Основы математической логики и логического программирования

В.А. Захаров

Билет 2.

Хорновские логические программы: синтаксис. Декларативная семантика логических программ. Операционная семантика логических программ. Стратегии вычисления логических программ.

ХОРНОВСКИЕ ЛОГИЧЕСКИЕ ПРОГРАММЫ

Синтаксис логических программ

Пусть $\sigma = \langle \mathit{Const}, \mathit{Func}, \mathit{Pred} \rangle$ — некоторая сигнатура, в которой определяются термы и атомы.

«заголовок» $::=$ «атом»

«тело» $::=$ «атом» $|$ «тело», «атом»

«правило» $::=$ «заголовок» $\leftarrow$ «тело»;

«факт» $::=$ «заголовок»;

«утверждение» $::=$ «правило» $|$ «факт»

«программа» $::=$ «пусто» $|$ «утверждение» «программа»

«запрос» $::=$ $\square$ $|$? «тело»

ХОРНОВСКИЕ ЛОГИЧЕСКИЕ ПРОГРАММЫ

Терминология

Пусть $G = ?C_1, C_2, \ldots, C_m$ — запрос. Тогда

- атомы $C_1, C_2, \ldots, C_m$ называются подцелями запроса G,
- переменные множества $\bigcup_{i=1}^{m} Var_{C_i}$ называются целевыми переменными ,
- запрос $\square$ называется пустым запросом ,
- запросы будем также называть целевыми утверждениями .

Для удобства обозначения условимся в дальнейшем факты A; рассматривать как правила $A \leftarrow$; с заголовком A и пустым телом.

ХОРНОВСКИЕ ЛОГИЧЕСКИЕ ПРОГРАММЫ

Как нужно понимать логические программы?

Главная особенность логического программирования — полисемантичность: одна и та же логическая программа имеет две равноправные семантики, два смысла.
Человек–программист и компьютер–вычислитель имеют две разные точки зрения на программу.

Программисту важно понимать, **ЧТО** вычисляет программа. Такое понимание программы называется декларативной семантикой программы.

Компьютеру важно «знать», **КАК** проводить вычисление программы. Такое понимание программы называется операционной семантикой программы.

ХОРНОВСКИЕ ЛОГИЧЕСКИЕ ПРОГРАММЫ

Как нужно понимать логические программы?

<table>
<tr><th>Декларативная семантика</th><th>Операционная семантика</th></tr>
<tr><td colspan="2">Правило $A_0 \leftarrow A_1, A_2, \ldots, A_n;$</td></tr>
<tr><td>Если выполнены условия $A_1, A_2, \ldots, A_n$, то справедливо и утверждение A_0.</td><td>Чтобы решить задачу A_0, достаточно решить задачи $A_1, A_2, \ldots, A_n$.</td></tr>
<tr><td colspan="2">Факт $A_0;$</td></tr>
<tr><td>Утверждение A_0 считается верным.</td><td>Задача A_0 объявляется решенной.</td></tr>
<tr><td colspan="2">Запрос $?C_1, C_2, \ldots, C_m$</td></tr>
<tr><td>При каких значениях целевых переменных будут верны все отношения $C_1, C_2, \ldots, C_m$?</td><td>Решить список задач $C_1, C_2, \ldots, C_m$.</td></tr>
</table>

ДЕКЛАРАТИВНАЯ СЕМАНТИКА

Более строгое описание семантик требует привлечения аппарата математической логики.

Логические программы и логические формулы

Каждому утверждению логической программы сопоставим логическую формулу:

Правило: $D' = \ A_0 \leftarrow A_1, A_2, \ldots, A_n$

$$D' = \forall X_1 \ldots \forall X_k (A_1 \& A_2 \& \ldots \& A_n \rightarrow A_0), \text{ где } \{X_1, \ldots, X_k\} = \bigcup_{i=0}^{n} Var_{A_i}$$

Факт: $D'' = \ A$

$$D'' = \forall X_1 \ldots \forall X_k A, \text{ где } \{X_1, \ldots, X_k\} = Var_A$$

Запрос: $G = \ ? \ C_1, C_2, \ldots, C_m$

$$G = C_1 \& C_2 \& \ldots \& C_m$$

Декларативная семантика

С точки зрения декларативной семантики,

- программные утверждения D и запросы G — это логические формулы,
- программа $\mathcal{P}$ — это множество формул (база знаний),
- а правильный ответ на запрос — это такие значения переменных (подстановка), при которой запрос оказывается логическим следствием базы знаний.

