Математическая логика

mk.cs.msu.ru → Лекционные курсы → Математическая логика (318, 319/2, 241, 242)

Вопрос 4

Логическое программирование
Декларативная семантика и операционная семантика, соотношение между ними
Стандартная стратегия выполнения логических программ.

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

ВМК МГУ, 2022/2023, весенний семестр

Логическая парадигма программирования

В логической парадигме программирования:

- Программа — это набор логических формул, представляющих собой совокупность знаний о понятиях задачи / описание необходимых и достаточных свойств результата / определение результата
- Правильный результат — это следствие программы как набора формул
- В основной (декларативной) семантике не говорится, **как** вычислять (извлекать) логические следствия, говорится только **что** является основанием для извлечения следствий

Несколько известных языков программирования с преобладающей логической парадигмой: **Prolog**, Datalog, Planner, Mercury, Gödel, ...

Основная математическая вычислительная модель этой парадигмы: логический вывод (логические исчисления)

ХЛП: синтаксис

Хорновская логическая программа (ХЛП) *сигнатуры σ логики предикатов* — это конечная последовательность программных утверждений, каждое из которых представляет собой факт или правило

Факт имеет вид «$A;$», где A — атом логики предикатов

Правило имеет вид «$A \leftarrow B_1, \ldots, B_k;$», где:

- $k \geq 1$
- A — заголовок: атом логики предикатов
- $B_1, \ldots, B_k$ — тело: последовательность атомов логики предикатов, разделённых запятой

Запрос к ХЛП (или, по-другому, целевое утверждение, или просто цель) имеет вид «$?C_1, \ldots, C_k$», где

- $k \geq 0$, и для случая $k = 0$ запрос принято записывать так: $\square$
- $C_1, \ldots, C_k$ — тело

ХЛП: синтаксис

Иными словами, ХЛП и запрос к ней задаются следующей БНФ:

```
ХЛП          ::=  | утверждение ХЛП
утверждение  ::=  факт | правило
факт         ::=  атом;
правило      ::=  заголовок ← тело;
заголовок    ::=  атом
тело         ::=  атом | атом, тело
запрос       ::=  □ | ? тело
```

Для технического единообразия будем считать, что факт — это правило с пустым телом:

$$«A;» = «A \leftarrow;»$$

Каждый атом в теле запроса (цели) принято также называть подцелью

Переменные, содержащиеся в запросе (цели), принято называть целевыми

ХЛП: синтаксис

Если захотите транслировать ХЛП и запрос в язык Prolog, то для этого достаточно сделать следующее:

1. Заменить все «`;`» на «`.`», «`←`» на «`:-`», «`?`» на «`?-`» и добавить «`.`» в конце запроса
   - В курсе оставим синтаксис ХЛП как есть, чтобы не изменять сложившуюся математику ради одного конкретного языка программирования

2. Начинать все переменные с прописной (большой) буквы, а остальные идентификаторы — со строчной (маленькой)
   - Будем стараться придерживаться такого написания в курсе, дополнительно различая категории символов шрифтами: $\mathtt{X}$ — переменная, **a** — константа или функциональный символ, p — предикатный символ

ХЛП: декларативная семантика

Почти каждому элементу ξ синтаксиса ХЛП можно естественным образом сопоставить формулу логики предикатов Φ_ξ:

Элемент	общий вид ξ	формула Φ_ξ
Факт	$A;$	$\forall \ldots\, A$
Тело	$B_1, \ldots, B_k$	$B_1 \,\&\, \ldots \,\&\, B_k$
Правило	$A \leftarrow \beta$	$\forall \ldots\, (\Phi_\beta \to A)$
Запрос	$?\gamma$	Φ_γ

Здесь $\forall \ldots\, \varphi(\widetilde{\mathrm{x}^{\mathrm{n}}})$ означает $\forall \widetilde{\mathrm{x}^{\mathrm{n}}}\, \varphi(\widetilde{\mathrm{x}^{\mathrm{n}}})$

