Fairness Constraints

Abhik Roychoudhury
CS 4271

A Lossy Channel in SMV

- MODULE lossy_chan(input)
- VAR output: boolean;
- ASSIGN
  - next(output) := {input, output};
- FAIRNESS
  - (input = 0 -> AF output = 0)
- FAIRNESS
  - (input = 1 -> AF output = 1)

CMU SMV syntax has been used in this slide.

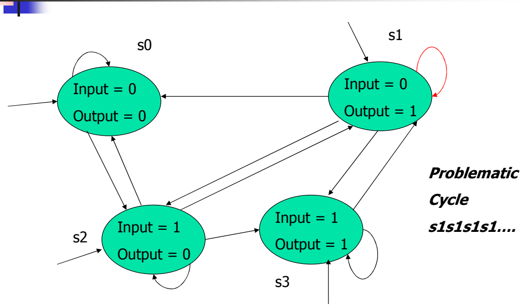
Fairness in SMV

- Specified as a CTL formula φ
  - SMV will define a path as unfair if φ is not true in it infinitely often
  - During Model Checking, the A and E quantifiers will be applied to fair paths only
  - In our example, this means
    - Eventual message delivery holds infinitely often.

Fairness Constraints

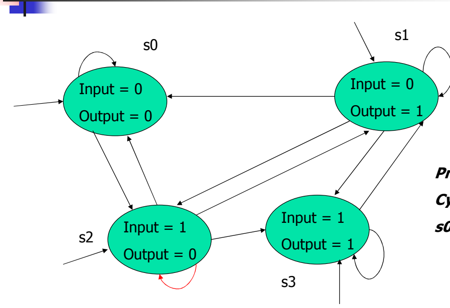
- We want to prove that a message is eventually delivered.
  - Channel does not drop forever.
  - Need to impose fairness constraint on channel.
- A fairness constraint marks some states as distinguished.
  - It forces a process to visit one of the distinguished states infinitely often.
  - Any execution trace which does this is considered a fair path.

Lossy channel



Problematic Cycle s1s1s1s1....

Lossy channel



Problematic Cycle s0s0s0s0....

Fairness

- A fairness constraint can be viewed as a set of states.
- A path in the Kripke structure is fair if it visits at least one member of this set infinitely often
- While model checking, we will restrict our attention to only the fair paths.

Fairness

- Given Kripke Structure M=(S,S0,→,L)
  - Each Fairness constraint is a set $S' \subseteq S$
  - A path π in the Kripke Structure satisfies the constraint if elements of S' appear infinitely often, i.e.
    - At least one element of S' must appear infinitely often in π
  - Model Checking is now restricted to ignore unfair computation traces while interpreting the path formulae in the property being verified.

Fair Kripke Structure

- M = (S, S0, R, L, F)

- S is set of states, R is transition relation and L is labelling function
- $F \subseteq 2^S$ is a set of fairness constraints.
- We consider only those paths which are fair w.r.t. each constraint in F.

These are called fair paths.

Fair CTL* semantics

- Meaning of path quantifiers must consider F.
  - $M, s \models_F Af$ iff for all fair paths π starting from s, $M, \pi \models_F f$
  - $M, s \models_F Ef$ iff there exists a fair path π starting from s, $M, \pi \models_F f$
- $M, s \models_F p$ iff $p \in L(s)$ and there exists a fair path from s.
- Meaning of the temporal operators $M, \pi \models_F \ldots$ does not change

Model Checking with fairness

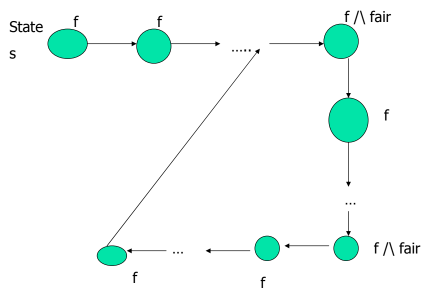
- $M,s \models_F$ EGf
  - There exists a fair path π starting from s which satisfies f globally.
  - For simplicity, assume a single fairness constraint represented by the formula fair. Thus
    - f holds globally in π
    - fair holds infinitely often in π
  - Not enough for an outgoing state to satisfy EGf
    - A state satisfying fair should hold in every finite segment

Comparison

- Without fairness
  - St_{EGf} = fixed point of
    - ***$func_f(Y) = f \wedge EX\ Y$***
  - starting from the set of all states as initial approximation
- With fairness
  - St_{EGf} = fixed point of
    - ***$func_{f,\ fair}(Y) = f \wedge E(f\ U\ (Y \wedge fair))$***
  - starting from the set of all states as initial approximation
  - The above equation leads to a symbolic model checking procedure using BDDs.

- Let us look at the pictorial description.

Pictorial representation



Fair CTL model checking

- What about EX, EU, $\neg$, $\vee$?
- Checking f $\vee$ g is not affected due to fairness constraints (involves a disjunction of BDDs)
  - Of course f, g might themselves involve EX, EU, EG the checking of which is affected due to fairness constraints.
- Similarly checking of $\neg$ f is not affected.
- For EX, EU
  - M, s $|=_F$ EX f iff M,s |= EX(f $\wedge$ fair)
  - M, s $|=_F$ E(f U g) iff M,s |= E(f U (g $\wedge$ fair))

Exercises

- Write down the modification to our symbolic MC computation for EX, EU.
- Write down the fairness constraints for our lossy channel explicitly
  - As a CTL* formula
  - As sets of states
- For the Kripke Structure of the lossy channel, are the following paths “fair” ?
  - s1 s3 s1 s3 s1 s3 s1 s3 …
  - s0 s2 s0 s2 s0 s2 s0 s2 …