IC3

Invariant Verification Problems

Inductive invariants and relative inductive invariants are central notions to solve the invariant verification problem. F is an *inductive invariant* for S iff $I(X) \models F(X)$, and $F(X) \wedge T(X, X') \models F(X')$. A typical verification strategy is to look for an inductive invariant F such that $F \models P$ (thus, yielding that $S \models P$). F is *inductive relative* to the formula $\phi(X)$ iff $I(X) \models F(X)$, and $\phi(X) \wedge F(X) \wedge T(X, X') \models F(X')$. It is sometimes useful to first prove some lemma and then search for an invariant that is inductive relative to such lemma.

IC3 main idea

The IC3 algorithm tries to prove that $S \models P$ by finding a suitable inductive invariant $F(X)$ such that $F(X) \models P(X)$. In order to construct F, IC3 maintains a sequence of formulas (called *trace*) $F_0(X), \ldots, F_k(X)$ such that: (i) $F_0 = I$; (ii) $F_i \models F_{i+1}$; (iii) $F_i(X) \wedge T(X, X') \models F_{i+1}(X')$; (iv) for all $i < k$, $F_i \models P$. Therefore, each element of the trace F_{i+1}, called *frame*, is inductive relative to the previous one, F_i. IC3 strengthens the frames by finding new relative inductive clauses. A clause c is inductive relative to the frame F, i.e. $F \wedge c \wedge T \models c'$, iff the formula

$$RelInd(F, T, c) \doteq F \wedge c \wedge T \wedge \neg c' \tag{1}$$

is unsatisfiable, so that a check of relative inductiveness can be directly tackled by a SAT (or SMT) solver.

implementation

- **while** is_sat($F[k] \wedge \neg P$):

  $c = \text{get\_state}(F[k] \wedge \neg P)$ # $c \models F[k] \wedge \neg P$

  extract a cube c from the model of $F_k \wedge \neg P$

  if Fk&!P is satifiable, then Fk is intersected with bad states(!P)

  construt a cube c which lead to the bad states Fk & !P

  c contains literals that if and(them) is satisfiable then Fk&!P is satisfiable

implementation

bool rec_block(s, i):

if $i == 0$: **return** False *# reached initial states*

while is_sat($F[i-1] \wedge \neg s \wedge T \wedge s'$):

  c = get_predecessor($i-1$, s') *# see §III-C*

  if not rec_block($c, i-1$): **return** False

g = generalize($\neg s$, i) *# see §III-B*

add g to $F[1] \ldots F[i]$

return True

- **while** is_sat($F[i-1] \wedge \neg s \wedge T \wedge s'$):

  c = get_predecessor($i-1$, s') *# see §III-C*

  if not rec_block($c, i-1$): **return** False

  while $RelInd(F_{i-1}, T, \neg s) \not\models \bot$:

  extract a cube c from the model of $RelInd(F_{i-1}, T, \neg s)$

  # c is a predecessor of s

  if not RecBlock($c, i-1$): **return** False

- finding new relative inductive clauses.
- If the formula RelInd is unsatisfiable, then ¬s is inductive relative to F(i-1), and the bad state s can be blocked at i, then generalize ¬s to ¬g and add ¬g to Fi (i <k)
- if the formula RelInd is satisfiable, then the overapproximation F(i-1) is not strong enough to show that s is unreachable,let c be a subset of the states in F(i-1)∧¬s such that all the states in c lead to a state in s in one transition step. Then, IC3 continues by trying to show that c is not reachable in one step from F(i-2)(that is, it tries to block the pair(c, i-1))

implementation

- 因为在blocking phase中，s |=Fk&¬P可能在小于k的位置上被block掉
- 但不可能在0位置处被block，所以从1开始(i=1...k-1)
- 对于Fi每个子句c，如果不满足(not is_sat)*RelInd*
- 则clause c is inductive relative to the frame Fi
- 即有 F ∧ c ∧ T |= c‘
- 将c添加到F(i+1), 则之前用来block掉s的子句也能被传播到Fk

```
# propagation phase
k = k + 1,  F[k] = ⊤
for i = 1 to k − 1:
    for each clause c ∈ F[i]:
        if not is_sat(F[i] ∧ c ∧ T ∧ ¬c′):
            add c to F[i + 1]
    if F[i] == F[i + 1]: return True  # property proved
```

$$RelInd(F, T, c) \doteq F \wedge c \wedge T \wedge \neg c'$$