EVALUATION OF CAUSE EFFECT GRAPHS BY PETRI NETS

CONTENTS

- introduction
  - What are cause effect graphs?
  - standard evaluation procedure

- basic fault trees & Petri nets
  - real trees

- advanced fault trees & Petri nets
  - problem: negation
  - problem: branching places

- cause effect graphs & Petri nets
  - example [Myers 1979]
  - example [Liggesmeyer 2002]

- summary
WHAT ARE CAUSE EFFECT GRAPHS?

-> EXAMPLE [Myers 1979, p. 58]

- verbal specification
  The character in column 1 must be an “A” or a “B”.
  The character in column 2 must be a digit.
  In this situation, the file update is made.
  If the first character is incorrect, message X12 is issued.
  If the second character is not a digit, message X13 is issued.

- causes
  1 character in column 1 is “A”
  2 character in column 1 is “B”
  3 character in column 2 is a digit

- effects
  70 file update, update message -> effect1
  71 message X12 is issued -> effect2
  72 message X13 is issued -> effect3

- cause-effect graph

STANDARD EVALUATION PROCEDURE, BASICS

- objective
  -> to get a characteristic set of abstract test cases

- compare
  -> [Myers 1979]
  -> [Liggesmeyer 2002]

- Select an effect to be present (TRUE).

- Trace back through the graph, and find all essential combinations of causes that will set this effect to TRUE.

- Doing so, consider suitable heuristics (next slide).
  -> to be efficient
  -> to eliminate situations that tend to be low-yield test cases

- Create a line in the decision table for each combination of causes.
  -> each line stands for a test case

- Determine the states of all other effects.

- Eliminate doubled lines in decision table.
STANDARD EVALUATION PROCEDURE, HEURISTICS

- remember: backward procedure

- if \( x \) then
  - enumerate all situations, where one input is TRUE & all other inputs are FALSE
  - else set all inputs to FALSE
  - endif

- if \( x \) then
  - set all inputs to TRUE
  - else enumerate all situations, where one input is FALSE & all other inputs are TRUE
  - endif

AN ALTERNATIVE APPROACH

SUPPORTING

- \( \rightarrow \) ANIMATION
- \( \rightarrow \) AUTOMATIC COMPUTATION

DEFINING

- \( \rightarrow \) A NEW COVERAGE MEASURE
BASIC FAULT TREES

EX1:

- minimal cuts ?

EX2:

- minimal runs (T-invariants) ?

BASIC FAULT TREES, EX1

-> T-INVARINTS
BASIC FAULT TREES, EX2
-> T-INVARINTS

OBSERVATIONS

- (minimal) cut:
  - (minimal) set of basic events
  - resulting into the top event

- (minimal) T-invariant:
  - (minimal) multiset of transitions
  - with zero total effect on marking
  - reproducing a given marking
  - potentially cyclic behaviour

- minimal T-invariants /cuts:
  - minimal runs
  - basic behaviour

- any behaviour is a non-negative linear combination of basic runs

- (minimal) cuts <-> (minimal) T-invariants <-> (minimal) test case

- CTI - Covered by T-Invariants:
  - each transition belongs to a (minimal) T-invariant
  - each transition contributes to system behaviour

- decomposition into minimal [ cuts / T-invariants / test cases ]
  - node / branch coverage
  - basic behaviour coverage
ADVANCED FAULT TREES, EX1

-> PROBLEM: NEGATION

ADVANCED FAULT TREES, EX1

-> T-INVARINTS
ADVANCED FAULT TREES, EX2
-> PROBLEM: BRANCHING PLACES

ADVANCED FAULT TREES, EX2
-> T-INVARINTS
CAUSE EFFECT GRAPH, [MYERS 1979]  
-> EVALUATION OF TEST CASES

- T-invariant 1 -> test case 1:
  - abstract test case: not_e3, don't-care: e1/e2 -> effect3
  - real test case: A, A -> X13 message

- T-invariant 2 -> test case 2:
  - abstract test case: not_e1 and not_e2, don't-care: e3 -> effect2
  - real test case: C, 1 -> X12 message

- T-invariant 3 -> test case 3:
  - abstract test case: e1 and e3 -> effect1
  - real test case: A, 1 -> update message

