Exploring Reachable States

Transition system: $M = (S, I, r, A)$, where $I \subseteq S$, $r \subseteq S \times A \times S$

$\bar{r} = \{(s, s') \mid \exists a \in A.(s, a, s') \in r\}$

Similarly to $post(X)$ previously, define:

$$G : 2^S \to 2^S$$
$$G(X) = I \cup \bar{r}[X]$$

What properties does function G have?

Exploring Reachable States

Transition system: $M = (S, I, r, A)$, where $I \subseteq S$, $r \subseteq S \times A \times S$

$\bar{r} = \{(s, s') \mid \exists a \in A.(s, a, s') \in r\}$

Similarly to $post(X)$ previously, define:

$$G : 2^S \to 2^S$$
$$G(X) = I \cup \bar{r}[X]$$

What properties does function G have?
G is monotonic!

Exploring Reachable States

Transition system: $M = (S, I, r, A)$, where $I \subseteq S$, $r \subseteq S \times A \times S$

$\bar{r} = \{(s, s') \mid \exists a \in A.(s, a, s') \in r\}$

Similarly to $post(X)$ previously, define:

$$G : 2^S \to 2^S$$
$$G(X) = I \cup \bar{r}[X]$$

What properties does function G have?
G is monotonic!
Indeed. Say $S \subseteq S'$. Then $\bar{r}[S] \subseteq \bar{r}[S']$, so $I \cup \bar{r}[S] \subseteq I \cup \bar{r}[S']$.
That is, $G(S) \subseteq G(S')$.

Define $G^0(X) = X$, $G^{n+1}(X) = G(G^n(X))$.

What $G^n(\emptyset)$ is

$$G(X) = I \cup \bar{r}[X]$$

$$G(\emptyset) =$$

What $G^n(\emptyset)$ is

$$G(X) = I \cup \bar{r}[X]$$

$$G(\emptyset) = I$$
$$G^2(\emptyset) =$$

What $G^n(\emptyset)$ is

$$G(X) = I \cup \bar{r}[X]$$

$$G(\emptyset) = I$$
$$G^2(\emptyset) = I \cup \bar{r}[I]$$
$$G^3(\emptyset) =$$

What $G^n(\emptyset)$ is

$$G(X) = I \cup \bar{r}[X]$$

$$\begin{aligned} G(\emptyset) &= I \\ G^2(\emptyset) &= I \cup \bar{r}[I] \\ G^3(\emptyset) &= I \cup \bar{r}[I \cup \bar{r}[I]] \end{aligned}$$

What $G^n(\emptyset)$ is

$$G(X) = I \cup \bar{r}[X]$$

$$G(\emptyset) = I$$
$$G^2(\emptyset) = I \cup \bar{r}[I]$$
$$G^3(\emptyset) = I \cup \bar{r}[I \cup \bar{r}[I]] \ I \cup \bar{r}[I] \cup \bar{r}^2[I]$$
$$\ldots$$

$$G^n(\emptyset) = \bigcup_{k<n} \bar{r}^n[I]$$

All states reachable in less than n steps!

What $G^n(\emptyset)$ is

$$G(X) = I \cup \bar{r}[X]$$

$$G(\emptyset) = I$$
$$G^2(\emptyset) = I \cup \bar{r}[I]$$
$$G^3(\emptyset) = I \cup \bar{r}[I \cup \bar{r}[I]] \; I \cup \bar{r}[I] \cup \bar{r}^2[I]$$
$$\ldots$$

$$G^n(\emptyset) = \bigcup_{k<n} \bar{r}^n[I]$$

All states reachable in less than n steps!

