Adapted by J-Pierre Corriveau from:

UCM-Based Generation of Test Goals

uOttawa

Daniel Amyot, University of Ottawa
(with Michael Weiss and Luigi Logrippo)
damyot@site.uottawa.ca

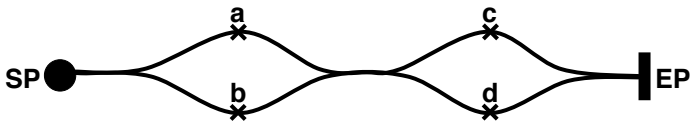
RDA Project (funded by NSERC)

Test Generation Approaches

- Based on UCM Testing Patterns (TPs)
  - Grey-box test selection strategies, applied to requirements scenarios
  - Manual
- Based on UCM Scenario Definitions
  - UCM + simple data model, initial values and start points, and path traversal algorithms
  - Semi-automatic
- Based on UCM Transformations
  - Exhaustive traversal
  - Mapping to formal language (e.g., LOTOS)
  - Automated

2

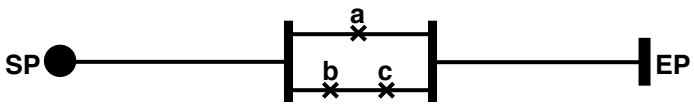
TP1: Alternatives



- 1A: *All results* (end points)
  {<SP, a, c, EP>}
- 1B: *All segments*
  {<SP, a, c, EP>, <SP, b, d, EP>}
- 1C: *All paths*
  {<SP, a, c, EP>, <SP, a, d, EP>,
  <SP, b, c, EP>, <SP, b, d, EP>}
- 1D: *All combinations of sub-conditions*
  (for composite conditions, e.g., *(X OR Y) AND Z*)

3

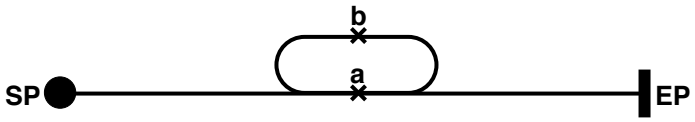
TP2: Concurrency



- 2A: *One combination (wrt ordering)*
  {<SP, a, b, c, EP>}
- 2B: *Some combinations*
  {<SP, a, b, c, EP>, <SP, b, a, c, EP>}
- 2C: *All combinations*
  {<SP, a, b, c, EP>, <SP, b, a, c, EP>,
  <SP, b, c, a, EP>}

4

TP3: Loops



- 3A: *All segments*
- 3B: *At most k iterations*
- 3C: *Valid boundaries [low, high]*
  Tests *low*, *low*+1, *high*-1, and *high*
- 3D: *All boundaries [low, high]*
  Tests valid ones (3C) and invalid ones (*low*-1 and *high*+1, for rejection tests)

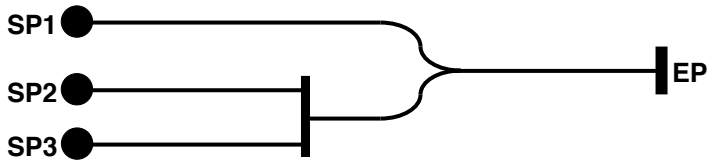
5

Flattening the Loops

- 3A: *All segments:*
  - *{<SP, a, b, a, EP>}*
- *3B: At most k iterations:*
  - *{<SP, a, EP>, <SP, a, b, a, EP>, <SP, a, b, a, b, a, EP>} (if k = 2)*
- *3C: Valid boundaries [low, high]: Tests low, low+1, high-1, and high. If low = 1 and high = 5:*
  - *{<SP,a,b,a,EP>, <SP,a,b,a,b,a,EP>, <SP,a,b,a,b,a,b,a,b,a,EP>, <SP,a,b,a,b,a,b,a,b,a,b,a,EP>}*
- *3D: All boundaries [low, high]: Tests valid ones (3C) and invalid ones (low-1 and high+1). If low = 1 and high = 5:*
  - *Accept: {<SP,a,b,a,EP>, <SP,a,b,a,b,a,EP>, <SP,a,b,a,b,a,b,a,b,a,EP>, <SP,a,b,a,b,a,b,a,b,a,b,a,EP>}*
  - *Reject: {<SP,a,EP>, <SP,a,b,a,b,a,b,a,b,a,b,a,b,a,EP>}*

