Логическое программирование

Кевролетин В.В. группа c8503a(256)

20 November 2012

Содержание

1 Задание

1.1 Условие

Имеется набор букв. Предложение состоит из последовательности букв. Есть словарь D - набор слов. Слово - последовательность букв. Дана строка букв. Разделить последовательность на слова. Найти то решение, которое содержит минимальное количество строк.

1.2 Решение

Воспользуемся поиском в глубину для поиска решений. Затем из всех решений выберем лучшее.

1.2.1 Исходный код

- Стандартный фреймворк для поиска в глубину

```
solve_dfs(State, _, []) :-
        final_state(State).
solve_dfs(State, History, [Move|Moves]) :-
        move(State,Move),
        update(State, Move, State1),
        legal(State1),
        \+ member(State1,History),
        solve_dfs(State1,[State1|History],Moves).

test_dfs(Problem, Moves) :-
        initial_state(Problem, State),
        solve_dfs(State, [State], Moves).
```

- Структура терма, описывающего состояние

  Текущее состояние представляется двумя последовательностями символов - префиксом и суффиксом.

```
initial_state(alph, state([], Pattern)) :- pattern(Pattern).
final_state(state([], [])).

dict([['c'], ['a', 't'],
      ['c', 'a', 't'],
      ['i', 's'],
      ['b', 'l', 'a', 'c', 'k']]).
pattern(['c', 'a', 't', 'i', 's', 'b', 'l', 'a', 'c', 'k']).
```

- Переходы между состояниями

  Далее необходимо определить переходы между состояниями и проверки допустимости текущего состояния:

  Доступны 2 перехода между состояниями:

  - inc_prefix - увеличить префикс на 1 символ за счет уменьшения суффкса
  - match_prefix - добавить накопленное в префиксе слово к результату

```
move(state(_, [_|_]), inc_prefix).
move(state([_|_], _), match_prefix).

update(state(Prefix, [X|Xs]), inc_prefix, state(NewPrefix, Xs)) :-
        append(Prefix, [X], NewPrefix).
update(state(Prefix, Suffix), match_prefix, state([], Suffix)) :-
        dict(D),
        member(Prefix, D).
```

  Для простоты реализации предикат update сочетает в себе функцию, обычно возложенную на legal.

```
legal(_).
```

- Тестирование поиска в глубину

  Приведенного кода достаточно, чтобы получить решения:

```
| ?- test_dfs(alph, X). a

X = [inc_prefix,inc_prefix,inc_prefix,match_prefix,
     inc_prefix,inc_prefix,match_prefix,inc_prefix,
     inc_prefix,inc_prefix,inc_prefix,inc_prefix,match_prefix] ?

X = [inc_prefix,match_prefix,inc_prefix,inc_prefix,
     match_prefix,inc_prefix,inc_prefix,match_prefix,
     inc_prefix,inc_prefix,inc_prefix,inc_prefix,inc_prefix,match_prefix]

no
| ?-
```

Решение состоит из последовательности ходов, для наглядности переведем последовательность ходов в предложение из слов:

```
show_solution(Moves, Res) :-
        initial_state(alph, State),
        show_solution(Moves, State, [], Res).

show_solution([], _, Res, Res).
show_solution([inc_prefix|Xs], State, CurrRes, Res) :-
        update(State, inc_prefix, NewState),
        show_solution(Xs, NewState, CurrRes, Res).
show_solution([match_prefix|Xs], state(Prefix, Suffix), CurrRes, Res) :-
        update(state(Prefix, Suffix), match_prefix, NewState),
        append(CurrRes, [Prefix], NewCurrRes),
        show_solution(Xs, NewState, NewCurrRes, Res).

| ?- test_dfs(alph, X), show_solution(X, Res). a

Res = [[c,a,t],[i,s],[b,l,a,c,k]]
X = [inc_prefix,inc_prefix,inc_prefix|...]

Res = [[c],[a,t],[i,s],[b,l,a,c,k]]
X = [inc_prefix,match_prefix,inc_prefix|...]

no
| ?-
```

- Выбор наилучшего решения

Теперь, используя системный findall выберем лучшее решение:

```
find_best_solution(Res) :-
        findall(X, test_dfs(alph, X), [Fst|Others]),
        choose_best(Others, Fst, Res).

words_cnt(Solution, Res) :-
        show_solution(Solution, Words),
        length(Words, Res).

choose_best([], Res, Res).
choose_best([X|Xs], CurrentBest, Res) :-
        words_cnt(X, NewLen),
        words_cnt(CurrentBest, BestLen),
        NewLen < BestLen,
        choose_best(Xs, X, Res).
choose_best([X|Xs], CurrentBest, Res) :-
        words_cnt(X, NewLen),
        words_cnt(CurrentBest, BestLen),
        NewLen >= BestLen,
        choose_best(Xs, CurrentBest, Res).
```

1.2.2 Тесты

```
| ?- find_best_solution(X), show_solution(X, Res). a

Res = [[c,a,t],[i,s],[b,l,a,c,k]]
X = [inc_prefix,inc_prefix,inc_prefix,match_prefix,inc_prefix,
     inc_prefix,match_prefix,inc_prefix,inc_prefix,inc_prefix,
     inc_prefix,inc_prefix,match_prefix] ?

no
| ?-
```

1.2.3 Тесты для бесуспешного вычисления

Добавим в конец последовательности символов слово, которого нет в словаре:

```
pattern(['c', 'a', 't', 'i', 's', 'b', 'l', 'a', 'c', 'k',
         'r', 'e', 'a', 'l', 'l', 'y']).

?- test_dfs(alph, X).
false.

?- find_best_solution(X).
false.

?-
```

Добавим в середину последовательности символов слово, которого нет в словаре:

```
pattern(['c', 'a', 't', 'i', 's', 'v', 'e', 'r', 'y',
         'b', 'l', 'a', 'c', 'k']).

?- test_dfs(alph, X).
false.

?- find_best_solution(X).
false.

?-
```

В 2х случаях решение не найдено.