

Lista zagadnień nr 14

Przed zajęciami

Tematem bieżącego tygodnia jest programowanie logiczne przy użyciu Rackloga. Przed zajęciami warto zapoznać się z dokumentacją (<http://docs.racket-lang.org/racklog/>). Należy rozumieć pojęcia **unifikacji**, **przeszukiwania z nawrotami**, **predykatu**, **klauzuli** i **celu**.

W trakcie zajęć

Ćwiczenie 1.

Rozbuduj kod z wykładu z relacjami rodzinnymi o następujące dodatkowe predykaty: `%grandson`, `%cousin` (binarne), `%is_mother`, `%is_father` (unarne).

Ćwiczenie 2.

Zapisz w formie zapytania `%find-all` do Rackloga następujące pytania:

- Czy John jest potomkiem Marka?
- Kto jest potomkiem Adama?
- Kto jest siostrą Ivonne?
- Kto ma w tej rodzinie kuzyna i kim ten kuzyn jest?

Ćwiczenie 3.

Zapisz w formie zapytania Rackloga następujące pytania:

- Jakie pary list mają tę własność, że druga lista jest wynikiem konkatencji dwóch kopii pierwszej z nich?
- Jaki element należy usunąć z listy `'(1 2 3 4)`, aby otrzymać `'(1 2 4)`?
- Jaką listę należy dołączyć do listy `'(1 2 3)`, aby otrzymać `'(1 2 3 4 5)`?

Ćwiczenie 4.

Narysuj drzewo przeszukiwań Rackloga dla zapytania:

```
(%which (x xs) (%select x (list 1 2) xs))
```

Ćwiczenie 5.

Zaprogramuj predykat %sublist, taki, że cel (%sublist xs ys) jest spełniony, gdy xs jest podlistą listy ys. Mówimy, że xs jest podlistą ys, jeśli powstała przez usunięcie niektórych elementów z ys (z zachowaniem kolejności). Precyzyjniej: mając daną listę $(x_1 x_2 \dots x_n)$, jej podlisty są postaci $(x_{i_1} x_{i_2} \dots x_{i_k})$, gdzie $0 \leq k \leq n$ i $1 \leq i_1 \leq i_2 \leq \dots \leq i_n \leq n$.

Ćwiczenie 6.

Zaprogramuj predykat %perm, taki, że cel (%perm xs ys) jest spełniony, gdy xs jest permutacją listy ys, tzn. zawiera te same elementy, w tej samej krotności, ale być może w innej kolejności. Precyzyjniej: mając daną listę $(x_1 x_2 \dots x_n)$, jej permutacje są postaci $(x_{i_1} x_{i_2} \dots x_{i_n})$, gdzie $i_1 \dots i_n$ są parami różnymi liczbami naturalnymi z zakresu $1 \dots n$. Można wykorzystać predykat %select z kodu przykładowego na SKOS. Uwaga: zadanie można wykonać na dwa istotnie różne sposoby.

Ćwiczenie 7.

Zaimplementuj w Rackecie funkcję list->num, która oblicza liczbę reprezentowaną przez listę cyfr. Przykładowo lista '(1 3 3 7) reprezentuje liczbę 1337. Następnie napisz zapytanie Rackloga, które rozwiązuje następującą łamigłówkę:

$$\begin{array}{rcccccc}
 & & S & E & N & D \\
 + & & M & O & R & E \\
 \hline
 = & M & O & N & E & Y
 \end{array}$$

Zapytanie powinno zawierać zmienne D, E, M, N, O, R, S i Y oraz powinno sprawdzać wszystkie możliwe podstawienia pod te zmienne parami różnych cyfr 0, 1, ..., 9. Cyfry podstawione pod S i M nie mogą być zerem. W zapytaniu możesz użyć: predykatu %length lub %gen-length z wykładu, predykatów %sublist i %perm z poprzednich zadań, wbudowanego predykatu %=\=, formy %is, arytmetyki Racketa, napisanej wcześniej funkcji list->num. Wykonanie zapytania może potrwać kilka minut!

Zadanie domowe

Zadanie 14

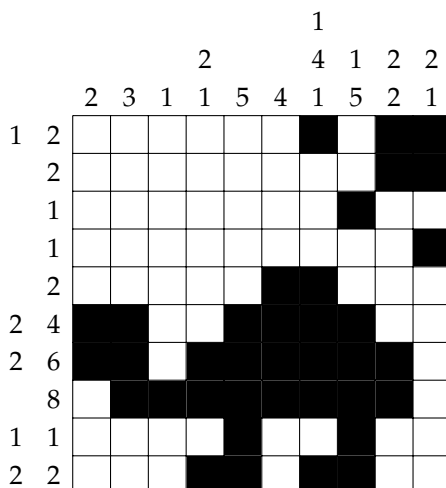
Obrazki logiczne (znane też jako: malowanie liczbami, nonogramy, picross) to rodzaj łamigłówki, której celem jest odkrycie zakodowanego rysunku przez zamalowywanie krutek prostokątnej siatki zgodnie z określonymi regułami. Na lewo od każdego wiersza oraz powyżej każdej kolumny siatki znajduje się ciąg cyfr oznaczający długości ciągłych bloków zamalowanych krutek znajdujących się w zadanym wierszu lub kolumnie. Bloki muszą wystąpić w tej samej kolejności, w jakiej są wymienione, muszą też być oddzielone co najmniej jedną niezamalowaną kratką.

Zaimplementuj przy użyciu Rackloga binarny predykat `%row-ok`. Jego pierwszym argumentem będą opisy wiersza lub kolumny obrazku logicznego (np. `'(2 4) lib '(1 4 1)`). Drugim argumentem natomiast będzie lista stanów kratki danego wiersza lub kolumny; zamalowaną kratkę oznacza symbol `'*`, zaś niezamalowaną – symbol `'_`. Przykładowa lista: `'(* * _ _ * * * _ _)`. Cel `%(row-ok xs ys)` ma być spełniony, gdy `xs` opisuje `ys` zgodnie z regułami obrazków logicznych. Przykładowo, poniższy cel zachodzi:

```
(%row-ok '(2 4) '(* * _ _ * * * * _ _))
```

Wykorzystując %row-ok, uzupełnij w szablonie pobranym ze SKOS funkcję solve, aby otrzymać program rozwiązujący obrazki logiczne.

Oto przykładowy obrazek logiczny, razem z rozwiązaniem:



Przykład pełni funkcję ilustracyjną. Może się zdarzyć, że napisane rozwiązanie nie będzie w stanie znaleźć rozwiązania dla powyższego przykładu odpowiednio szybko. Niewielkie przykłady testowe znajdują się w pliku szablonu rozwiązania.