Model Checking

- Symbolic: Manipulate sets of states.
  - Represent a state as a record of booleans.
  - Represent a set of states as a boolean formula (BDD/CNF).
  - Typically large sets can be represented with small formula's.
  - Worst case for N states with K variables is $\mathcal{O}(N \cdot K)$.
- Explicit: Manipulate single states.
  - Represent a state as a term (tree).
  - Represent a set of states as a container over terms (trees).
  - Worst case for N states with K variables is $\mathcal{O}(N \cdot K)$.
  - Best case is $\mathcal{O}(N)$ is possible.

Representing sets of states

If the variables used are $\vec{x}$ then

- A set of states $S \subseteq \{T, F\}^n$ is a formula $\phi_S(\vec{x})$, such that

$$(b_1, \cdots, b_n) \in S \text{ iff } \phi_S(\vec{b})$$

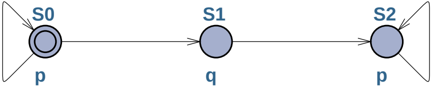
- A binary relation $R \subseteq \{T, F\}^n \times \{T, F\}^n$ on states is a formula $\phi_R(\vec{x}, \vec{x'})$, such that

$$(b_1, \cdots, b_n) \; R \; (b'_1, \cdots, b'_n) \text{ iff } \phi_R(\vec{b}, \vec{b'})$$

- Operation on sets translate to operations on formulas:

$\cup$	$\vee$
$\cap$	$\wedge$
^{c}	$\neg$

Example



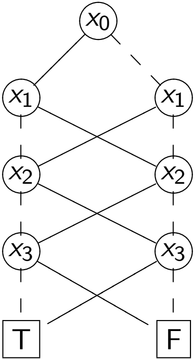
	s_1	s_0
S0	F	F
S1	F	T
S2	T	F

$$
\begin{array}{rcl}
S^0 & = & p \wedge \overline{q} \wedge \overline{s_1} \wedge \overline{s_0} \\
S & = & (p \wedge \overline{q} \wedge \overline{s_1} \wedge \overline{s_0}) \vee (\overline{p} \wedge q \wedge \overline{s_1} \wedge s_0) \vee (p \wedge \overline{q} \wedge s_1 \wedge \overline{s_0}) \\
 & = & (p \wedge \overline{q} \wedge \overline{s_0}) \vee (\overline{p} \wedge q \wedge \overline{s_1} \wedge s_0) \\
\rightarrow & = & (p \wedge \overline{q} \wedge \overline{s_1} \wedge \overline{s_0} \wedge p' \wedge \overline{q'} \wedge \overline{s_1'} \wedge \overline{s_0'}) \\
 & \vee & (p \wedge \overline{q} \wedge \overline{s_1} \wedge \overline{s_0} \wedge \overline{p'} \wedge q' \wedge \overline{s_1'} \wedge s_0') \\
 & \vee & (\overline{p} \wedge q \wedge \overline{s_1} \wedge s_0 \wedge p' \wedge \overline{q'} \wedge s_1' \wedge \overline{s_0'}) \\
 & \vee & (p \wedge \overline{q} \wedge s_1 \wedge \overline{s_0} \wedge p' \wedge \overline{q'} \wedge s_1' \wedge \overline{s_0'})
\end{array}
$$

- A BDD is a DAG built from if-x-then-?-else-?, true and false.
- A good example of compact representation is odd parity

$x_0 \oplus x_1 \oplus x_2 \oplus x_3$ is represented as
($\oplus$ is eXclusive OR.)

———	:	true branch
– – –	:	false branch



Application of BDD.

Reachability:

- Given a set of initial states $S^0 \subseteq \{T, F\}^n$.
- Given a transition relation $\rightarrow \subseteq \{T, F\}^n \times \{T, F\}^n$.
- Compute the set

$$\{s \in \{T, F\}^n \mid \exists \vec{s} : s_0 \in S^0, s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n = s\}$$

Reachability Algorithm

- A simple reachability algorithm is

  reachable()

      visited := $\emptyset$

      next := S^0

      while next $\neq$ visited do

          visited := next ;

          next := $\{s \in \{T, F\}^n \mid \exists s' \in \text{next} : s' \to s\}$

          next := next $\cup$ visited

  return visited

- The assignment

$$\text{next} := \{s \in \{T, F\}^n \mid \exists s' \in \text{next} : s' \to s\}$$

  can be implemented with BDD operations:

$$\text{next}(\vec{x}) := \left(\exists\vec{x}.\text{next}(\vec{x}) \wedge T(\vec{x}, \vec{x'})\right)[\vec{x'} := \vec{x}]$$