ДЕКЛАРАТИВНАЯ СЕМАНТИКА

Определение (правильного ответа)

Пусть $\mathcal{P}$ — логическая программа, G — запрос к $\mathcal{P}$ с множеством целевых переменных $Y_1, \ldots, Y_k$.

Тогда всякая подстановка $\theta = \{Y_1/t_1, \ldots, Y_k/t_k\}$ называется ответом на запрос G к программе $\mathcal{P}$.

Ответ $\theta = \{Y_1/t_1, \ldots, Y_k/t_k\}$ называется правильным ответом на запрос G к программе $\mathcal{P}$, если

$$\mathcal{P} \models \forall Z_1 \ldots \forall Z_N G\theta, \qquad \text{где } \{Z_1, \ldots, Z_N\} = \bigcup_{i=1}^{k} Var_{t_i}.$$

ДЕКЛАРАТИВНАЯ СЕМАНТИКА

Теорема (об основном правильном ответе)

Пусть $G = ?C_1, C_2, \ldots, C_m$ — запрос к хорновской логической программе $\mathcal{P}$. Пусть $Y_1, \ldots, Y_k$ — целевые переменные, $t_1, \ldots, t_k$ — основные термы.
Тогда подстановка $\theta = \{Y_1/t_1, \ldots, Y_k/t_k\}$ является правильным ответом на запрос G к программе $\mathcal{P}$ тогда и только тогда, когда $\mathcal{P} \models (C_1 \& \ldots \& C_m)\theta$.

ОПЕРАЦИОННАЯ СЕМАНТИКА ЛОГИЧЕСКИХ ПРОГРАММ

Концепция операционной семантики

Под операционной семантикой понимают правила построения **вычислений программы**. Операционная семантика описывает, КАК достигается результат работы программы.

Результат работы логической программы — это **правильный ответ** на запрос к программе. Значит, операционная семантика должна описывать метод вычисления правильных ответов.

Запрос к логической программе порождает задачу о логическом следствии. Значит, вычисление ответа на запрос должно приводить к решению этой задачи.

Таким методом вычисления может быть разновидность **метода резолюций**, учитывающая особенности устройства программных утверждений

SLD-РЕЗОЛЮТИВНЫЕ ВЫЧИСЛЕНИЯ

Определение (SLD-резолюции)

Пусть

- $G = ?\ C_1, \ldots, C_i, \ldots, C_m$ — целевое утверждение, в котором выделена подцель C_i,
- $D' = A'_0 \leftarrow \mathbf{A'_1}, \mathbf{A'_2}, \ldots, \mathbf{A'_n}$ — **вариант** некоторого программного утверждения, в котором $Var_G \cap Var_{D'} = \emptyset$,
- $\theta \in HOY(C_i, A'_0)$ — наиб. общ. унификатор подцели C_i и заголовка программного утверждения A_0.

Тогда запрос

$$G' = ?(C_1, \ldots, C_{i-1}, \mathbf{A'_1}, \mathbf{A'_2}, \ldots, \mathbf{A'_n}, C_{i+1}, \ldots, C_m)\theta$$

называется SLD-резольвентой программного утверждения D' и запроса G с выделенной подцелью C_i и унификатором θ.

SLD-РЕЗОЛЮТИВНЫЕ ВЫЧИСЛЕНИЯ

Определение (SLD-резолютивного вычисления)

Пусть

- $G_0 = ?\ C_1, C_2, \ldots, C_m$ — целевое утверждение,
- $\mathcal{P} = \{D_1, D_2, \ldots, D_N\}$ — хорновская логическая программа.

Тогда (частичным) SLD-резолютивным вычислением , порожденным запросом G_0 к логической программе $\mathcal{P}$ называется последовательность троек (конечная или бесконечная)

$$(D_{j_1}, \theta_1, G_1), (D_{j_2}, \theta_2, G_2), \ldots, (D_{j_n}, \theta_n, G_n), \ldots,$$

в которой для любого i, $i \geq 1$,

- $D_{j_i} \in \mathcal{P}$, $\theta_i \in Subst$, G_i — целевое утверждение (запрос);
- запрос G_i является SLD-резольвентой программного утверждения D_{j_i} и запроса G_{i-1} с унификатором θ_i.

