Programowanie w Logice
Korutyny i wątki
(Lista 7)

Przemysław Kobylański

Wstęp

Korutyna czekająca na wartość zmiennej `X` musi być zapisana w postaci jednej klauzuli:

```
korutyna(X, ...) :-
        freeze(X, ODROCZONY_CEL).
```

Zadania

Zadanie 1 (5 pkt)

Załóżmy, że w strumieniu liczb reprezentowanym listą otwartą liczby pojawiają się w kolejności niemalejącej.

Wykorzystując korutyny napisz predykat `merge(IN1, IN2, OUT)` scalający dwa niemalejące strumienie liczb `IN1` i `IN2` w jeden niemalejący strumień liczb `OUT2`.

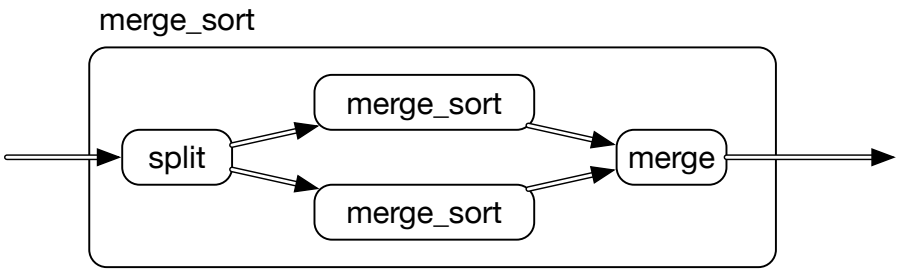
Przykłady

```
?- merge([1, 3], [2, 4], X).
X = [1, 2, 3, 4].

?- merge([1, 3 | A], [2, 4 | B], X).
X = [1, 2, 3|_G3582],
freeze(...).
```

Zadanie 2 (3 pkt)

Wykorzystując korutyny napisz predykat `split(IN, OUT1, OUT2)`, który rozrzuca elementy strumienia wejściowego `IN` na dwa strumienie `OUT1` i `OUT2`, przy czym:



Rysunek 1: Schemat merge_sort

- Każdy element ze strumienia `IN` pojawia się albo w strumieniu `OUT1` albo w strumieniu `OUT2`.
- Kiedy strumień wejściowy `IN` zakończy się, to w obu strumieniach `OUT1` i `OUT2` powinno znaleźć się mniej więcej tyle samo elementów (liczba elementów może różnić się o co najwyżej 1).

Do predykatów `split/3` i `merge/3` (z poprzedniego zadania) dopisz predykat `merge_sort(IN, OUT)`, który stosując korutyny sortuje strumień liczb `IN` dzieląc go na dwa podstrumienie, następnie sortując je rekurencyjnie i scalając dwa uporządkowane strumienie wyjściowe.

Na rysunku 1 przedstawiono schemat `merge_sort`. Podwójnymi strzałkami zaznaczono strumienie danych.

Przykłady

```
?- split([a, b, c, d], OUT1, OUT2).
OUT1 = [a, c],
OUT2 = [b, d].

?- split([a, b, c, d | A], OUT1, OUT2).
OUT1 = [a, c|_G3578],
OUT2 = [b, d|_G3590],
freeze(...).

?- merge_sort([5, 1, 3, 2, 4], X).
X = [1, 2, 3, 4, 5].

?- merge_sort(X, Y), X = [4, 1, 3, 2].
X = [4, 1, 3, 2],
Y = [1, 2, 3, 4].

?- merge_sort([5, 1, 3, 2, 4 | A], X).
```

Rysunek 2: Ucztujący filozofowie (źródło wikipedia).

```
freeze(...),
freeze(...),
...
freeze(...).
```

Zadanie 3 (2 pkt)

W tym zadaniu rozpatrujemy problem pięciu ucztujących filozofów. Na rysunku 2 przedstawiono pięciu filozofów siedzących przy okrągłym stole. Między nimi leży pięć widelców, przy czym każdy z nich jest współdzielony przez dwóch sąsiadujących ze sobą filozofów.

Napisz predykat `filozofowie/0`, który uruchamia pięć wątków reprezentujących filozofów (po jednym wątku dla każdego filozofa).

Każdy filozof działa w następującej nieskończonej pętli:

```
loop
  filozof myśli;
  filozof stara się podnieść prawy widelec;
  filozof stara się podnieść lewy widelec;
  filozof je;
  filozof odkłada prawy widelec;
  filozof odkłada lewy widelec;
```

```
end loop;
```

Predykat wykonywany w wątku odpowiadającym danemu filozofowi powinien drukować komunikaty o tym, że filozof zaczyna myśleć, podnosi widelce, je i odkłada widelce.

Czy możliwe jest zakleszczenie się filozofów (każdy z nich czeka na jeden z widelców i nie może kontynuować swojego działania)?

Wskazówka

Pomyśl o użyciu mutexów.

Przykład

Początkowy fragment śladu wykonania predykatu `filozofowie/0` (liczba w kwadratowych nawiasach oznacza numer filozofa):

```
?- filozofowie.
 [1]  mysli
 [1]  chce prawy widelec
 [1]  podniosl prawy widelec
[0]  mysli
 [1]  chce lewy widelec
  [2]  mysli
[0]  chce prawy widelec
 [1]  podniosl lewy widelec
   [3]  mysli
    [4]  mysli
  [2]  chce prawy widelec
[0]  podniosl prawy widelec
 [1]  je
   [3]  chce prawy widelec
    [4]  chce prawy widelec
[0]  chce lewy widelec
 [1]  odklada prawy widelec
   [3]  podniosl prawy widelec
    [4]  podniosl prawy widelec
 [1]  odklada lewy widelec
...
```