Хорновской логической программе $\mathcal{P} = (\mathcal{R}_1 \ldots \mathcal{R}_m)$ сопоста́вим систему формул $S_\mathcal{P} = \{\Phi_{\mathcal{R}_1}, \ldots, \Phi_{\mathcal{R}_m}\}$

ХЛП: декларативная семантика

Содержательно, декларативная (основная) семантика ХЛП $\mathcal{P}$ и запроса $\mathcal{Q}$ к ней устроена так:

- Программа — это база имеющихся заведомо верных знаний
  - Правило «$A \leftarrow [B_1, \ldots, B_k];$» — это утверждение о том, что для любых значений переменных утверждение A верно [если для тех же значений верны все утверждения $B_1, \ldots, B_k$]
- Запрос к программе — это входные данные, определяющие вопрос об имеющихся знаниях, на который требуется ответить, записав ответ в целевые переменные (*как запрос к базе данных*)
  - Запрос «$?C_1, \ldots, C_m$» отвечает вопросу «для каких значений целевых переменных становятся одновременно верными все утверждения $C_1, \ldots, C_m$?»
- Правильный ответ на запрос к программе — это значения целевых переменных $\mathcal{Q}$, при которых этот запрос как формула следует из программы как формулы

ХЛП: декларативная семантика

Формально, центральное понятие декларативной семантики — это правильный ответ, и это понятие определяется так

$\mathrm{Var}_{\mathcal{Q}} = \mathrm{Var}_{\Phi_{\mathcal{Q}}}$ — множество всех переменных запроса $\mathcal{Q}$

Ответ на запрос $\mathcal{Q}$ — это подстановка θ, такая что $\mathrm{Dom}_{\theta} \subseteq \mathrm{Var}_{\mathcal{Q}}$

Правильный ответ на запрос $\mathcal{Q}$ к программе $\mathcal{P}$ — это ответ θ на запрос $\mathcal{Q}$, для которого выполнено соотношение

$$S_{\mathcal{P}} \models \forall \ldots (\Phi_{\mathcal{Q}}\theta)$$

ХЛП: декларативная семантика

Для технической простоты далее будем считать, что импликация

$$B_1 \,\&\, \ldots \,\&\, B_k \to A \;—$$

это форма записи дизъюнкта-правила

$$\neg B_1 \vee \cdots \vee \neg B_k \vee A,$$

а отрицание конъюнкции

$$\neg(C_1 \,\&\, \ldots \,\&\, C_m) \;—$$

это форма записи дизъюнкта-запроса

$$\neg C_1 \vee \cdots \vee \neg C_m$$

Тогда

- Формула $\Phi_{\mathcal{R}}$ для любого правила $\mathcal{R}$ — это дизъюнкт-правило
- Система формул $S_{\mathcal{P}}$ — это система дизъюнктов
- Формула $\neg\Phi_{\mathcal{Q}}$ для любого запроса $\mathcal{Q}$ — это дизъюнкт-запрос

ХЛП: декларативная семантика

Результаты $\mathcal{R}\theta$, $\mathcal{Q}\theta$ применения подстановки θ соответственно к правилу $\mathcal{R} = A \leftarrow B_1, \ldots, B_k$ и к запросу $\mathcal{Q} = ?C_1, \ldots, C_m$ определим так:

$$\mathcal{R}\theta = A\theta \leftarrow B_1\theta, \ldots, B_k\theta;$$
$$\mathcal{Q}\theta = ?C_1\theta, \ldots, C_m\theta$$

$\mathrm{Var}_{\mathcal{R}}$ и $\mathrm{Var}_{\mathcal{Q}}$ — так обозначим все переменные, содержащиеся в правиле $\mathcal{R}$ и в запросе $\mathcal{Q}$

Терминология, относящаяся к применению подстановок к выражениям (вариант, пример, основной пример, основное правило, основной запрос), без изменений переносится с дизъюнктов на правила и запросы

