Akademia Górniczo-Hutnicza im. Stanisława Staszica

Kraków

Generator specyfikacji logicznej

Autorzy:
Marcin Jędrzejczyk
Paweł Ogorzały

Prowadzący:
Dr inż. Radosław Klimek

5 czerwca 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 P:

- Sequence(f1,f4)
  - f1 $\Rightarrow \diamond$f4
  - $\neg$f1 $\Rightarrow \neg \diamond$f4
  - $\Box \neg($ f1 $\wedge$ f4)
- Concurrency(f1,f2,f3,f4)
  - f1 $\Rightarrow \diamond$f2 $\wedge \diamond$f3
  - $\neg$f1 $\Rightarrow \neg($ $\diamond$f2 $\wedge \diamond$f3)
  - f2 $\wedge$ f3 $\Rightarrow \diamond$f4
  - $\neg$(f2 $\wedge$ f3) $\Rightarrow \neg \diamond$f4
  - $\Box \neg$(f1 $\wedge$ (f2 $\vee$ f3))
  - $\Box \neg$((f2 $\vee$ f3) $\wedge$ f4)
  - $\Box \neg$(f1 $\wedge$ f4)
- Branching(f1,f2,f3,f4)

- f1 $\Rightarrow$ ($\diamond$f2 $\wedge$ $\neg$ $\diamond$f3) $\vee$ ($\neg$ $\diamond$f2 $\wedge$ $\diamond$f3)
- $\neg$f1 $\Rightarrow$ $\neg$(($\diamond$f2 $\wedge$ $\neg$ $\diamond$f3) $\vee$ ($\neg$ $\diamond$f2 $\wedge$ $\diamond$f3))
- f2 $\vee$ f3 $\Rightarrow$ $\diamond$f4
- $\neg$(f2 $\vee$ f3) $\Rightarrow$ $\neg$ $\diamond$f4
- $\Box$ $\neg$(f1 $\wedge$ f4)
- $\Box$ $\neg$(f2 $\wedge$ f3)
- $\Box$ $\neg$(f1 $\wedge$ (f2 $\vee$ f3))
- $\Box$ $\neg$((f2 $\vee$ f3) $\wedge$ f4)

- LoopWhile(a,b,c,d)
  - f1$\Rightarrow$ $\diamond$f2
  - $\neg$f1$\Rightarrow$ $\neg$ $\diamond$f2
  - f2 $\wedge$ c(f2) $\Rightarrow$ $\diamond$c $\wedge$ $\neg$ $\diamond$f4
  - $\neg$(f2 $\wedge$ c(f2)) $\Rightarrow$ $\neg$($\diamond$f3 $\wedge$ $\neg$ $\diamond$f4)
  - f2 $\wedge$ $\neg$c(f2) $\Rightarrow$ $\neg$ $\diamond$f3 $\wedge$ $\diamond$f4
  - $\neg$(f2 $\wedge$ $\neg$c(f2) $\Rightarrow$ $\neg$($\neg$ $\diamond$f3 $\wedge$ $\diamond$f4)
  - f3 $\Rightarrow$ $\diamond$f2
  - $\neg$f3$\Rightarrow$ $\neg$ $\diamond$f2
  - $\Box$ $\neg$(f1 $\wedge$ f2)
  - $\Box$ $\neg$(f1 $\wedge$ f3)
  - $\Box$ $\neg$(f1 $\wedge$ f4)
  - $\Box$ $\neg$(f2 $\wedge$ f3)
  - $\Box$ $\neg$(f2 $\wedge$ f4)
  - $\Box$ $\neg$(f3 $\wedge$ f4)

Wyjście programu dla:

- W_L = Concur(a,b,c,d) to:
  $L = \{a \Rightarrow \diamond b \wedge \diamond c,\ \neg a \Rightarrow \neg(\diamond b \wedge \diamond c),\ b \wedge c \Rightarrow \diamond d,\ \neg(b \wedge c) \Rightarrow \neg \diamond d,\ \Box\neg(a \wedge (b \vee c)),$
  $\Box\neg((b \vee c) \wedge d),\ \Box\neg(a \wedge d)\}$