Thus,

$$Reach(M) = \bigcup_{n \geq 0} G^n(\emptyset)$$

Sequence of States

Consider the infinite sequence $G^i(\emptyset)$ for all i:

$$\emptyset, G(\emptyset), G(G(\emptyset)), \ldots, G^n(\emptyset), \ldots,$$

Sequence of States

Consider the infinite sequence $G^i(\emptyset)$ for all i:

$$\emptyset, G(\emptyset), G(G(\emptyset)), \ldots, G^n(\emptyset), \ldots,$$

Note: $\emptyset \subseteq G(\emptyset)$ $\longleftarrow$ because $\emptyset \subseteq Y$ for every Y

Sequence of States

Consider the infinite sequence $G^i(\emptyset)$ for all i:

$$\emptyset, G(\emptyset), G(G(\emptyset)), \ldots, G^n(\emptyset), \ldots,$$

Note: $\emptyset \subseteq G(\emptyset)$ $\longleftarrow$ because $\emptyset \subseteq Y$ for every Y

then: $G(\emptyset) \subseteq G(G(\emptyset))$ $\longleftarrow$ because G is monotonic!

Sequence of States

Consider the infinite sequence $G^i(\emptyset)$ for all i:

$$\emptyset, G(\emptyset), G(G(\emptyset)), \ldots, G^n(\emptyset), \ldots,$$

Note: $\emptyset \subseteq G(\emptyset)$ $\longleftarrow$ because $\emptyset \subseteq Y$ for every Y
then: $G(\emptyset) \subseteq G(G(\emptyset))$ $\longleftarrow$ because G is monotonic!
then: $G(G(\emptyset)) \subseteq G(G(G(\emptyset)))$ $\longleftarrow$ because G is monotonic!

Sequence of States

Consider the infinite sequence $G^i(\emptyset)$ for all i:

$$\emptyset, G(\emptyset), G(G(\emptyset)), \ldots, G^n(\emptyset), \ldots,$$

Note: $\emptyset \subseteq G(\emptyset)$ $\longleftarrow$ because $\emptyset \subseteq Y$ for every Y
then: $G(\emptyset) \subseteq G(G(\emptyset))$ $\longleftarrow$ because G is monotonic!
then: $G(G(\emptyset)) \subseteq G(G(G(\emptyset)))$ $\longleftarrow$ because G is monotonic!
...

$$\emptyset \subseteq G(\emptyset) \subseteq G^2(\emptyset) \ldots \subseteq G^n(\emptyset) \subseteq G^{n+1}(\emptyset) \subseteq \ldots$$

Sequence of States

Consider the infinite sequence $G^i(\emptyset)$ for all i:

$$\emptyset, G(\emptyset), G(G(\emptyset)), \ldots, G^n(\emptyset), \ldots,$$

Note: $\emptyset \subseteq G(\emptyset)$ $\longleftarrow$ because $\emptyset \subseteq Y$ for every Y
then: $G(\emptyset) \subseteq G(G(\emptyset))$ $\longleftarrow$ because G is monotonic!
then: $G(G(\emptyset)) \subseteq G(G(G(\emptyset)))$ $\longleftarrow$ because G is monotonic!
...

$$\emptyset \subseteq G(\emptyset) \subseteq G^2(\emptyset) \ldots \subseteq G^n(\emptyset) \subseteq G^{n+1}(\emptyset) \subseteq \ldots$$

Suppose that S is finite. What must happen?

Sequence of States

Consider the infinite sequence $G^i(\emptyset)$ for all i:

$$\emptyset, G(\emptyset), G(G(\emptyset)), \ldots, G^n(\emptyset), \ldots,$$

Note: $\emptyset \subseteq G(\emptyset)$ $\longleftarrow$ because $\emptyset \subseteq Y$ for every Y
then: $G(\emptyset) \subseteq G(G(\emptyset))$ $\longleftarrow$ because G is monotonic!
then: $G(G(\emptyset)) \subseteq G(G(G(\emptyset)))$ $\longleftarrow$ because G is monotonic!
...

$$\emptyset \subseteq G(\emptyset) \subseteq G^2(\emptyset) \ldots \subseteq G^n(\emptyset) \subseteq G^{n+1}(\emptyset) \subseteq \ldots$$

Suppose that S is finite. What must happen? $G(G^n(\emptyset)) = G^n(\emptyset)$

A Reachability Procedure

```
def findXstar(S,I,r,A) =
  def G(X) = I ∪ r̄[X]
  var X = ∅; var GX = G(X)
  while GX != X do
    X = GX; GX = G(X)
  end while
  X
```

What do we know about any set X^* where for some n, $X^* = G^n(\emptyset)$ and $G(X^*) = X^*$?