Утверждение. **Правило $\mathcal{R}'$ является вариантом (примером) [основным примером] правила $\mathcal{R}$ $\Leftrightarrow$ дизъюнкт $\Phi_{\mathcal{R}'}$ является вариантом (примером) [основным примером] дизъюнкта $\Phi_{\mathcal{R}}$ согласно той же подстановке**

Утверждение. **Запрос $\mathcal{Q}'$ является вариантом (примером) [основным примером] запроса $\mathcal{Q}$ $\Leftrightarrow$ дизъюнкт $\neg\Phi_{\mathcal{Q}'}$ является вариантом (примером) [основным примером] дизъюнкта $\neg\Phi_{\mathcal{Q}}$ согласно той же подстановке**

Правило SLD-резолюции

Пусть

- $\mathcal{Q} = ?C_1, \ldots, C_{i-1}, C_i, C_{i+1}, \ldots, C_m$ — запрос
- $\mathcal{R} = A \leftarrow B_1, \ldots, B_k;$ — правило ХЛП, такое что $\mathrm{Var}_{\mathcal{R}} \cap \mathrm{Var}_{\mathcal{Q}} = \emptyset$
- $\theta \in \mathrm{НОУ}(C_i, A)$

Тогда запрос

$$\mathcal{Q}' = (?C_1, \ldots, C_{i-1}, B_1, \ldots B_k, C_{i+1}, \ldots, C_m)\theta$$

называется SLD-резольвентой запроса $\mathcal{Q}$ и правила $\mathcal{R}$ для подцели C_i и унификатора θ

То, что $\mathcal{Q}'$ — SLD-резольвента $\mathcal{Q}$ и $\mathcal{R}$ как написано выше, будем обозначать записями

$$\mathcal{Q} \xrightarrow{\mathcal{R}, i, \theta} \mathcal{Q}' \quad \text{и} \quad \begin{array}{c} \mathcal{Q} \\ \mathcal{R} \end{array} \overset{i,\,\theta}{\rightrightarrows} \mathcal{Q}'$$

SLD-резолютивные вычисления

(Частичным) SLD-резолютивным вычислением программы $\mathcal{P}$, порождённым запросом $\mathcal{Q}$, называется последовательность запросов (конечная или бесконечная) вида

$$\mathcal{Q}_1 \xrightarrow{\mathcal{R}_1, k_1, \theta_1} \mathcal{Q}_2 \xrightarrow{\mathcal{R}_2, k_2, \theta_2} \dots ,$$

где:

- $\mathcal{Q}_1 = \mathcal{Q}$
- $\mathcal{R}_i$, k_i и θ_i, $i \in \{1, 2, \dots\}$ — соответственно вариант какого-либо правила из $\mathcal{P}$, номер подцели и унификатор для резольвенты $\mathcal{Q}_{i+1}$

По умолчанию в примерах будем использовать вариант $\mathcal{R}'$ правила $\mathcal{R}$, в котором ко всем переменным правила добавлены штрихи

Вычисление будем называть

- успешным, если оно конечно и оканчивается запросом $\square$
- тупиковым, если оно конечно, неуспешно и не может быть продолжено до более длинного вычисления

SLD-резолютивные вычисления

Для подстановки θ и множества переменных V записью $\theta|_V$ обозначим проекцию подстановки θ на множество V, то есть подстановку, устроенную так:

- Если $\mathrm{y} \in V$, то $\theta|_V(\mathrm{y}) = \theta(\mathrm{y})$
- Иначе $\theta|_V(\mathrm{y}) = \mathrm{y}$

Результатом конечного вычисления

$$\mathcal{Q}_1 \xrightarrow{\mathcal{R}_1, k_1, \theta_1} \dots \xrightarrow{\mathcal{R}_{n-1}, k_{n-1}, \theta_{n-1}} \mathcal{Q}_n$$

будем называть подстановку $\theta_1 \dots \theta_{n-1}|_{\mathrm{Var}_{\mathcal{Q}_1}}$

