Akademia Górniczo-Hutnicza im. Stanisława Staszica

Kraków

Generator specyfikacji logicznej

Autorzy:
Marcin Jędrzejczyk
Paweł Ogorzały

Prowadzący:
Dr inż. Radosław Klimek

19 maja 2016

1 Cel projektu

Celem projektu jest wytworzenie programu, który dla podanego diagramu będzie w stanie go sparsować do formatu pozwalającego na wygenerowanie specyfikacji logicznej.

2 Powód tworzenia generatora

- Ręczne tworzenie specyfikacji logiki jest trudne dla niedoświadczonych w tym użytkowników.
- Formalna weryfikacja modelu oprogramowania pozwala obniżyć koszty i zwiększyć niezawodność.
- Brak takich narzędzi.

3 Ważne

- Diagram aktywności musi składać się z wcześniej zdefiniowanych wzorców, zagnieżdżanie jest dozwolone.
- Diagram aktywności składa się tylko z atomicznych aktywności, zidentyfikowanych podczas tworzenia scenariuszy przypadków użycia.
- Generator musi działać automatycznie, usuwa to błąd ludzki.

4 Algorytmy

Wzorce przepływu:

- Sekwencja, sequence
- Współbieżność, concurrent fork/join
- Pętla while, loop while
- Rozgałęzienie, branching

Wyrażenie logiczne W_L jest strukturą stworzoną według poniższych zasad:

- każdy elementarny zbiór $pat(a_i)$, gdzie $i > 0$ i każde a_i jest formułą atomiczną, jest wyrażeniem logicznym,
- każde $pat(A)$, gdzie $i > 0$ i każde A_i jest albo
  - atomiczną formułą lub
  - logicznym wyrażeniem $pat()$

  także jest wyrażeniem logicznym.

Wstępny algorytm:

1. Analiza diagramów aktywności w celu wyciągnięcia z nich wcześniej zdefiniowanych wzorców przepływu.
2. Przetłumaczenie wyłuskanych wzorców na wyrażenia logiczne W_L.
3. Generowanie specyfikacji logicznej L z wyrażeń logicznych,

Algorytm Π generujący specyfikację logiczną :

1. Na początku specyfikacja jest pusta, np. L=ø;
2. Najbardziej zagnieżdżone wzorce są przetwarzane jako pierwsze, a następnie mniej zagnieżdżone;
3. Jeśli obecnie analizowany wzorzec składa się wyłącznie z formuł atomicznych, specyfikacja logiczna jest rozszerzana, poprzez sumowanie zbiorów, których formuły są złączone z obecnie analizowanym wzorcem pat(), np. $L = L \cup pat()$;
4. Jeżeli jakiś argument jest wzorem sam w sobie to:

   - po pierwsze formuła $f1$, a potem
   - formuła fk

   tego wzoru(jeśli jakiegoś), lub w innym wypadku wziąć pod uwagę tylko najbardziej zagnieżdżony daleko? na lewo lub prawo,odpowiednio, są podstawiane osobno w miejsce wzorca jako argument.

5 Przykłady

Podane wzorce:

- Sequence(a,b)
  - ◇a=>b
  - a=> ◇b
  - □ (a & b)
- Concurrency(a,b,c,d)
  - a=> ◇b & ◇c
  - a=> (◇b & ◇c)
  - b & c => ◇d
  - (b & c) => ◇d
  - □ (a & (b | c))
  - □ ((b | c) & d)
  - □ (a & d)
- Branching(a,b,c,d)

- a => (◇b & ◇c) | (◇b & ◇c)
- a=> ((◇b & ◇c) | (◇b & ◇c))
- b | c => ◇d
- (b | c) => ◇d
- □ (a & d)
- □ (b & c)
- □ (a & (b | c))
- □ ((b | c) & d)

- LoopWhile(a,b,d,d)
  -

Wyjście programu dla:

- W_L = Seq(Seq(a,b),c) to:
  $L = \{\diamond a \Rightarrow b,\ a \Rightarrow \diamond b, \Box\ (a \& b)\} \cup$
  $\cup \{\diamond a \Rightarrow c,\ a \Rightarrow \diamond c, \Box\ (a \& c)\} \cup$
  $\cup \{\diamond b \Rightarrow c,\ b \Rightarrow \diamond c, \Box\ (b \& c\}$

6 Pseudokod

Input: Wyrażenie logiczne W_L, zdefiniowane wzorce przepływu P
Output: Specyfikacja logiczna L

$L := 0$
for $l := max(W'_L)$ **to** 1 **do**
  $p := getPat(W'_L, l)$;
  repeat
    if pattern p consists only atomic formulas **then**
      $L := L \cup p.pat()$
    end if
    if any argument of the p is a pattern itself **then**
      Specification L' for every combination $C_i = 1, ..n$, i.e. $L'(C_i)$, are calculated considering $ini-$ and fin-expressions for every non-atomic arguments and substituting consolidated expressions in places of these patterns as arguments, i.e.
      $L := L \cup L'(C_i)$
    end if
    $p := getPat(W'_L, l)$
  until p is empty
end for

7 Literatura

Radosław Klimek: From Extraction of Logical Specifications to Deduction-Based Formal Verification of Requitements Models. Strony 61-75.