6

TP4: Multiple Start Points



Strategies based on necessary, redundant, insufficient, and racing subsets (8 strategies based on path sensitization)

Case #	SP1	SP2	SP3	SP1 ∨ (SP2 ∧ SP3)	*Subset*
0	F	F	F	F	Insufficient stimuli. Not interesting.
1	F	F	T	F	Insufficient stimuli
2	F	T	F	F	Insufficient stimuli
3	F	T	T	T	Necessary stimuli
4	T	F	F	T	Necessary stimuli
5	T	F	T	T	Redundant stimuli
6	T	T	F	T	Redundant stimuli
7	T	T	T	T	Racing stimuli

7

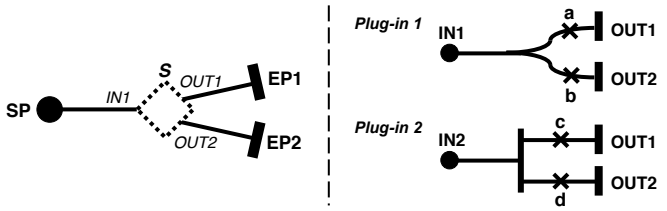
Eight strategies for start points

Based on necessary, redundant, insufficient, and racing subsets of inputs:

- 4A: *One necessary subset, one goal:*
  - *{<SP2, SP3, EP>} (if case 3 is selected)*
- *4B: All necessary subsets, one goal:*
  - *{<SP2, SP3, EP>, <SP1, EP>} (assume interleaving)*
- *4C: All necessary subsets, all goals:*
  - *{<SP2, SP3, EP>, <SP3, SP2, EP>, <SP1, EP>}*
- *4D: One redundant subset, one goal:*
  - *{<SP1, SP2, EP>}*
- *4E: All redundant subsets, one goal:*
  - *{<SP1, SP2, EP>, <SP3, SP1, EP>}*
- *4F: One insufficient subset, one goal:*
  - *{<SP2, EP>} (rejection)*
- *4G: All insufficient subsets, one goal:*
  - *{<SP3, EP>, <SP2, EP>} (rejection)*
- *4H: Some racing subsets, some goals:*
  - *{<SP1, SP3, SP2, EP, EP>, <SP2, SP3, SP1, EP, EP>}*

8

TP5: Single Stub



Flattens one stub from a hierarchical UCM.
Strategies:

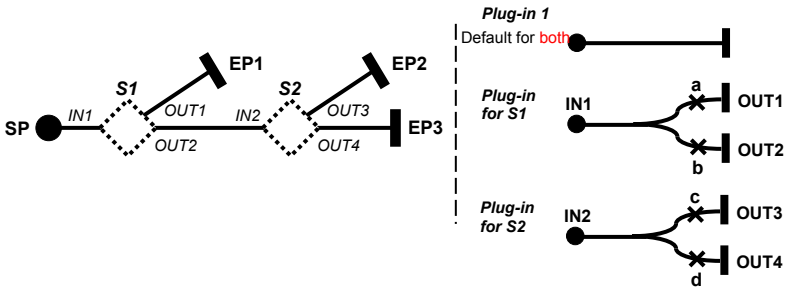
- 5A: *Static flattening* (when only one plug-in in the static stub)
- 5B: *Dynamic flattening, some plug-ins* (when several plug-ins in the dynamic stub)
- 5C: *Dynamic flattening, all plug-ins* (when several plug-ins in the dynamic stub)

Assuming all branch coverage for each plug-in:

- 5A: 2 paths {<SP, a, EP1>, <SP, b, EP2>} *IF S were static*
- 5B: 2 paths { <SP, c, EP1>, <SP, d, EP2>} or same as 5A (ie test 1 of the 2 plugins)
- 5C: 4 paths {<SP, a, EP1>, <SP, b, EP2>, <SP, c, EP1>, <SP, d, EP2>}

9

TP6: Causally-Linked Stubs



Handles combinations of plug-ins bound to causally linked stubs.
→ causal: selection of plug-in for 2nd stub depends on (is caused by) 1st stub's selection of plug-in

Strategies:

- 6A: *Default behavior* (when no feature is active)
- 6B: *1 combination*
- 6C: *several or all combinations…*

10