Ответ на запрос $\mathcal{Q}$ к программе $\mathcal{P}$ будем называть SLD-резолютивно вычислимым (или просто SLD-вычислимым), если существует успешное SLD-резолютивное вычисление, результатом которого является этот ответ

SLD-резолютивные вычисления

Теорема (о корректности операционной семантики ХЛП)
Для любой ХЛП $\mathcal{P}$ и любого запроса $\mathcal{Q}$ верно следующее: любой SLD-вычислимый ответ на $\mathcal{Q}$ к $\mathcal{P}$ является правильным ответом на $\mathcal{Q}$ к $\mathcal{P}$

Теорема (о полноте операционной семантики ХЛП)
Для любых программы $\mathcal{P}$ и запроса $\mathcal{Q}$ любой правильный ответ на запрос $\mathcal{Q}$ к $\mathcal{P}$ является частным случаем хотя бы одного SLD-вычислимого ответа на запрос $\mathcal{Q}$ к $\mathcal{P}$

Соотношение между семантиками ХЛП

Для ХЛП были введены две семантики:

- Декларативная:
  - программа и запрос — это формулы
  - правильный ответ — это такой, для которого запрос логически следует из программы
- Операционная:
  - шаг вычисления — это построение SLD-резольвенты
  - SLD-вычислимый ответ — это такой, который можно получить при помощи последовательности таких шагов

С одной стороны, операционная семантика может расцениваться как «чисто техническое» дополнение к декларативной, в котором говорится, как именно на компьютере вычисляются правильные ответы

С другой стороны, так как вычисление программы на компьютере — это важно и неизбежно, то хотелось бы иметь более «интуитивно ясное» понимание происходящего

Соотношение между семантиками ХЛП

Запрос «$?C_1, \ldots, C_k$» — это:

- Согласно декларативной семантике: обращённый к программе вопрос «для каких значений целевых переменных утверждения $C_1, \ldots C_k$ обязательно верны, если считать верным всё то, что записано в программе?»
- Согласно операционной семантике: список задач, записанных в виде атомов, общий ответ к которым нас интересует

Факт «$A;$» — это:

- Согласно декларативной семантике: утверждение о том, что A безусловно верно (*для любых значений используемых переменных*)
- Согласно операционной семантике: тривиальный способ решения задачи A, в котором сразу выдаётся ответ
  - Правило SLD-резолюции — это объявление об успешном решении задачи A с записью ответа в унификатор

Соотношение между семантиками ХЛП

Правило «$A \leftarrow B_1, \ldots, B_k$» для $k \geq 1$ — это:

- Согласно декларативной семантике: утверждение о том, что если верны все утверждения $B_1, \ldots, B_k$, то верно и A
- Согласно операционной семантике: запись способа решения задачи A, согласно которому следует решить (под)задачи $B_1, \ldots, B_k$ и скомбинировать ответ к A из ответов к этим подзадачам
  - Правило SLD-резолюции — переход к решению подзадач с записью способа комбинирования ответов в унификатор

ХЛП $\mathcal{P}$ — это:

- Согласно декларативной семантике: база знаний, посчитанных правильными, существенными и достаточными для ответов на интересующие вопросы
- Согласно операционной семантике: запись всевозможных способов решения (достаточного набора решений) интересующих задач и всех возникающих подзадач

Соотношение между семантиками ХЛП

С точки зрения декларативной семантики, порядок записи утверждений в программе и атомов в телах правил и запросов неважны:

- Конъюнкция коммутативна (можно переставлять атомы в телах без изменения смысла формул)
- Множество — это неупорядоченная совокупность элементов

С точки зрения операционной семантики, резонно предположение, что порядок записи утверждений и атомов в телах может быть важен:

- Зависят ли результат вычисления от выбора конкретных способов решения подзадач?
  - *Очевидно, что да*: ранее уже были примеры вычислений, порождённых одним и тем же запросом к одной и той же программе, но приводящих к разным ответам или вовсе не приводящих к ответу
- Зависит ли результат вычисления от выбора порядка решения подзадач?
  - *А это не так очевидно*