Reachability Applied

- Given a set of bad states $Bad(\vec{x})$, we can verify

  $$A\Box\neg bad$$

  by checking if

  $$\mathrm{reachable}() \wedge Bad(\vec{x}) = \mathrm{F}$$

  (Remember that $\mathrm{F} \equiv \emptyset$)
- However, the chance of a BDD using lots of memory is non-negligible.
- So how can we answer the question

  $$\exists n : \ \exists s_i (i = 0 \cdots n) : \ s_0 \in S^0 \wedge s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_n \in Bad$$

  in a more efficient way?

Bounded Model Checking

- Answers the question

$$\exists s_i(i = 0 \cdots n) : \; s_0 \in S^0 \wedge s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_n \in Bad$$

- This is not a complete answer.
- It is a good way of finding bugs.
- Can also answer the question:

$$\begin{aligned} &\exists s_i(i = 0 \cdots n) : \; \exists t_j(j = 1 \cdots k) : \\ &\quad s_0 \in S^0 \wedge \\ &\quad s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_n \in Accept \wedge \\ &\quad s_n = t_1 \longrightarrow t_2 \longrightarrow \cdots \longrightarrow t_n \longrightarrow t_1 \end{aligned}$$

To extend form safety formula's to arbitrary LTL formula's

Bounded Model Checking - algorithm

To answer the question

$$\exists s_i (i = 0 \cdots n) : \; s_0 \in S^0 \wedge s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_n \in Bad$$

- Build the formula

$$\phi = S^0(\vec{x_0}) \wedge T(\vec{x_0}, \vec{x_1}) \wedge \cdots \wedge T(\vec{x_{n-1}}, \vec{x_n}) \wedge Bad(\vec{x_n})$$

- Find a CNF formula ψ such that

$$\mathrm{SAT}(\phi) \text{ iff } \mathrm{SAT}(\psi)$$

- Use a SAT-solver to test if ψ is satisfiable.
  - If ψ is satisfiable then the assignment is a path to a bad state.
  - If ψ is not satisfiable then no such path of length n exists.

Tseitin Transformation

To find a CNF ψ for any given formula ϕ, such that

$$\mathrm{SAT}(\phi) \text{ iff } \mathrm{SAT}(\psi)$$

First, apply Common Subexpression Elimination to obtain

$$\phi(x_1, \cdots, x_n) = \mathrm{let}\ x_{n+1} = \phi_{n+1}, \cdots, x_{n+m} = \phi_{n+m} \ \mathrm{in}\ x_{n+m}$$

such that

$$\phi_i = \neg x_j \text{ or } \phi_i = x_j \triangle x_k \text{ where } j, k < i \text{ and } \triangle \in \{\wedge, \vee, \leftrightarrow\}$$

then we have

$$\mathrm{SAT}(\phi) \text{ iff } \mathrm{SAT}(x_{n+1} \leftrightarrow \phi_{n+1} \wedge \cdots \wedge x_{n+m} \leftrightarrow \phi_{n+m} \wedge x_{n+m})$$

Tseitin Transformation

Second, replace $x_i \leftrightarrow \phi_i$ according to the table

$$
\begin{aligned}
x \leftrightarrow \neg y &\overset{sat}{\Leftrightarrow} (x \vee y) \wedge (\neg x \vee \neg y) \\
x \leftrightarrow (y \vee z) &\overset{sat}{\Leftrightarrow} (x \vee \neg y) \wedge (x \vee \neg z) \wedge (\neg x \vee y \vee z) \\
x \leftrightarrow (y \wedge z) &\overset{sat}{\Leftrightarrow} (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y) \wedge (\neg x \vee z) \\
x \leftrightarrow (y \leftrightarrow z) &\overset{sat}{\Leftrightarrow} (x \vee \neg y \vee \neg z) \wedge (x \vee y \vee z) \wedge \\
&\qquad (\neg x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z) \\
&\vdots
\end{aligned}
$$

Summary

- When sets are represented as BDDs,
  we can check for equality of sets.
  Translating formulas to BDDs can cause exponential blow-up
- When sets are represented as arbitrary formulas,
  we can check for emptiness using Tseitin/SAT.
  SAT is NP-complete
- A tool that uses both BDDs and SAT is NuSMV
  (`http://nusmv.irst.itc.it/`)