Systems and Software Verification

# Part II. Specifying with Temporal Logic

Lecturer: JUNBEOM YOO
jbyoo@konkuk.ac.kr
http://dslab.konkuk.ac.kr

Ver. 2.0

# Introduction

- Writing the temporal logic formulas expressing desired system properties

- 4 classification of verification goals
  1. Reachability property
     - Some particular situation *can be reached*.
  2. Safety property
     - Under certain condition, something *never occurs*.
  3. Liveness property
     - Under certain condition, something *will ultimately occur*.
  4. Fairness property
     - Under certain condition, something will (or not) occur *infinitely often*.

  \+ Deadlock freeness

  \+ Abstraction methods

# Chapter 6. Reachability Properties

- Reachability property
  - Some particular situation can be reached.

  - Examples:
    - (R1) " We can obtain n<0 "
    - (R2) " We can enter a critical section " ← simple
    - (R3) " We cannot have n<0 "
    - (R4) " We cannot reach the crash state " ← negation of the simple
    - (R5) " We can enter the critical section without traversing n=0 " ← with conditional restricts
    - (R6) " We can always return to the initial state " ← stronger / nested
    - (R7) " We can return to the initial state "

- Organization of Chapter 6
  - Reachability in Temporal Logic
  - Model Checkers and Reachability
  - Computation of the Reachability Graph

# 6.1 Reachability in Temporal Logic

- EF $\Phi$
  - " There exists a path from the current state along which some state satisfying $\Phi$ "
  - (R1) " We can obtain n<0 "
    - EF (n<0)
  - (R2) " We can enter a critical section "
    - EF crit_sec
  - (R3) " We cannot have n<0 "
    - ¬EF (n<0)
  - (R4) " We cannot reach the crash state "
    - ¬EF crash
    - AG ¬crash
    - " Along every path, at any time, ¬crash "
  - (R5) " We can enter the critical section without traversing n=0 "
    - E (n≠0) U crit_sec
    - " There exists a path along which n ≠ 0 holds until crit_sec becomes true. "
  - (R6) " We can always return to the initial state "
    - AG ( EF init )
  - (R7) " We can return to the initial state "
    - EF init

# 6.2 Model Checkers and Reachability

- Reachability properties are typically the easiest to verify.
- All model checkers can answer it in principle by simply examining their reachability graph.

- But they do vary in richness.
  - conditional reachability
  - nested reachability
  - etc.

- Design/CPN is specifically designed for reachability property verification.

# 6.3 Computation of the Reachability Graph

- The effective construction of set of reachable states are non-trivial.
  - Several automata are synchronized.

- Algorithms dealing with reachability problems
  1. Forward chaining
  2. Backward chaining
  3. "On-the-fly" exploration

- Forward chaining
  - A natural approach
  - from initial states → add their successors → until saturation
  - Difficulty: potential explosion of the set constructed

- Backward chaining
  - from target states → add immediate predecessors → until saturation
  - then, test whether some initial states are in there (like $pre^*(S)$ in Section 4.1)
  - Drawback
    1. Target states need to be fixed before.
    2. Computing immediate predecessors is generally more complicated than that of successors.

- “On-the-fly” exploration
  - Explore the reachability graph without actually building it
  - Construction is performed partially, as the exploration proceeds, without remembering everything already visited.
  - Background assumption
    - Present-day computers are more limited in memory resources than in processing speed
  - It is efficient mostly when
    1. Target set is indeed reachable. (“Yes” requires no exhaustive explorations)
    2. Can operate in forward or backward manners (The forward is the traditional)
    3. May apply to some systems with infinitely many states