The IC3 Algorithm

Shoham Ben-David

- Aaron R. Bradley: *SAT-Based Model Checking without Unrolling*. VMCAI 2011
- *Incremental Construction of Inductive Clauses for Indubitable Correctness* : IC3
  - Known also as Property Directed Reachability

- As of today: the state-of-art symbolic model checking algorithm.

Symbolic Model Checking

- The main problem in model checking: the *size* problem
  - For $|V| = 100$ we have 2^{100} states to explore

Symbolic Model Checking

- The main problem in model checking: the *size* problem
  - For $|V| = 100$ we have 2^{100} states to explore
- Symbolic model checking deals with it by never referring to single states
  - Rather: always refer to *sets of states*

Symbolic Model Checking

- The main problem in model checking: the *size* problem
  - For $|V| = 100$ we have 2^{100} states to explore
- Symbolic model checking deals with it by never referring to single states
  - Rather: always refer to *sets of states*
- A Boolean formula F over the variables V represents a *set of states* in M:
  - All the states that satisfy F.

Some Definition

- A *cube* is a conjunction of literals
  - For a clause c, $\neg c$ is a cube

Some Definition

- A *cube* is a conjunction of literals
  - For a clause c, $\neg c$ is a cube
- S, I, T, P, V, V' as before

Some Definition

- A *cube* is a conjunction of literals
  - For a clause c, $\neg c$ is a cube
- S, I, T, P, V, V' as before
- Primed formulas (e.g. s') are defined on V'

PDR: Frames

The PDR algorithm is based on maintaining a sequence of "frames"

$$R_0, R_1, ..., R_N.$$

1. Each frame is a CNF formula over the variables V, representing a set of states in the model ($R_j \subseteq S$).
2. Each frame R_j is an over-approximations of the states reachable from the initial states I in j steps or less.

Properties of Frames

The frames R_j fulfill the following conditions:

1. $R_0 = I$.
2.
   1. $R_j \subseteq R_{j+1}$.
   2. $\mathrm{CL}(R_{j+1}) \subseteq \mathrm{CL}(R_j)$, for $j > 0$.
3. $T(R_j) \subseteq R_{j+1}$.
4. $R_j \subseteq P$, for $j < N$.

Note that R_N is different from the other frames, as it does not necessarily satisfy P.

Termination of Algorithm

The PDR algorithm proceeds by refining the frames, adding more clauses when possible, while maintaining the conditions discussed above.

The algorithm terminates in one of two cases:

1. For some j, $R_j = R_{j+1}$. In this case a fix point of reachable states have been found, and thus $M \models P$.
2. An error state $s_I \in I$ is found, from which a path to $\neg P$ exists. In this case $M \not\models P$.

Taking a Step Forward

Set a query to the SAT solver:

$$SAT?[R_N \wedge \neg P] \tag{1}$$

Taking a Step Forward

Set a query to the SAT solver:

$$SAT?[R_N \wedge \neg P] \tag{1}$$

- If *not*, then $R_N \subseteq P$.

Taking a Step Forward

Set a query to the SAT solver:

$$SAT?[R_N \wedge \neg P] \tag{1}$$

- If *not*, then $R_N \subseteq P$.
  - Open a new empty frame R_{N+1}

Taking a Step Forward

Set a query to the SAT solver:

$$SAT?[R_N \wedge \neg P] \tag{1}$$

- If *not*, then $R_N \subseteq P$.
  - Open a new empty frame R_{N+1}
  - For every $0 < j$, try to "push" clauses from R_j to R_{j+1}.

Taking a Step Forward

Set a query to the SAT solver:

$$SAT?[R_N \land \neg P] \tag{1}$$

- If *not*, then $R_N \subseteq P$.
    - Open a new empty frame R_{N+1}
    - For every $0 < j$, try to "push" clauses from R_j to R_{j+1}.
        - A clause $\mathrm{c} \in R_j$ can be pushed forward if

$$SAT?[R_j \land T \land \neg \mathrm{c}'] \tag{2}$$

is not satisfiable.