Сильная полнота операционной семантики

Правило выбора подцели — это отображение $\mathfrak{R}$, сопоставляющее каждому неуспешному вычислению $\mathfrak{S}$ номер подцели $\mathfrak{R}(\mathcal{Q})$ его последнего запроса

$\mathfrak{R}$-вычисление — это SLD-резолютивное вычисление, в котором для каждого префикса $\mathfrak{S} \to \mathcal{Q}$ резольвента $\mathcal{Q}$ строится для $\mathfrak{R}(\mathfrak{S})$-й подцели

Результаты $\mathfrak{R}$-вычислений называются $\mathfrak{R}$-вычислимыми ответами

Теорема (о сильной полноте операционной семантики ХЛП). Для любого правила выбора подцели $\mathfrak{R}$ и любого правильного ответа θ существует $\mathfrak{R}$-вычислимый ответ, являющийся обобщением θ

Стандартное правило выбора подцели

Согласно теореме о сильной полноте операционной семантики ХЛП, независимо от того, какое именно используется правило выбора подцели $\mathfrak{R}$, всегда есть возможность, придерживаясь правила $\mathfrak{R}$, получить всевозможные SLD-вычислимые ответы

Чтобы упростить анализ и использование ХЛП, как на практике, так и в теории зачастую используется стандартное правило выбора подцели: в каждом запросе всегда выбирается первая (самая левая) подцель

В изображении способа получения SLD-резольвенты $\mathcal{Q}_1 \xrightarrow{\mathcal{R},1,\theta} \mathcal{Q}_2$ согласно стандартному правилу номер подцели (1) будет опускаться: $\mathcal{Q}_1 \xrightarrow{\mathcal{R},\theta} \mathcal{Q}_2$

Деревья SLD-резолютивных вычислений ХЛП

Способ выбора правил программы при построении SLD-резолютивных вычислений существенно влияет на то, какой именно результат будет получен (и будет ли получен хоть какой-нибудь результат)

Этот факт нетруден для осознания:

- Правило — это описание одного из способов решения рассматриваемой задачи
- Бывают как успешные, так и неуспешные способы решения задач
  - Например, если заданный элемент списка располагается только в голове, то искать его в хвосте — заведомый неуспех
- Задачу можно успешно решать по-разному, получая разные ответы
  - Например, если в ответе требуется произвольный элемент списка, то можно выдать голову списка, или голову его хвоста, или голову хвоста его хвоста, ...

Чтобы умело рассуждать о способах выбора правил и соответствующих возможностях получать ответы, объединим всевозможные SLD-резолютивные вычисления программ в единую удобную структуру

Деревья SLD-резолютивных вычислений ХЛП

Дерево SLD-резолютивных вычислений для запроса $\mathcal{Q}$ к программе $\mathcal{P}$, построенное по правилу выбора подцели $\mathfrak{R}$, — это размеченное корневое ориентированное (возможно, бесконечное) дерево $T^{\mathfrak{R}}_{\mathcal{P},\mathcal{Q}}$, устроенное так:

1. Каждая вершина помечена запросом
2. Корень помечен запросом $\mathcal{Q}$
3. Дýги имеют вид $\mathcal{Q}_1 \xrightarrow{\mathcal{R},k,\theta} \mathcal{Q}_2$ и отвечают всем способам построения SLD-резольвенты $\mathcal{Q}_1$ и $\mathcal{R}$ для подцели $\mathfrak{R}(k)$ и унификатора θ (*с точностью до переименования*)
4. Дýги упорядочены согласно порядку записи правил в $\mathcal{P}$
5. Листьями являются запросы, у которых нет ни одной SLD-резольвенты с вариантами правил из $\mathcal{P}$ (в том числе $\square$)

