

Математическая логика и логическое программирование

mk.cs.msu.ru → Лекционные курсы
→ Математическая логика и логическое программирование (3-й поток)

Блок 32

Хорновские логические программы:
списки

Лектор:
Подымов Владислав Васильевич
E-mail:
valdus@yandex.ru

Вступление

Список — это популярная структура данных, представляющая конечные последовательности элементов тех или иных множеств

В некоторых языках используется широкое понимание списка, согласно которому элементом может быть всё, что можно записать

В частности, элементами списков могут быть и другие списки

Чтобы избежать аналога **парадокса Рассела** для списков, спискам принято придавать особую характеристику — **глубину** (вложенности одних списков в другие), и разрешать в качестве элементов списка использовать только списки меньше глубины

Здесь обойдёмся без понятия глубины, не ограничивая свободу построения списков, за счёт неявного использования глубины в БНФ

Определение списка

Список L сигнатуры σ логики предикатов задаётся следующей БНФ:

$$L ::= \mathbf{nil} \mid t.L,$$

где $t \in \text{Term}$

nil — это **пустой список**: особая константа, представляющая пустую последовательность

Головой непустой последовательности S будем называть первый элемент, а **хвостом** — последовательность, получающуюся из S вычёркиванием головы

. — это **конструктор списков**: особый двуместный функциональный символ, позволяющий описывать списки большего размера на основе списков меньшего размера **в инфиксной форме**

Символ **.** считается **ассоциативным вправо**: $a.b.c = a.(b.c)$

Списком $t.L$ представлена последовательность с головой t и хвостом L

Терминологию, применяющуюся к последовательностям, будем применять и к спискам, представляющим эти последовательности

Например, в записи « $t.L$ » t — это **голова** и L — это **хвост** списка

Примеры списков

Последовательность элементов a_1, \dots, a_k будем в примерах обрамлять в скобки: (a_1, \dots, a_k)

Пустую последовательность будем изображать как «пустую» пару скобок: $()$

Примеры списков

Список

nil

a.nil

a.X.b.nil

a.X.b

nil.nil

nil.nil.nil

(nil.nil).nil

(a.b.nil).(c.d.nil).nil

((a.b.nil).nil.nil).(nil.nil).nil

Соответствующая последовательность

$()$

(a)

(a, X, b)

Это корректный терм, но не список

$(())$

$((), ())$

$((()))$

$((a, b), (c, d))$

$((((a, b), ()), ()))$

Примеры списков

Если захотите транслировать списки из предложенных обозначений в язык Prolog, то достаточно сделать следующее:

1. Вместо «**nil**» писать «**[]**»
2. Вместо « $a_1 \dots a_k.L$ » писать « $[a_1, \dots, a_k \mid L]$ », соответственно транслировав a_1, \dots, a_k и L
3. По желанию вместо « $[a_1, \dots, a_k \mid []]$ » писать просто « $[a_1, \dots, a_k]$ »

Например:

Список в ХЛП

nil

a.X.b.nil

nil.nil.nil

(nil.nil).nil

(a.b.nil).(c.d.nil).nil

((a.b.nil).nil.nil).(nil.nil).nil

((a.b.nil).nil.nil).(nil.nil).L

Список в Prolog

[]

[a, X, b] или **[a, X, b | []]**

[[], []]

[[]]

[[a, b], [c, d]]

[[[a, b], []], [[]]]

[[[a, b], [], []] | L]

Программы со списками

Пример

1. $\text{elem}(X, X.L)$;
2. $\text{elem}(X, Y.L) \leftarrow \text{elem}(X, L)$;

В этой программе записано *индуктивное определение* понятия « x является элементом списка ℓ »:

1. Голова списка является его элементом
2. Если X является элементом хвоста списка, то X является элементом списка

Несложно видеть, что этими двумя пунктами «покрываются» все случаи принадлежности элемента списку

Правильными ответами на запрос

`?elem(X, тропинка.лесок.колосок.колосок.колосок.речка.небо.nil)`

являются все подстановки элементов списка на место X :

`{X/тропинка} {X/лесок} {X/колосок} {X/речка} {X/небо}`

Программы со списками

Декларативная семантика подпрограмм естественным образом используется в декларативной семантике «над» программ

Например:

```
elem(X, X.L);  
elem(X, Y.L) ← elem(X, L);  
common(X, L1, L2) ← elem(X, L1), elem(X, L2);
```

Согласно первым двум правилам, запись $\text{elem}(X, L)$ означает, что X является элементом списка L

Значит $\text{common}(X, L_1, L_2)$ означает, что

X — элемент списка L_1 и X — элемент списка L_2

То есть

X — общий элемент списков L_1 и L_2

В частности, вот все правильные ответы на запрос

$? \text{common}(X, \text{п.о.п.nil}, \text{к.л.о.п.nil})$

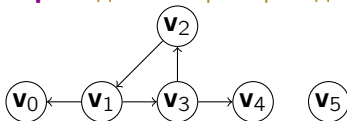
к этой программе:

$\{X/\text{п}\}$

$\{X/\text{о}\}$

Программы со списками

И ещё пример: задача о проверке достижимости в графе из блока 30



$v_1 \rightsquigarrow v_4 ?$

Путь $w_1 \rightarrow \dots \rightarrow w_k$ можно представить как **последовательность** (w_1, \dots, w_k) , и её — как **список** $w_1 \cdot \dots \cdot w_k \cdot \mathbf{nil}$

Систему дизъюнктов из блока 30 можно переписать как такие логическую программу и запрос:

$$\begin{aligned} & \text{arc}(v_1, v_0); \text{ arc}(v_1, v_3); \text{ arc}(v_3, v_2); \\ & \text{arc}(v_2, v_1); \text{ arc}(v_3, v_4); \text{ reach}(X, X, X.\mathbf{nil}); \\ & \text{reach}(X, Z, X.P) \leftarrow \text{arc}(X, Y), \text{ reach}(Y, Z, P); \quad ?\text{reach}(v_1, v_4, P) \end{aligned}$$

Существует бесконечно много правильных ответов на этот запрос к этой программе, отвечающих всевозможным путям из v_1 в v_4 , — например:

$$\begin{aligned} & \{P/v_1.v_3.v_4.\mathbf{nil}\} \\ & \{P/v_1.v_3.v_2.v_1.v_3.v_4.\mathbf{nil}\} \\ & \{P/v_1.v_3.v_2.v_1.v_3.v_2.v_1.v_3.v_4.\mathbf{nil}\} \end{aligned}$$