Taking a Step Forward

Set a query to the SAT solver:

$$SAT?[R_N \wedge \neg P] \tag{1}$$

- If *not*, then $R_N \subseteq P$.
  - Open a new empty frame R_{N+1}
  - For every $0 < j$, try to "push" clauses from R_j to R_{j+1}.
    - A clause c $\in R_j$ can be pushed forward if

      $$SAT?[R_j \wedge T \wedge \neg \mathrm{c}'] \tag{2}$$

      is not satisfiable.
  - If two frames are found to be equal, terminate.

Taking a Step Forward

Set a query to the SAT solver:

$$SAT?[R_N \wedge \neg P] \tag{1}$$

- If *not*, then $R_N \subseteq P$.
  - Open a new empty frame R_{N+1}
  - For every $0 < j$, try to "push" clauses from R_j to R_{j+1}.
    - A clause $c \in R_j$ can be pushed forward if

      $$SAT?[R_j \wedge T \wedge \neg c'] \tag{2}$$

      is not satisfiable.
  - If two frames are found to be equal, terminate.
  - Otherwise – continue with Query 1 on R_{N+1}.

Now suppose that $SAT?[R_N \wedge \neg P]$ is satisfiable.

Now suppose that $SAT?[R_N \wedge \neg P]$ is satisfiable.

- The SAT solver provides a satisfying assignment to the variables V

Now suppose that $SAT?[R_N \wedge \neg P]$ is satisfiable.

- The SAT solver provides a satisfying assignment to the variables V
  - A cube s, such that s $\subseteq R_N$, but s $\subseteq S \setminus P$.

The IC3 Algorithm

Now suppose that $SAT?[R_N \wedge \neg P]$ is satisfiable.

- The SAT solver provides a satisfying assignment to the variables V
  - A cube s, such that s $\subseteq R_N$, but s $\subseteq S \setminus P$.
  - If P holds in M, then the states in s are not reachable in M
    - exist in R_N only because R_N is an over-approximation

The IC3 Algorithm

Now suppose that $SAT?[R_N \wedge \neg P]$ is satisfiable.

- The SAT solver provides a satisfying assignment to the variables V
    - A cube s, such that s $\subseteq R_N$, but s $\subseteq S \setminus P$.
    - If P holds in M, then the states in s are not reachable in M
        - exist in R_N only because R_N is an over-approximation
    - We want to *block* s in frame R_N

The IC3 Algorithm

Now suppose that $SAT?[R_N \wedge \neg P]$ is satisfiable.

- The SAT solver provides a satisfying assignment to the variables V
  - A cube s, such that s $\subseteq R_N$, but s $\subseteq S \setminus P$.
  - If P holds in M, then the states in s are not reachable in M
    - exist in R_N only because R_N is an over-approximation
  - We want to *block* s in frame R_N
    - Check

$$SAT?[R_{N-1} \wedge T \wedge \mathrm{s}'] \tag{3}$$

The IC3 Algorithm

Now suppose that $SAT?[R_N \wedge \neg P]$ is satisfiable.

- The SAT solver provides a satisfying assignment to the variables V
  - A cube s, such that $s \subseteq R_N$, but $s \subseteq S \setminus P$.
  - If P holds in M, then the states in s are not reachable in M
    - exist in R_N only because R_N is an over-approximation
  - We want to *block* s in frame R_N
    - Check
$$SAT?[R_{N-1} \wedge T \wedge s'] \tag{3}$$
    - If (3) is not satisfiable, s is blocked
    - Add $\neg s$ to R_N; Continue with Query 1.

The IC3 Algorithm

Check
$SAT?[R_{N-1} \wedge T \wedge s']$

- If (3) is satisfiable
  - We get a cube s_1
  - Needs to be blocked in frame R_{N-1}

Check
$SAT?[R_{N-1} \wedge T \wedge s']$

- If (3) is satisfiable
  - We get a cube s_1
  - Needs to be blocked in frame R_{N-1}
  - Check Query 3 with frame R_{N-2} and s'_1

The IC3 Algorithm

Check
$SAT?[R_{N-1} \wedge T \wedge s']$

- If (3) is satisfiable
  - We get a cube s_1
  - Needs to be blocked in frame R_{N-1}
  - Check Query 3 with frame R_{N-2} and s'_1
  - ...
- If none of the cubes can be blocked during this process, then a query finally returns a cube $s_I \subseteq I$
  - Cannot be blocked
  - P does not hold in the model!