- T-invariant 4 -> test case 4:
  - abstract test case: e2 and e3 -> effect1
  - real test case: B, 1 -> update message

- these four test cases guarantee basic behaviour coverage

- don’t care’s: prefer TRUE assignment;
  - -> to avoid fault masking

- THESE ARE EXACTLY THE FOUR TEST CASES
  WE GET BY THE STANDARD EVALUATION PROCEDURE

---

CAUSE EFFECT GRAPH, [LIGGESMEYER 2002]

- T-invariant 1 -> test case 1:
  - abstract test case: not_e3, don't-care: e1/e2 -> effect3
  - real test case: A, A -> X13 message

- T-invariant 2 -> test case 2:
  - abstract test case: not_e1 and not_e2, don't-care: e3 -> effect2
  - real test case: C, 1 -> X12 message

- T-invariant 3 -> test case 3:
  - abstract test case: e1 and e3 -> effect1
  - real test case: A, 1 -> update message

- T-invariant 4 -> test case 4:
  - abstract test case: e2 and e3 -> effect1
  - real test case: B, 1 -> update message

- these four test cases guarantee basic behaviour coverage

- don’t care’s: prefer TRUE assignment;
  - -> to avoid fault masking

- THESE ARE EXACTLY THE FOUR TEST CASES
  WE GET BY THE STANDARD EVALUATION PROCEDURE
CAUSE EFFECT GRAPH, [LIGGESMEYER 2002]

- T-INvariants 1, 2

resolving of branching places
CAUSE EFFECT GRAPH, [LIGGESMEYER 2002]
-> T-INVARIANTS 3, 4

CAUSE EFFECT GRAPH, [LIGGESMEYER 2002]
-> EVALUATION OF TEST CASES

- **T-invariant 1** -> test case 1:
  abstract test case: not_e3, don’t-care: e1/e2 -> effect4
  real test case: total = MAXINT; B

- **T-invariant 2** -> test case 2
  abstract test case: not_e1 and not_e2, don’t care: e3 -> effect4
  real test case: total < MAXINT; 0

- **T-invariant 3** -> test case 3
  abstract test case: e1 and e3 -> effect1, effect3
  real test case: total < MAXINT; B

- **T-invariant 4** -> test case 4
  abstract test case: e2 and e3 -> effect1, effect2, effect3
  real test case: total < MAXINT; A

- these four test cases guarantee basic behaviour coverage
- again: don’t care’s get TRUE assignment;
- standard evaluation procedure splits test case 1 into two cases:
  e1 and not_e3 (and not_e2) -> effect4
  e2 and not_e3 (and not_e1) -> effect4
- compare [Liggesmeyer 2002, p. 68]
FINAL QUESTION

How TO COMPUTE MINIMAL T-INVARInANTS?

-> BASICS OF PETRI NET THEORY [LAUTENBACH 1973]

-> RELIABLE TOOL SUPPORT AVAILABLE, E. G. CHARLIE

SUMMARY

- cause effect graphs can be represented adequately by Petri nets
- straightforward transformation -> automatic translation
- minimal T-invariants in Petri net representation correspond to minimal abstract test cases in cause effect graph representation
  -> input transitions - causes
  -> output transitions - effects
- covering by T-invariants corresponds to covering by abstract test cases
  -> BASIC BEHAVIOUR COVERAGE
- computation of all minimal T-invariants
  -> there can be exponentially many
  -> reliable tool support available, e. g. Charlie (inspired by INA)
REFERENCES

❑ K Lautenbach:
   Exakte Bedingungen der Lebendigkeit für eine Klasse von Petrinetzen.
   Berichte der GMD 82, Bonn 1973 (in German).

❑ P Liggesmeyer:
   Software-Qualität; Testen, Analysieren und Verifizieren von Software;
   Spektrum 2002 (in German).

❑ GJ Myers:
   The Art of Software Testing;
   Wiley & Sons 1979.

❑ KH Pascoletti:
   Diophantische Systeme und Lösungsmethoden zur Bestimmung aller
   Invarianten in Petri-Netzen;
   Berichte der GMD 160, Bonn 1986 (in German).

❑ S Roch, PH Starke:
   INA - Integrated Net Analyser, Version 2.2;

❑ M Heiner, M Schwarick and J Wegener:
   Charlie – an extensible Petri net analysis tool;