Programowanie w Logice

Przykłady programów

Przemysław Kobylański

Przykłady programów

Analiza pseudokodu

- Analizować będziemy poprawność programów w pseudokodzie podobnym do języka Ada (bez pustej instrukcji `null`).
- Analiza opiera się na następującym założeniu:

  Jeśli program jest niepoprawny, to jego złe działanie można zaobserwować już dla najprostszych danych.
- Przez „proste dane" rozumiemy takie dane, przy których program wykonuje się po stosunkowo krótkich ścieżkach wykonania.
- Ścieżka wykonania, to ciąg instrukcji zmieniających stan obliczeń[1] jakie wykonywane są podczas działania programu.

[1] Instrukcji podstawienia.

Przykłady programów

Analiza pseudokodu

Example (Sumowanie od 1 do n)

```
sum := 0; i := 0;
while i <= n loop
  sum := sum + i; i := i + 1;
end loop;
```

Coraz dłuższe ścieżki wykonania:

```
sum:=0; i:=0;
sum:=0; i:=0; sum:=sum+i; i:=i+1;
sum:=0; i:=0; sum:=sum+i; i:=i+1; sum:=sum+i; i:=i+1;
sum:=0; i:=0; sum:=sum+i; i:=i+1; sum:=sum+i; i:=i+1; sum:=sum+i; i:=i+1;
...
```

Powyższe ścieżki zapisać możemy w postaci wyrażenia regularnego:

sum := 0; i := 0; (sum := sum + i; i := i + 1;)*

Przykłady programów

Analiza pseudokodu

- Analiza programu opiera się na logice Hoare'a i pojęciu *najsłabszego warunku wstępnego*.
- Najsłabszym warunkiem wstępnym (ang. *weakest precondition*) $wp(P, C)$ dla programu P i warunku końcowego C, jest taki warunek, że jeśli dane go spełniają, to po wykonaniu programu P będzie spełniony warunek końcowy C.
- Logika Hoare'a mówi nam między innymi, że

$$\begin{aligned} wp(x := e, C) &= C[x/e], \text{ (operacja substytucji)} \\ wp(S_1; S_2, C) &= wp(S_1, wp(S_2, C)) \end{aligned}$$

Przykłady programów

Analiza pseudokodu

Example (Niepoprawny SWAP)

$wp(x := y; y := x, x = a \wedge y = b) =$
$wp(x := y, wp(y := x, x = a \wedge y = b)) =$
$wp(x := y, x = a \wedge x = b) =$
$y = a \wedge y = b$

Dla $a \neq b$ powyższy warunek jest sprzeczny, czyli nie ma takich danych aby po wykonaniu sekwencji dwóch podstawień zachodził warunek końcowy $x = a \wedge y = b$.

Przykłady programów

Analiza pseudokodu

Example (Poprawny SWAP)

$wp(t := x; x := y; y := t, x = a \wedge y = b) =$
$wp(t := x, wp(x := y, wp(y := t, x = a \wedge y = b))) =$
$wp(t := x, wp(x := y, x = a \wedge t = b)) =$
$wp(t := x, y = a \wedge t = b) =$
$y = a \wedge x = b$

Zatem, jeśli początkowo $x = b \wedge y = a$ (dla dowolnych wartości a i b), to po wykonaniu ciągu podstawień `t := x; x := y; y := t` będzie spełniony warunek końcowy $x = a \wedge y = b$.

Przykłady programów

Analiza pseudokodu

- Generując ścieżki dodatkowo będziemy umieszczać na nich asercje zawierające warunki jakie musiały być spełnione aby program poszedł taką ścieżką:
  1. Jeśli program wszedł do wnętrza pętli `while C loop S end loop`, to dołączona jest asercja C.
  2. Jeśli program opuścił pętlę `while C loop S end loop`, to dołączona jest asercja $\neg C$.
  3. Jeśli program wszedł do części `then` instrukcji `if C then S1 else S2 end if`, to dołączana jest asercja C.
  4. Jeśli program wszedł do części `else` instrukcji `if C then S1 else S2 end if`, to dołączana jest asercja $\neg C$.