Дерево вычислений для запроса $\mathfrak{Q}$ к программе $\mathcal{P}$, построенное по стандартному правилу выбора подцели, будет обозначаться так: $T_{\mathcal{P},\mathcal{Q}}$

Порядок дуг дерева в иллюстрациях — слева направо

Стратегии вычисления ХЛП

Деревья вычислений бывают разные:

- Конечные и бесконечные
- С конечным и с бесконечным числом ветвей
- Всюду успешные и с тупиками
- Содержащие один ответ, или много ответов, или ни одного ответа

Каждая ветвь в дереве $T^{\mathfrak{R}}_{\mathcal{Q},\mathcal{P}}$ отвечает SLD-резолютивному вычислению программы $\mathcal{P}$ для запроса $\mathcal{Q}$ согласно правилу $\mathfrak{R}$, и в дереве перечислены все такие вычисления

Значит, перебор всех таких вычислений можно устроить как **обход** дерева $T^{\mathfrak{R}}_{\mathcal{Q},\mathcal{P}}$

Стратегии вычисления ХЛП

Стратегия вычисления ХЛП состоит из

- правила выбора подцели и
- способа обхода дерева вычислений

Стратегия вычисления с правилом выбора подцели $\mathfrak{R}$ называется полной, для любого запроса $\mathcal{Q}$ к любой программе $\mathcal{P}$ она позволяет построить (перечислить) все успешные $\mathfrak{R}$-вычисления $\mathcal{P}$ для запроса $\mathcal{Q}$

Деревом SLD-резолютивных вычислений для запроса $\mathcal{Q}$ к программе $\mathcal{P}$, построенным согласно заданной стратегии вычисления, будем называть фрагмент дерева SLD-резолютивных вычислений для $\mathcal{Q}$, $\mathcal{P}$ и правила выбора подцели из стратегии, состоящий из всех вершин, посещаемых при обходе, и всех соединяющих их дуг

Стратегии вычисления ХЛП

Два основных вида обходов деревьев вычислений ХЛП:

- Обход в ширину: вершины дерева обходятся поярусно по неубыванию удалённости от корня
  - Это можно представить как перебор всех вычислений по возрастанию длины

- Обход в глубину: при обходе из вершины сначала последовательно по порядку обходятся вершины по исходящим дугам, и по окончании обхода выполняется возврат в предыдущую вершину
  - Это можно представить себе как последовательные попытки достроить (одно) рассматриваемое вычисление до более длинного пошагово всеми возможными способами по порядку правил

Стратегии вычисления ХЛП

Стратегия обхода в ширину полна:

- Из каждой вершины исходит конечное число дуг (не больше чем правил в программе), а значит, каждый ярус дерева конечен
- Каждая вершина каждого яруса рано или поздно будет посещена
- Каждое успешное вычисление конечно, а значит, завершается на некотором ярусе
- Значит, каждое успешное вычисление рано или поздно будет построено

Но у этой стратегии есть и серьёзный недостаток, присущий обходам больших графов в ширину:

- При обходе нужно хранить в памяти **все** вершины очередного яруса
- Число таких вершин может расти экспоненциально относительно номера яруса
  - *можете посчитать, на каком ярусе двоичного дерева или, например, дерева со степенью ветвления 4 содержится гугóл вершин*

Стратегии вычисления ХЛП

Стратегия, основанная на обходе в глубину

- неполна и
- чувствительна к порядку правил: даже если успешных вычислений сколь угодно много, в зависимости от порядка правил могут быть
  - найдены они все, или
  - найдены некоторые успешные вычисления, но не все, или
  - не найдены никакие успешные вычисления

Но использование этой стратегии позволяет избежать проблем с использованием памяти, возникающих при обходе в ширину, и она удобна и достаточно эффективна для использования на практике

Поэтому в стандартной стратегии вычисления ХЛП, используемой обычно в интерпретаторах, используются стандартное правило выбора подцели и обход в глубину

Выбор надлежащего порядка правил и атомов в телах правил при этом становится задачей программиста