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

ВМК МГУ, 2024/2025, осенний семестр

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 меньшей глубины

Здесь обойдёмся без понятия глубины за счёт неявного использования глубины в БНФ

Определение списка

Список L *сигнатуры* σ *логики предикатов* задаётся следующей БНФ:

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

где $t \in \mathrm{Term}$

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

Головой непустой последовательности $\mathcal{S}$ будем называть первый элемент, а хвостом — последовательность, получающуюся из $\mathcal{S}$ удалением 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 $a_1, \ldots, a_k$ будем в примерах обрамлять в скобки: $(a_1, \ldots, a_k)$

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

Примеры списков

Список	Соответствующая последовательность
nil	()
a.nil	(**a**)
a.X**.b.nil**	(**a**, X, **b**)
a.X**.b**	Это корректный терм, но не список
nil.nil	(())
nil.nil.nil	(), ())
(**nil.nil**)**.nil**	((()))
(**a.b.nil**)**.**(**c.d.nil**)**.nil**	((**a**, **b**), (**c**, **d**))
((**a.b.nil**)**.nil.nil**)**.**(**nil.nil**)**.nil**	(((**a**, **b**), ()), (()))

Примеры списков

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

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

Например:

Список в ХЛП	Список в Prolog
nil	[]
a.X**.b.nil**	[**a**, X, **b**] или [**a**, X, **b** | []]
nil.nil.nil	[[], []]
(**nil.nil**)**.nil**	[[[]]]
(**a.b.nil**)**.**(**c.d.nil**)**.nil**	[[**a**, **b**], [**c**, **d**]]
((**a.b.nil**)**.nil.nil**)**.**(**nil.nil**)**.nil**	[[[**a**, **b**], []], [[]]]
((**a.b.nil**)**.nil.nil**)**.**(**nil.nil**)**.**ℓ	[[[**a**, **b**], []], [[]] | ℓ]

Программы со списками

Пример

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

В этой программе записано *индуктивное определение* отношения $\mathrm{elem}(x, \ell)$ «x является элементом списка ℓ»:

1. Голова списка является его элементом
2. Если $\mathtt{X}$ является элементом хвоста списка, то $\mathtt{X}$ является элементом списка

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

Правильными ответами на запрос
$?\mathrm{elem}(\mathtt{X}, \textbf{тропинка.лесок.колосок.колосок.речка.небо_голубое.nil})$
являются все подстановки элементов списка на место $\mathtt{X}$:
$\{\mathtt{X}/\textbf{тропинка}\}$ $\{\mathtt{X}/\textbf{лесок}\}$ $\{\mathtt{X}/\textbf{колосок}\}$ $\{\mathtt{X}/\textbf{речка}\}$ $\{\mathtt{X}/\textbf{небо_голубое}\}$

Программы со списками

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

Например:

$$\mathrm{elem}(\mathtt{X}, \mathtt{X}.\mathtt{L});$$
$$\mathrm{elem}(\mathtt{X}, \mathtt{Y}.\mathtt{L}) \leftarrow \mathrm{elem}(\mathtt{X}, \mathtt{L});$$
$$\mathrm{common}(\mathtt{X}, \mathtt{L}_1, \mathtt{L}_2) \leftarrow \mathrm{elem}(\mathtt{X}, \mathtt{L}_1), \mathrm{elem}(\mathtt{X}, \mathtt{L}_2);$$

Согласно первым двум правилам, запись $\mathrm{elem}(x, \ell)$ означает, что x является элементом списка ℓ

Значит $\mathrm{common}(\mathtt{X}, \mathtt{L}_1, \mathtt{L}_2)$ означает, что

$\mathtt{X}$ — элемент списка $\mathtt{L}_1$ и $\mathtt{X}$ — элемент списка $\mathtt{L}_2$

То есть

$\mathtt{X}$ — общий элемент списков $\mathtt{L}_1$ и $\mathtt{L}_2$

В частности, вот все правильные ответы на запрос

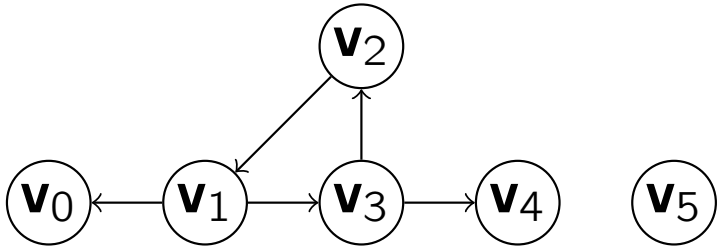
$$?\mathrm{common}(\mathtt{X}, \textbf{п.о.п.nil}, \textbf{к.л.о.п.nil})$$

к этой программе:

$$\{\mathtt{X}/\textbf{п}\} \qquad\qquad \{\mathtt{X}/\textbf{о}\}$$

Программы со списками

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



$\mathbf{v}_1 \rightsquigarrow \mathbf{v}_4$?

Путь $\mathbf{w}_1 \hookrightarrow \ldots \hookrightarrow \mathbf{w}_k$ можно представить как последовательность $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$, и её — как список $\mathbf{w}_1\mathbf{.}\cdots\mathbf{.w}_k\mathbf{.nil}$

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

$$\mathrm{arc}(\mathbf{v}_1, \mathbf{v}_0); \quad \mathrm{arc}(\mathbf{v}_1, \mathbf{v}_3); \quad \mathrm{arc}(\mathbf{v}_3, \mathbf{v}_2);$$
$$\mathrm{arc}(\mathbf{v}_2, \mathbf{v}_1); \quad \mathrm{arc}(\mathbf{v}_3, \mathbf{v}_4); \quad \mathrm{reach}(\mathrm{X}, \mathrm{X}, \mathrm{X}\mathbf{.nil});$$
$$\mathrm{reach}(\mathrm{X}, \mathrm{Z}, \mathrm{X}\mathbf{.}\mathrm{P}) \leftarrow \mathrm{arc}(\mathrm{X}, \mathrm{Y}), \mathrm{reach}(\mathrm{Y}, \mathrm{Z}, \mathrm{P}); \qquad ?\mathrm{reach}(\mathbf{v}_1, \mathbf{v}_4, \mathrm{P})$$

Существует бесконечно много правильных ответов на этот запрос к этой программе, отвечающих всевозможным путям из $\mathbf{v}_1$ в $\mathbf{v}_4$, — например:

$$\{\mathrm{P}/\mathbf{v}_1\mathbf{.}\mathbf{v}_3\mathbf{.}\mathbf{v}_4\mathbf{.nil}\}$$
$$\{\mathrm{P}/\mathbf{v}_1\mathbf{.}\mathbf{v}_3\mathbf{.}\mathbf{v}_2\mathbf{.}\mathbf{v}_1\mathbf{.}\mathbf{v}_3\mathbf{.}\mathbf{v}_4\mathbf{.nil}\}$$
$$\{\mathrm{P}/\mathbf{v}_1\mathbf{.}\mathbf{v}_3\mathbf{.}\mathbf{v}_2\mathbf{.}\mathbf{v}_1\mathbf{.}\mathbf{v}_3\mathbf{.}\mathbf{v}_2\mathbf{.}\mathbf{v}_1\mathbf{.}\mathbf{v}_3\mathbf{.}\mathbf{v}_4\mathbf{.nil}\}$$