Przykłady programów

Analiza pseudokodu

Example (Jednokrotne wykonanie pętli)

```
sum := 0; i := 0;
while i <= n loop
   sum := sum + i; i := i + 1;
 end loop;
```

Ścieżka przy jednokrotnym wykonaniu pętli:

```
  sum := 0;
  i := 0;
assert(i =< n);      // wchodzimy do pętli
  sum := sum + i;
  i := i + 1;
assert(i > n).       // opuszczamy pętlę
```

Przykłady programów

Analiza pseudokodu

Konstrukcja najsłabszego warunku wstępnego dla danej ścieżki:

```
% constraint(+Path, -Condition)
%
constraint([], true).
constraint([X := E | P], C) :-
    constraint(P, C1),
    substitute(C1, X, E, C).
constraint([assert(C) | P], and(C, C1)) :-
    constraint(P, C1).
```

Konstrukcja najsłabszego warunku wstępnego dla ścieżki PATH, warunku początkowego PRE i warunku końcowego POST:

```
append([assert(PRE) | PATH], [assert(not(POST))], NEW_PATH),
constraint(NEW_PATH, CONDITION)
```

Jeśli istnieją dane spełniające tak skonstruowany warunek, to program nie jest poprawny (dane spełniają warunek początkowy ale po wykonaniu programu nie jest spełniony warunek końcowy).

Przykłady programów

Analiza pseudokodu

Poszukiwanie kontrprzykładu:

```
analyze(FileName, Initial_State) :-
    file_parser(FileName, PRE, BLOCK, POST),
    length(PATH, _),
    path(BLOCK, PATH), % generowanie coraz dłuższych ścieżek
    append([assert(PRE) | PATH], [assert(not(POST))], PATH1),
    constraint(PATH1, CONDITION),
    solve(CONDITION, Initial_State).
```

Przykłady programów

Analiza pseudokodu

Example (Niepoprawne sumowanie)

Program w pseudokodzie:

```
pre (n >= 0);
sum := 0; i := 0;
while i < n loop
    sum := sum + i; i := i + 1;
end loop;
post sum = (n*(n + 1)) div 2;
```

```
n = 1
sum = 0, i = 0
0 < 1
sum = 0, i = 1
not 1 < 1
not sum = 1
```

```
?- analyze('sumowanie.pc', X).
X = [n = 1]
```

Należy poprawić warunek sterujący pętlą na `i<=n`.

Przykłady programów

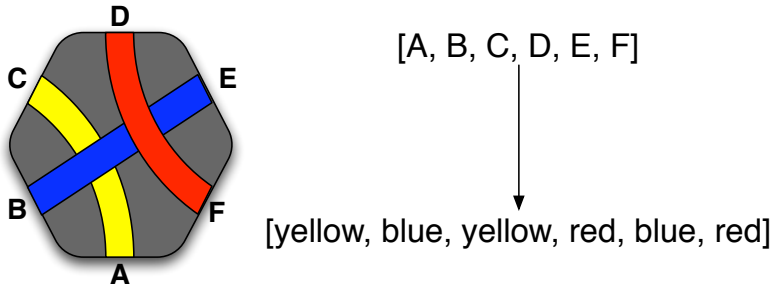
TANTRIX Discovery

- Łamigłówka TANTRIX Discovery (odkrywanka) polega na układaniu kolorowych cykli z sześciokątnych płytek.
- Każda płytka zawiera trzy różnokolorowe łuki łączące jej brzegi.
- Kolory łuków na stykających się brzegach muszą być zgodne.
- Utworzona pętla nie może zwierać pustych miejsc w swoim wnętrzu.

Przykłady programów

TANTRIX Discovery

- W zabawie korzysta się z pierwszych 30-tu płytek łamigłówki TANTRIX o numerach od 1 do 30.
- Każda płytka ma na odwrotnej stronie numer w jednym z trzech kolorów: żółtym, czerwonym lub niebieskim.
- Dla danego $n \in \{3, 4, \ldots, 30\}$ używamy n płytek o numerach od 1 do n i układamy z nich cykl koloru takiego jak kolor numeru n-tej płytki.
- Reprezentacja płytki:



Przykłady programów

TANTRIX Discovery

Pierwszych dziesięć płytek:

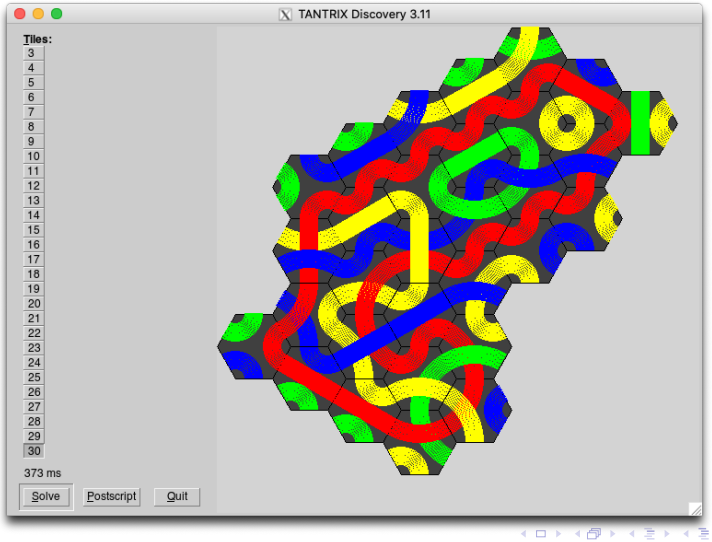
```
tile( 1, [blue,red,blue,red,yellow,yellow], yellow).
tile( 2, [blue,red,red,blue,yellow,yellow], yellow).
tile( 3, [blue,blue,yellow,yellow,red,red], yellow).
tile( 4, [blue,red,yellow,blue,yellow,red], red).
tile( 5, [red,blue,blue,red,yellow,yellow], red).
tile( 6, [yellow,blue,red,yellow,red,blue], blue).
tile( 7, [yellow,red,yellow,red,blue,blue], blue).
tile( 8, [red,yellow,red,yellow,blue,blue], blue).
tile( 9, [red,blue,yellow,red,yellow,blue], yellow).
tile(10, [red,blue,red,blue,yellow,yellow], red).
...
```

Dla $n = 30$, przestrzeń rozwiązań ma $30! \cdot 6^{30} \approx 5.86 \cdot 10^{56}$ elementów.

Przykłady programów

TANTRIX Discovery

Example (Program w SWI-Prologu)

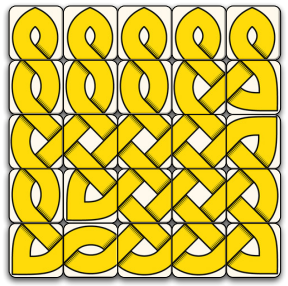


Przykłady programów

Węzły celtyckie

Z poniższych płytek będziemy układać węzły celtyckie:

Przykład poprawnego węzła:



Przykłady programów

Węzły celtyckie

- Celem jest wypełnienie płytkami (możliwe obroty) planszy 5×5, tak aby płytki pasowały do siebie na stykających się krawędziach.
- Możliwe są dwa przypadki:
  1. Węzeł nie przechodzi przez krawędź płytki (kod $= 0$).
  2. Węzeł dwukrotnie przechodzi przez krawędź płytki: raz dołem i raz górą (kod $= 1$).
- Z każdym z 25 pól kwadratu 5×5 związane są cztery zmienne (po jednej na każdą krawędź):

$$[N_{ij}, E_{ij}, S_{ij}, W_{ij}] \text{ ins } 0..1$$

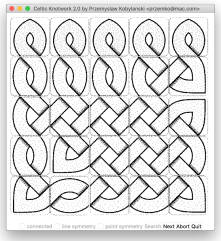
- Na wszystkie 100 zmiennych zero-jedynkowych narzucone są więzy:
  1. Równość zmiennych na stykających się krawędziach.
  2. Odpowiednie rodzaje płytek.
  3. Odpowiednie obroty płytek.

Przykłady programów

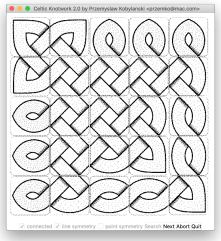
Węzły celtyckie

- Znalezione wartości wszystkich zmiennych reprezentują rozwiązanie (przeszukiwana przestrzeń ma $2^{100} \approx 1.27 \cdot 10^{30}$ elementów).

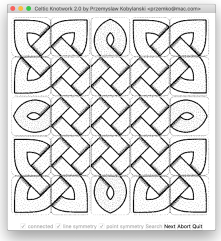
Example (Program w SICStus Prologu)



a) dowolny



b) osiowa



c) osiowa i środkowa

Przykłady programów

Podział prostokąta na kwadraty

Problem

Czy można podzielić prostokąt na n parami różnych kwadratów?

- W sformułowaniu brakuje informacji o rozmiarze prostokąta i rozmiarach kwadratów.
- Czy mimo to da się rozwiązać powyższy problem?
- Między innymi do rozwiązywania tak sformułowanych problemów stworzono Prolog.

Przykłady programów

Podział prostokąta na kwadraty

- Do narzucania ograniczeń na zmienne użyjemy modułu **clpq** (ograniczenia na zmiennych o wartościach wymiernych).
- Ograniczenia umieszcza się między nawiasami klamrowymi.
- Liczby wymierne wyrażone są jako ułamki, których liczniki i mianowniki są dowolnie dużymi liczbami całkowitymi (biblioteka **GMP**: *GNU Multiple Precision Arithmetic Library*).
- Istnieje też szybciej działający moduł **clpr** z ograniczeniami na zmiennych o wartościach zmiennopozycyjnych ale zależy nam na rozwiązaniu **dokładnym** (typ zmiennopozycyjny tego nie gwarantuje).