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

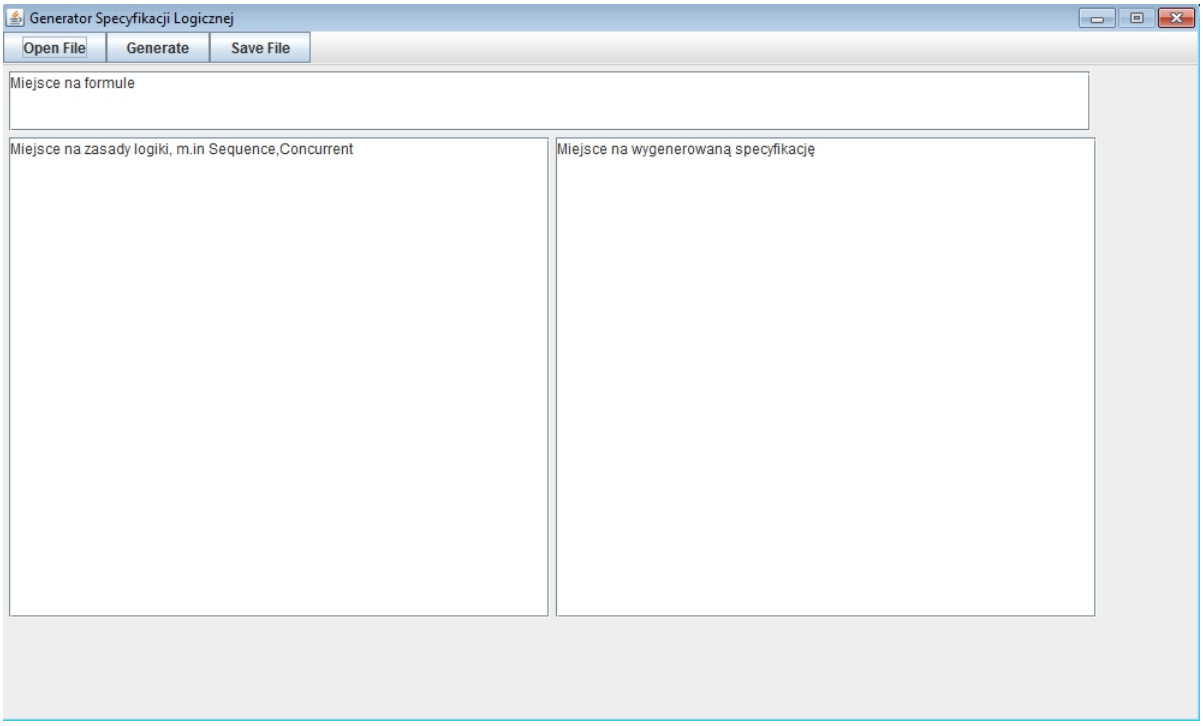
- Branch(Seq(a,b),c,d,e) to:
  $L =$
  $\{a \Rightarrow \diamond b, \neg a \Rightarrow \neg \diamond b, \Box\neg(a \wedge b)\} \cup\{a \Rightarrow (\diamond c \wedge \neg \diamond d) \vee (\neg \diamond c \wedge \diamond d),\ \neg a \Rightarrow \neg((\diamond c \wedge \neg \diamond d) \vee$
  $\vee(\neg \diamond c \wedge \diamond d)),\ c \vee d \Rightarrow \diamond e,\ \neg(c \vee d) \Rightarrow \neg \diamond e\ ,\ \Box\neg(a \wedge e)\ ,\ \Box\neg(c \wedge d),\ \Box\neg(a \wedge (c \vee d)),$
  $\Box\neg((c \vee d) \wedge e)\} \cup\{b \Rightarrow (\diamond c \wedge \neg \diamond d) \vee (\neg \diamond c \wedge \diamond d),\ \neg b \Rightarrow \neg((\diamond c \wedge \neg \diamond d) \vee (\neg \diamond c \wedge \diamond d)),$
  $c \vee d \Rightarrow \diamond e,\ \neg(c \vee d) \Rightarrow \neg \diamond e\ ,\ \Box\neg(b \wedge e)\ ,\ \Box\neg(c \wedge d),\ \Box\neg(b \wedge (c \vee d)),\ \Box\neg((c \vee d) \wedge e)\}$

6 Pseudokod

Input: Wyrażenie logiczne W_L, zdefiniowane wzorce przepływu(nie puste) P
Output: Specyfikacja logiczna L

$L := 0$
for $i := 1$ **to** $|W_L|$ **do**
  $L2 := W_L[i]P' \backslash \{W_L[i]'P.f_{en}, W_L[i]'P.f_{ex}\}$;
  for $j := 1$ **to** $|W_L[i]|$ **do**
    if $W_L[i] \uparrow a_j$ is non-atomic then
      $agg := (W_L[i]'P \uparrow a_J())^e + "V" + (W_L[i]'P \uparrow a_J())^x$;
      replace in L2 every pattern $W_L[i]'P \uparrow a_j()$ by agg;
    end if
  end for
  L:=L $\cup$ L2
end for

7 Wygląd GUI



Rysunek 1: Interfejs graficzny

8 Literatura

Radosław Klimek: From Extraction of Logical Specifications to Deduction-Based Formal Verification of Requitements Models. Strony 61-75.
Radosław Klimek: Elicitation of logical specifications from RUP-like processes for formal verification.