

Or equivalently:

$$L(A_{Model}) \cap \overline{L(A_{Spec})} = \emptyset$$

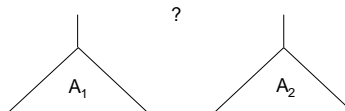
Algorithm: Complementation, intersection, emptiness check

Union

Given: Buchi Automata $A_i = (Q_i, \Sigma, \Delta_i, I_i, F_i)$, $i \in \{1, 2\}$:

$$A_1 \cup A_2 = (Q_1 \cup Q_2, \Sigma, \Delta_1 \cup \Delta_2, I_1 \cup I_2, F_1 \cup F_2)$$

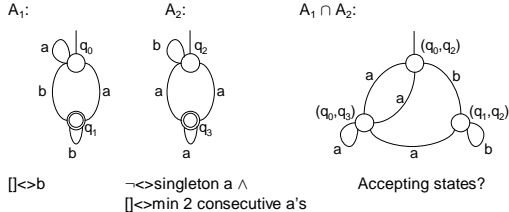
Q_1, Q_2 must be disjoint
 Then just a matter of choosing the right initial state



Intersection

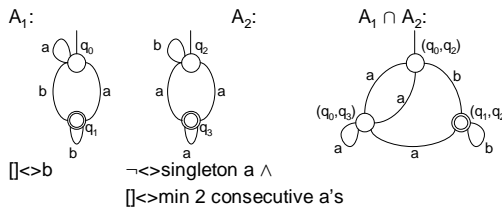
Given: Buchi Automata $A_i = (Q_i, \Sigma, \Delta_i, I_i, F_i)$, $i \in \{1, 2\}$:
 • $A_1 \cap A_2 = (Q_1 \times Q_2, \Sigma, \{((q_1, q_2), \alpha, (q_1', \alpha, q_2')) \mid q_1 \rightarrow \alpha q_1', q_2 \rightarrow \alpha q_2'\}, I_1 \times I_2, F_1 \times F_2)$

Example:



Attempt #1

State (q_1, q_2) accepting if q_1 or q_2 accepting



Doesn't work: b^0 is accepted

Attempt #2

State (q_1, q_2) accepting if q_1 and q_2 accepting

A_1 :

$\llbracket \langle \rangle b$

A_2 :

$\neg \langle \rangle \text{singleton } a \wedge$
 $\llbracket \langle \rangle \text{min 2 consecutive } a\text{'s}$

$A_1 \cap A_2$:

Doesn't work: Accepts nothing

2005 Mads Dam IMIT, KTH192G1516 Formal Methods

Generalized Buchi Automata

Structure $A = (Q, \Sigma, \Delta, I, F)$
where

- Q, Σ, Δ, I as before
- $F = \{f_1, \dots, f_m\}$, for all $i: 1 \leq i \leq m, f_i \subseteq Q$

Word $w = a_1 a_2 \dots a_n \dots$ is accepted:
Exists sequence

$$q_0 \rightarrow^{a_1} q_1 \rightarrow^{a_2} \dots \rightarrow^{a_n} q_n \rightarrow^{a_{n+1}} \dots$$

such that

1. $q_0 \in I$
2. for each $i: 1 \leq i \leq m$, exists infinitely many j such that $q_j \in f_i$

2005 Mads Dam IMIT, KTH202G1516 Formal Methods

Intersection Again

Given: Buchi Automata $A_i = (Q_i, \Sigma, \Delta_i, I_i, F_i)$, $i \in \{1, 2\}$:

- $A_1 \cap A_2 = (Q_1 \times Q_2, \Sigma, \{((q_1, q_2), \alpha, (q_1', \alpha, q_2')) \mid q_1 \rightarrow \alpha q_1', q_2 \rightarrow \alpha q_2'\}, I_1 \times I_2, \{F_1 \times Q_2, Q_1 \times F_2\})$

A_1 :

$\llbracket \langle \rangle b$

A_2 :

$\neg \langle \rangle \text{singleton } a \wedge$
 $\llbracket \langle \rangle \text{min 2 consecutive } a\text{'s}$

$A_1 \cap A_2$:

2005 Mads Dam IMIT, KTH212G1516 Formal Methods

Generalized -> "Standard" Automata

Given: Generalized Buchi Automaton $A = (Q, \Sigma, \Delta, I, \{f_1, \dots, f_m\})$
Result: Ordinary Buchi automaton

Idea: Create m copies of A . Run each in turn. Once accepting state in i 'th copy is visited, move to $i \oplus 1$ 'th. Pick one of the set of accepting state at random

f_1 :

f_2 :

Standard automaton:

2005 Mads Dam IMIT, KTH222G1516 Formal Methods

Complementation

Finite automata:

- Determinisation (subset construction), then complementation

Doesn't work here:

Recognizes $\langle \rangle [a$

Accepts $(ab)^\omega$

Fairly complex construction - left out
Exponential size blow-up, lower bound

2005 Mads Dam IMIT, KTH232G1516 Formal Methods

Checking Emptiness

Does A accept some ω -word?

Must be possible to reach some accepting state q

- Solvable by DFS search

Must exist cycle $q \rightarrow \dots \rightarrow q$

- Solvable by second DFS search

Linear complexity
Return to this later

2005 Mads Dam IMIT, KTH242G1516 Formal Methods

Summary

To check whether A satisfies spec B, as Buchi automata:

1. Complement B
2. Compute $C = A \cap \bar{B}$
3. Find strongly connected components, SCC, of C
4. Is there an accepting state in a member of SCC?
5. If no: A satisfies spec B
6. If yes: A does not satisfy B
7. Pick SCC C'
8. Construct path σ_1 from initial state to accepting state q in C'
9. Construct path σ_2 from q to q in C'
10. Path $\sigma_1\sigma_2^\omega$ is counterexample