CSC410 – Fall 2026

Lecture 6

This is the practice problem for the inverted lecture 6.

You have almost certainly heard about this puzzle before. There is a farmer who wants to transform a fox, a goose, and a bag of beans from one side of a river to another using a boat which can only hold one item in addition to the farmer, subject to the constraints that the fox cannot be left alone with the goose, and the goose cannot be left alone with the beans. The goals are

- to discover if there is a solution, and
- find the minimum number of moves required to do so.

Solution:

See the posted `smtlib` files.

The goal of this lecture's exercise is to practice our understanding of the symbolic reachability algorithms presented in this week's lecture video and find a solution to this puzzle.

In order to contrast the standard one and PDR (property-directed reachability), we can look at a variation of the puzzle above:

- We add a bag of corn as well. So now the Goose can eat both the beans and the corn.
- We add k tortoises, with the idea that they are neither eaten by anything nor eat anything themselves. They are safe to be left with anything.

We solve the puzzle again.

Solution:

Forget the tortoise positions and consider the three items x, b, c. We claim:

$$\boxed{J = (\neg x \vee \neg b) \wedge (\neg x \vee \neg c) \wedge (\neg b \vee \neg c).}$$

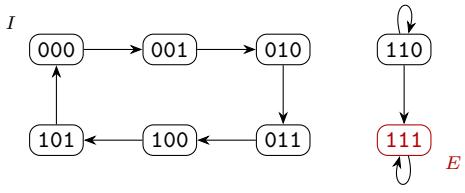
In words, *at most one of the fox, beans, and cabbage is on the right.*

Formally, this is an inductive invariant proving the unreachability of the goal, because the following three queries are all UNSAT:

$$\mathit{Init} \wedge \neg J, \qquad J(v) \wedge T(v, v') \wedge \neg J(v'), \qquad J \wedge \mathit{Bad}.$$

You can learn this invariant using PDR, but it has too many steps to demonstrate with pen and paper. So, instead, as a live example of PDF, consider the simpler example below.

A state is a bitvector `xyz`, and the transition system is illustrated on the right. Note that forward reachability requires 6 + 1 frames; 6 rounds to discover all states, and the last one to confirm that it has reached saturation. We want to discover the much simpler invariant $\neg(x \wedge y)$ that proves the unreachability of the marked error state through PDR.



PDR algorithm based on the course slides

Frames: $R_0 = I$; every other frame is a set of clauses (a new frame has none, i.e. $\top$).

Check if $R_N \wedge E$ is SAT.

- **No?** $R_N \subseteq \neg E$. New empty frame R_{N+1}. For all $j > 0$, push clauses from R_j to R_{j+1}: clause c can be pushed if $R_j \wedge T \wedge \neg c'$ is not SAT. Terminate if two equal frames are found.
- **Yes?** There is a satisfying assignment s.
  $(*)$ Check if $R_{N-1} \wedge T \wedge s'$ is SAT.
  - No? Add $\neg s$ to R_N and start over.
  - Yes? Get assignment t; repeat $(*)$ with (R_{N-2}, t) (if that is UNSAT, add $\neg t$ to R_{N-1}).

#	query	result	action	R_1	R_2
$N = 0$					
1	$R_0 \wedge E$	UNSAT	new frame R_1; nothing to push	$\top$	
$N = 1$					
2	$R_1 \wedge E$	SAT	$s = 111$	$\top$	
3	$R_0 \wedge T \wedge s'$	UNSAT	000 only moves to 001: add $\neg 111$ to R_1	$\neg 111$	
4	$R_1 \wedge E$	UNSAT	new frame R_2	$\neg 111$	$\top$
5	push $\neg 111$ $R_1 \to R_2$: $R_1 \wedge T \wedge 111'$	SAT	$110 \to 111$: cannot push	$\neg 111$	$\top$
$N = 2$					
6	$R_2 \wedge E$	SAT	$s = 111$	$\neg 111$	$\top$
7	$R_1 \wedge T \wedge s'$	SAT	predecessor $t = 110$	$\neg 111$	$\top$
8	$R_0 \wedge T \wedge t'$	UNSAT	add $\neg 110$ to R_1	$\neg 111, \neg 110$	$\top$
9	$R_2 \wedge E$	SAT	start over: $s = 111$	$\neg 111, \neg 110$	$\top$
10	$R_1 \wedge T \wedge s'$	UNSAT	both predecessors (110, 111) excluded: add $\neg 111$ to R_2	$\neg 111, \neg 110$	$\neg 111$
11	$R_2 \wedge E$	UNSAT	new frame R_3	$\neg 111, \neg 110$	$\neg 111$
12	push $\neg 110$ $R_1 \to R_2$: $R_1 \wedge T \wedge 110'$	UNSAT	only predecessor of 110 is 110: push $R_1 = R_2$: **terminate**	$\neg 111, \neg 110$	$\neg 111, \neg 110$

Note: In Wednesday's class, the question came up about why we do not push the literals from the initi clause in round 1. The simple answer is that because the algorithm as stated, has a condition $j > 0$ and only pushes from frame 1 up. It is sound to push from frame 0 up. As discussed, in this case, it will delay saturation by one frame. In general, this is avoided because the initial states often have very precise facts that are contrary to the spirit of "generic" invariants.