IC3 Proposal

""

October 31, 2018

1 Implementing IC3 Model Checking

1.1 Our understanding of the algorithm

- We need to maintain a sequence of frames R_i.
- Each frame is a CNF formula representing the set of states.
- We query to check (R_n /\ ¬ P) is sat.
- If (R_n /\ ¬ P) is unsat
  - We create a new empty frame $R_{(n+1)}$.
  - We try to push clauses from previous frames to this.
  - A clause c can be pushed if R_j /\ T /\ ¬ c' is unsat. (this is the propogation phase)
- If (R_n /\ ¬ P) is sat
  - We can find a cube s such that s is subset of R_n and doesn't satisfy the property.
  - We need to block such s in R_n.
  - To block this in R_n, we query ($R_{(n-1)}$ /\ T /\ s').
  - If unsat, then s is blocked in R_n. We need to add ¬ s to R_n.
  - If sat, We get a new cube which needs to be blocked in $R_{(n-1)}$.
  - At this point, we proceed as we did previously.
  - If the model doesn't satisfy the property, we'll get a cube which needs to be blocked, but it intersects with the initial state (and can't be blocked)

Each time IC3 blocks Bad for one additional level, it enters the propogation phase.

1.2 Pseudo code for the blocking phase.

- Q is the queue maintaning the proof obligations
- A proof obligatioin is a tuple <s, f> where f is a frame, and s is the cube we want to block in frame f.

```
While !Empty(Q) do
  <s,f> <- pop(Q)
  assert ~s is (f - 1) invariant

  if f = 0 then
    return Error "Counter Example Found"

  if SAT(~s /\ R_(f-1) /\ T /\ s')          -- Optimization 2 in the paper

     t <- get predecessors of (s)
     Add (Q, <t, f-1> )
     Add (Q, <s, f>)
  else
     ?? (generalizaiton part)               -- Optimization 1 in the paper
```