LTL, Dekker's algorithm, …

K. V. S. Prasad
Dept of Computer Science
Chalmers University
Monday 3 Oct 2016

Questions?

- Student reps see me after class
- GU students other than IT program:
  - Meet in the break, nominate some reps

Linear Temporal Logic (LTL)

- From Huth + Ryan
  - ftp://ftp.cs.bham.ac.uk/pub/authors/M.D.Ryan/tmp/Anongporn/Ch1+3.pdf
  - Defn. 3.4 (p 186)
  - Fig. 3.5, defn. 3.5, defn 3.6 (p 188)
    - G means □ and F means ◊ (also used in Wikipedia LTL)
    - Can ignore X, U, W and R
      - But X (next) and U (until) are useful at least for LTL practice
  - Defn. 3.8, p. 190, and the sentence preceding.
  - These definitions establish
    - A propositional formula A can hold at a state s
    - An LTL formula can hold or fail for a path
    - An LTL formula holds for a state s if it holds for all paths from s

Why temporal logic?

- For safety claims, we can usually manage with
  - Assertions
    - In the CS for p, say “q is not in its CS”
  - Or a monitor process
    - With just one command, assert ¬(p in CS ^ q in CS)
    - Runs in parallel with p and q
      - So the assert can run any time, and SPIN will catch any run where it fails
- But liveness properties cannot in general be caught in this way
  - Though special cases such as termination might be caught by ad-hoc methods

Counterexamples

- For a safety statement (typically □A)
  - A state sn such ¬A holds at sn
  - This then yields
    - If π is a path that includes sn, then $\pi \not\models \Box A$, i.e. π does not satisfy □A
    - So if s is a state from which π runs, then $s \not\models \Box A$
- A liveness statement (typically ◊A) fails for s
  - A path π from s includes a loop, such that A does not hold in the loop or before it
- WARNING: □ and ◊ are duals so either can be used above. What is a counterexample depends on the content of the claim, safety or liveness, not on whether the outermost symbol is □ or ◊.

Temporal algebra

- distributes □ over ^, i.e., □A ^ □B iff □(A ^ B)
  - Both sides say that both A and B hold for t≥0
- Why doesn't □ distribute over v ?
  - □A v □B = either A holds from now on, or B does
  - □(A v B) = either A or B holds from now on
    - This is true in a system where only A holds after 1, 3, 5 … steps and only B holds after 0, 2, 4, 6 … steps. Then neither A nor B holds always
- ◊ distributes over v, i.e., ◊A v ◊B iff ◊(A v B)
  - Both say every path has a time when either A or B holds
- Why doesn't ◊ distribute over ^ ?
  - ◊A ^ ◊B = exist t1, t2 such that A(t1) and B(t2)
  - ◊(A ^ B) = exist t such that A^B holds at t

More temporal algebra

- $\neg\Box A = \Diamond\neg A$
    - $\Box A = \neg\Diamond\neg A$ (so we only need $\Diamond$)
- $\neg\Diamond A \equiv \Box\neg A$
- $\Box\Box A$ iff $\Box A$
- $\Diamond\Diamond A$ iff $\Diamond A$
  - For some r,s,t ≥0, lhs says A(r+s) and rhs says A(t)
- $\Diamond\Box\Diamond A$ iff $\Box\Diamond A$
  - Rhs = “A will be true infinitely often”
  - Lhs = “at some time, A will be true infinitely often”
- Sketched the ideas here. Formally, use the definitions 4.6 and 4.7 in the book (p72,73). Or better, use Ruth+Ryan.

Temporal algebra using X and U

- ◊A = true U A
  - Eventually, A becomes true

- X(A v B) = XA v XB
- X(A^B) = XA ^ XB
- ¬**X**A = **X**¬A

- □A = A ∧ **X**□A
- **◊**A = A ∨ **X**◊A

- **X** (A **U** B) = (**X** A) **U** (**X** B)
- A **U** B ≡ A **U** (A **U** B)
- A **U** B ≡ B ∨ (A ∧ **X**(A **U** B))

Mutex proof for Dekker's algorithm

- Abbreviations: ti means turn = i, wp=wantp
- Invariants (prove by induction)
  - □ t1 v t2   ()
  - (p3..p5 v p8..p10) iff   wp   similar for q
  - (p1 v p2 v p6 v p7) iff ¬wp   similar for q
  - p8 -> ¬wq (else, cannot pass while in p3)
- Imply mutex:
  - p8 ^ q8 iff wp ^ wq but p8 -> ¬wq

Dekker progress proof, 1
(variant of UTwente proof)

- To prove: $\Box$(p2 -> $\Diamond$p8)
  - Every path from a p2 will lead to a p8
- First, note that $\Box$(p2 -> $\Diamond$p3) by fairness
- Will show $\Box$(p3 -> $\Diamond$p8)
  - Case 1: $\Diamond\Box$q1 (q gets stuck in NCS)
    - q1 iff ¬wq, so $\Box$q1 -> $\Box$ ¬wq
    - $\Box$(p3 ^ $\Box$q1) => $\Box$(p3 ^ $\Box$¬wq) => $\Box$ $\Diamond$ p8

      by while loop

Dekker progress proof, 2 (variant of UTwente proof)

- To show □(p3 -> ◊p8), continued
  - Case 2: □ ◊¬q1
    - the other case, q leaves NCS
    - Proof by contradiction, assume p3 ^ ¬p8, i.e., □p3..p7
    - Lemma1: □ ◊t1
      - Again, by contradiction, assume □ t2
      - □(p3..p7 ^ t2) => ◊□p6

        => ◊¬wp

        => q9 (by progress of q)

        => t1 (Contradiction!)

      So □ p3..p7 => ◊t1

Dekker progress proof, 3 (variant of UTwente proof)

- To show □(p3 -> <> p8) continued
  - Case 2: □◊¬q1 continued
  - □p3..p7 => ◊t1 prev page
    - => ◊□t1 (never reach p9)
    - => ◊□(p3 v p4) (p3..p7)
    - => ◊□wp (by invariant)
    - => ◊□q6
    - => ◊□¬wq (also by invariant)
    - => ◊p8 (contradiction!)
  - Hence ¬□p3..p7 and □(p3 ^ □◊¬q1 => ◊p8)
  - Putting both cases together (q and NCS), □(p3->◊p8)

Notes on Dekker

- An alternative approach might be to try to improve the proof in the textbook. Reformulate the correct but unusable statement in the middle of p 81
  p4^□(turn=2)->◊p5
- What do we need instead of the □(turn=2)?

On progress proofs

- Delicate (many cases, did we miss any?)
- Labour intensive
- Error prone (even Ben-Ari's book?)
- Need machine check
- Then why study them at all by hand?
  - To know what to assert
    - Build the right system
    - The system will check that the system is built right