A Reachability Procedure

```
def findXstar(S,I,r,A) =
  def G(X) = I ∪ r̄[X]
  var X = ∅; var GX = G(X)
  while GX != X do
    X = GX; GX = G(X)
  end while
  X
```

What do we know about any set X^* where for some n, $X^* = G^n(\emptyset)$ and $G(X^*) = X^*$?

$$Reach(M) = \bigcup_{i \geq 0} G^i(\emptyset) = \underbrace{G^0(\emptyset) \cup \ldots G^{n-1}(\emptyset)}_{\subseteq X^*} \cup \underbrace{G^n(\emptyset)}_{=X^*} \cup \underbrace{G^{n+1}(\emptyset) \cup \ldots}_{=X^*} = X^*$$

Stops in step that exceeds the length of the longest trace of non-repeating states.
Need not terminate for infinite systems.

Fixed Point of a Function

Given a function on some set, $f : S \longrightarrow S$, a value $x \in S$ is a fixed point iff $f(x) = x$.

Fixed Point of a Function

Given a function on some set, $f : S \rightarrow S$, a value $x \in S$ is a fixed point iff $f(x) = x$.

Conclusion: reachable states are a fixed point of function $G(X) = I \cup \bar{r}[X]$.

Fixed Point of a Function

Given a function on some set, $f : S \to S$, a value $x \in S$ is a fixed point iff $f(x) = x$.

Conclusion: reachable states are a fixed point of function $G(X) = I \cup \bar{r}[X]$.

How did we call a set of states X with the property $G(X) \subseteq X$?

Fixed Point of a Function

Given a function on some set, $f : S \to S$, a value $x \in S$ is a fixed point iff $f(x) = x$.

Conclusion: reachable states are a fixed point of function $G(X) = I \cup \bar{r}[X]$.

How did we call a set of states X with the property $G(X) \subseteq X$?

$$I \cup \bar{r}[X] \subseteq X$$

Fixed Point of a Function

Given a function on some set, $f : S \to S$, a value $x \in S$ is a fixed point iff $f(x) = x$.

Conclusion: reachable states are a fixed point of function $G(X) = I \cup \bar{r}[X]$.

How did we call a set of states X with the property $G(X) \subseteq X$?

$$I \cup \bar{r}[X] \subseteq X$$

Equivalent to:

Fixed Point of a Function

Given a function on some set, $f : S \to S$, a value $x \in S$ is a fixed point iff $f(x) = x$.

Conclusion: reachable states are a fixed point of function $G(X) = I \cup \bar{r}[X]$.

How did we call a set of states X with the property $G(X) \subseteq X$?

$$I \cup \bar{r}[X] \subseteq X$$

Equivalent to:

1. $I \subseteq X$
2. $\bar{r}[X] \subseteq X$, that is, if $s \in X$ and $(s, s') \in \bar{r}$, then $s' \in X$

Inductive invariants are precisely sets X for which $G(X) \subseteq X$ (called postfix points)

How to implement X and $G(X)$?

def findXstar(S,I,r,A) =
  def G(X) = $I \cup \bar{r}[X]$
  var X = $\emptyset$; **var** GX = G(X)
  while GX != X **do**
    X = GX; GX = G(X)
  end while
  X

We can use:

- **explicit-state model checking**: sets of states, e.g. hash tables

  val GX =
    for (s ← X,
          s' ← $\bar{r}[\{s\}]$)
    yield s'

- **symbolic model checking**: formulas and their normal forms, such as BDDs (binary decision diagrams)

Symbolic Algorithm

Instead of a set $X_1 \subseteq S$, we use a formula X that is true precisely for states in X_1

var X = False; **var** GX = G(X)
while SAT($GX \wedge \neg X$) **do**
  X = GX; GX = G(X)
end while
X

- Empty set is formula False.
- Checking $GX \neq X$ can be replaced by checking $\neg(GX \subseteq X)$
  - the other inclusion always holds
  - If GX and X are formulas with free variables, $\neg(GX \subseteq X)$ becomes satisfiability of $GX \wedge \neg X$
- It remains to implement $G(X)$ on formulas

Symbolic G

$$
\begin{array}{rcll}
GX_1 & = & I & \cup \; \bar{r}[X_1] \\
s \in GX_1 & \iff & s \in I & \vee \; s \in r[X_1] \\
s \in GX_1 & \iff & s \in I & \vee \; \exists s_0.\ s_0 \in X_1 \wedge \exists a.(s_0, a, s) \in r
\end{array}
$$

We use formulas X, GX over Boolean variables $\bar{s}$. Relation R has variables $\bar{s}, \bar{a}, \bar{s}'$, formula *Init* stands for the set I. Let R' denote $\exists \bar{a}.R[\bar{s} := \bar{s}_0, \bar{s}' := \bar{s}]$
We define:

$$G(X) = Init \ \vee \ \exists \bar{s}_0.(X[\bar{s} := \bar{s}_0] \wedge R')$$

If we want to keep formulas to be quantifier-free, we need to eliminate $\exists \bar{s}_0$ at every step (exponential blowup, substitute all truth values).

(Simple) Symbolic Algorithm Summary

def G(X) $=$ *Init* $\vee$ *eliminate*$(\bar{s}_0,\ X[\bar{s} := \bar{s}_0] \wedge R')$
var X $=$ False; **var** GX $=$ G(X)
while SAT$(GX \wedge \neg X)$ **do**
  X $=$ GX; GX $=$ G(X)
end while
X

(Simple) Symbolic Algorithm Summary

def G(X) = $\mathit{Init} \ \vee \ \mathit{eliminate}(\bar{s}_0, \ X[\bar{s} := \bar{s}_0] \wedge R')$
var X = False; **var** GX = G(X)
while SAT($GX \wedge \neg X$) **do**
  X = GX; GX = G(X)
end while
X

Does this algorithm terminate for all finite-state systems?

(Simple) Symbolic Algorithm Summary

def G(X) $=$ *Init* $\vee$ $\mathit{eliminate}(\bar{s}_0,\ X[\bar{s} := \bar{s}_0] \wedge R')$
var X $=$ False; **var** GX $=$ G(X)
while SAT($GX \wedge \neg X$) **do**
  X $=$ GX; GX $=$ G(X)
end while
X

Does this algorithm terminate for all finite-state systems? **Yes.**

(Simple) Symbolic Algorithm Summary

def G(X) $=$ *Init* $\vee$ *eliminate*$(\bar{s}_0,\ X[\bar{s} := \bar{s}_0] \wedge R')$
var X $=$ False; **var** GX $=$ G(X)
while SAT$(GX \wedge \neg X)$ **do**
  X $=$ GX; GX $=$ G(X)
end while
X

Does this algorithm terminate for all finite-state systems? **Yes.**

To keep formulas smaller, we can try to simplify and normalize formulas at every step.

Disjunctive normal form $\rightarrow$ similar to explicit-state model checking.