Przykłady programów

Podział prostokąta na kwadraty

Example

Liczby zmiennopozycyjne:

```
?- use_module(library(clpr)).
?- {3 * X =< 1}, maximize(X).
X = 0.3333333333333333 .
```

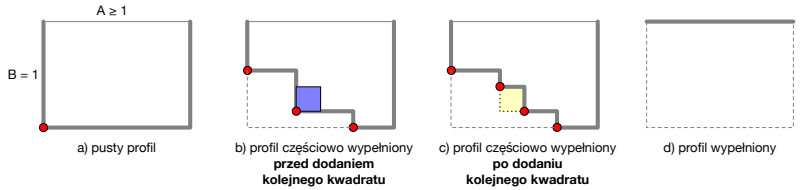
Liczby wymierne:

```
?- use_module(library(clpq)).
?- {3 * X =< 1}, maximize(X).
X = 1r3.
```

Przykłady programów

Podział prostokąta na kwadraty

Wypełnianie profilu kwadratami:



Uwaga: wszystkie rozmiary są zmiennymi z narzuconymi na nie ograniczeniami. Dodanie kolejnego kwadratu dołącza kolejne ograniczenia. W chwili wypełnienia całego profilu układ wszystkich ograniczeń zostanie rozwiązany i poznamy rozmiary prostokąta i kwadratów. Jeśli podczas dodawania kwadratu zostanie wykryta sprzeczność zbioru ograniczeń, to program wycofa się ze wstawiania w to miejsce i spróbuje w inne (układ może być sprzeczny chociaż nie są jeszcze ustalone wartości zmiennych).

Przykłady programów

Podział prostokąta na kwadraty

Example (Program w SWI-Prologu)

Wszystkie rozwiązania dla $n = 9$ znalezione modułem **clpq**:

```
?- length(X, 9), filled_rectangle(A, X).
X = [15r32, 9r16, 1r4, 7r32, 1r8, 7r16, 1r32, 5r16, 9r32],
A = 33r32 ;
X = [33r61, 36r61, 28r61, 5r61, 2r61, 9r61, 25r61, 7r61, 16r61],
A = 69r61 ;
X = [9r16, 15r32, 7r32, 1r4, 7r16, 1r8, 5r16, 1r32, 9r32],
A = 33r32 ;
X = [36r61, 33r61, 5r61, 28r61, 25r61, 9r61, 2r61, 7r61, 16r61],
A = 69r61 ;
X = [9r32, 5r16, 7r16, 1r4, 1r32, 7r32, 1r8, 9r16, 15r32],
A = 33r32 ;
X = [28r61, 16r61, 25r61, 7r61, 9r61, 5r61, 2r61, 36r61, 33r61],
A = 69r61 ;
X = [25r61, 16r61, 28r61, 9r61, 7r61, 2r61, 5r61, 36r61, 33r61],
A = 69r61 ;
X = [7r16, 5r16, 9r32, 1r32, 1r4, 1r8, 7r32, 9r16, 15r32],
A = 33r32 ;
false.
```

Przykłady programów

Podział prostokąta na kwadraty

Example (Program w SWI-Prologu cd.)

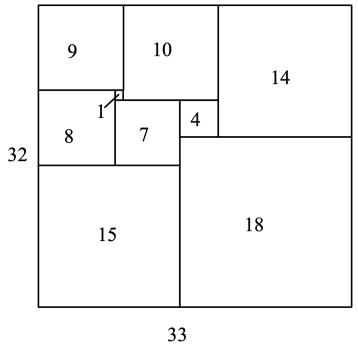
Prawdę mówiąc, nawet n nie musi być zadane:

```
?- length(X, N), filled_rectangle(A, X).
X = [1],
N = A, A = 1 ;
X = [15r32, 9r16, 1r4, 7r32, 1r8, 7r16, 1r32, 5r16, 9r32],
N = 9,
A = 33r32 .
```

Jak widać, najmniejsze **nietrywialne** rozwiązanie wymaga 9-ciu kwadratów.

Example (Program w SWI-Prologu cd.)

Najmniejsze nietrywialne rozwiązanie po 32-krotnym powiększeniu:



źródło: Colmerauer A.: An Introduction to Prolog III, Communications of the ACM, 33(7), 69-90, 1990.