A popular alternative normal form: BDDs (binary decision diagrams)

Binary Decision Diagrams (BDDs)

Representation of propositional formulas using rooted directed acyclic graphs with three kinds of nodes:

- two truth value constant nodes 0, 1 (sink nodes)
- ternary connective $ite(c, a, b)$ meaning **if** c **then** a **else** b where c is always a variable and where a, b are formulas (can lead to other parts of the diagram)

How to express:

- $x \wedge y$:

Binary Decision Diagrams (BDDs)

Representation of propositional formulas using rooted directed acyclic graphs with three kinds of nodes:

- two truth value constant nodes 0, 1 (sink nodes)
- ternary connective $ite(c, a, b)$ meaning **if** c **then** a **else** b where c is always a variable and where a, b are formulas (can lead to other parts of the diagram)

How to express:

- $x \land y$: **if** x **then** y **else** 0

Binary Decision Diagrams (BDDs)

Representation of propositional formulas using rooted directed acyclic graphs with three kinds of nodes:

- two truth value constant nodes 0, 1 (sink nodes)
- ternary connective $ite(c, a, b)$ meaning **if** c **then** a **else** b where c is always a variable and where a, b are formulas (can lead to other parts of the diagram)

How to express:

- $x \wedge y$: **if** x **then** y **else** 0
- $x \vee y$:

Binary Decision Diagrams (BDDs)

Representation of propositional formulas using rooted directed acyclic graphs with three kinds of nodes:

- two truth value constant nodes 0, 1 (sink nodes)
- ternary connective $ite(c, a, b)$ meaning **if** c **then** a **else** b where c is always a variable and where a, b are formulas (can lead to other parts of the diagram)

How to express:

- $x \wedge y$: **if** x **then** y **else** 0
- $x \vee y$: **if** x **then** 1 **else** y

Binary Decision Diagrams (BDDs)

Representation of propositional formulas using rooted directed acyclic graphs with three kinds of nodes:

- two truth value constant nodes 0, 1 (sink nodes)
- ternary connective $ite(c, a, b)$ meaning **if** c **then** a **else** b where c is always a variable and where a, b are formulas (can lead to other parts of the diagram)

How to express:

- $x \wedge y$: **if** x **then** y **else** 0
- $x \vee y$: **if** x **then** 1 **else** y
- $\neg x$:

Binary Decision Diagrams (BDDs)

Representation of propositional formulas using rooted directed acyclic graphs with three kinds of nodes:

- two truth value constant nodes 0, 1 (sink nodes)
- ternary connective $ite(c, a, b)$ meaning **if** c **then** a **else** b where c is always a variable and where a, b are formulas (can lead to other parts of the diagram)

How to express:

- $x \wedge y$: **if** x **then** y **else** 0
- $x \vee y$: **if** x **then** 1 **else** y
- $\neg x$: **if** x **then** 0 **else** 1

Two important additional constraints often imposed to make BDDs into ROBDDs:

- **ordered**: impose a total ordering on variables, must always test them in that order
- **reduced**: cannot have two distinct nodes in the graph that have the same definition (test same variable, lead to same outcome nodes)

Operations on Binary Decision Diagrams

ROBDDs representation of a formula can be made unique!

Consequence: we can check formula equivalence $\models F_1 \iff F_2$ by computing ROBDDs for both and checking if they are identical nodes in a global DAG (constant time!)

Unsatisfiable formula is always represented by node 0.

ROBDD for a formula is in some cases exponential in formula size, but the normal form is still one of the most useful ones known.
It is possible to define operations to:

- convert arbitrary formulas into ROBDDs
- eliminate existential variables

We get reasonably efficient symbolic reachability algorithms; used in practice.

Symbolic Reachability Using BDDs

def G(Xnode) = or(initNode,
eliminate($\bar{s}_0$, and(rename(Xnode,$\bar{s}$:= $\bar{s}_0$),
RprimeNode)))
var Xnode = zeroNode; **var** GXnode = G(Xnode)
while and(GXnode, not(X)) != zeroNode **do**
Xnode = GXnode; GXnode = G(Xnode)
end while
Xnode