SLD-РЕЗОЛЮТИВНЫЕ ВЫЧИСЛЕНИЯ

Определение (SLD-резолютивного вычисления)

Частичное SLD-резолютивное вычисление

$$comp = (D_{j_1}, \theta_1, G_1), (D_{j_2}, \theta_2, G_2), \dots, (D_{j_k}, \theta_n, G_n)$$

называется

- успешным вычислением (SLD-резолютивным опровержением), если $G_n = \square$;
- бесконечным вычислением , если *comp* — это **бесконечная** последовательность;
- тупиковым вычислением , если *comp* — это **конечная** последовательность, и при этом для запроса G_n невозможно построить ни одной SLD-резольвенты.

SLD-резолютивные вычисления

Определение (SLD-резолютивного вычисления)

Пусть

- $G_0 = ?\ C_1, C_2, \ldots, C_m$ — целевое утверждение с целевыми переменными $Y_1, Y_2, \ldots, Y_k$,
- $\mathcal{P} = \{D_1, D_2, \ldots, D_N\}$ — хорновская логическая программа,
- $comp\ =\ (D_{j_1}, \theta_1, G_1), (D_{j_2}, \theta_2, G_2), \ldots, (D_{j_n}, \theta_n, \Box)$ — успешное SLD-резолютивное вычисление, порожденное запросом G к программе $\mathcal{P}$.

Тогда подстановка $\theta = (\theta_1\theta_2\ldots\theta_n)|_{Y_1,Y_2,\ldots,Y_k}$, представляющая собой композицию всех вычисленных унификаторов $\theta_1,\ \theta_2, \ldots, \theta_n$, ограниченную целевыми переменными $Y_1, Y_2, \ldots, Y_k$,

называется вычисленным ответом на запрос G_0 к программе $\mathcal{P}$.

SLD-РЕЗОЛЮТИВНЫЕ ВЫЧИСЛЕНИЯ

Теперь у нас есть два типа ответов на запросы к логическим программам:

- **правильные ответы**, которые логически следуют из программы;
- **вычисленные ответы**, которые конструируются по ходу SLD-резолютивных вычислений.

Правильные ответы — это то, что мы хотим получить, обращаясь с вопросами к программе.

Вычисленные ответы — это то, что нам в действительности выдает компьютер (интерпретатор программы).

Какова связь между правильными и вычисленными ответами?

КОРРЕКТНОСТЬ ОПЕРАЦИОННОЙ СЕМАНТИКИ

Теорема (корректности операционной семантики относительно декларативной семантики)

Пусть

- $G_0 = ?\ C_1, C_2, \ldots, C_m$ — целевое утверждение,
- $\mathcal{P} = \{D_1, D_2, \ldots, D_N\}$ — хорновская логическая программа,
- θ — вычисленный ответ на запрос G_0 к программе $\mathcal{P}$.

Тогда θ — правильный ответ на запрос G_0 к программе $\mathcal{P}$.

ПОЛНОТА ОПЕРАЦИОННОЙ СЕМАНТИКИ

Теорема полноты (главная).

Пусть θ — правильный ответ на запрос $?G$ к хорновской логической программе $\mathcal{P}$.

Тогда существует такой вычисленный ответ η на запрос $?G$ к программе $\mathcal{P}$, что $\theta = \eta\rho$ для некоторой подстановки ρ.

ПРАВИЛА ВЫБОРА ПОДЦЕЛЕЙ

Определение.

Отображение R, которое сопоставляет каждому непустому запросу $G : ?C_1, C_2, \ldots, C_m$ одну из подцелей $C_i = R(G)$ в этом запросе, называется **правилом выбора подцелей** .

Для заданного правила выбора подцелей R вычисление запроса G к логической программе $\mathcal{P}$ называется **R-вычислением** , если на каждом шаге вычисления очередная подцель в запросе выбирается по правилу R.

Ответ, полученный в результате успешного R-вычисления, называется **R-вычисленным** .

Теорема сильной полноты

Каково бы ни было правило выбора подцелей R, если θ — правильный ответ на запрос G_0 к хорновской логической программе $\mathcal{P}$, то существует такой R-**вычисленный ответ** η, что равенство

$$\theta = \eta\rho$$

ДЕРЕВЬЯ ВЫЧИСЛЕНИЙ ЛОГИЧЕСКИХ ПРОГРАММ

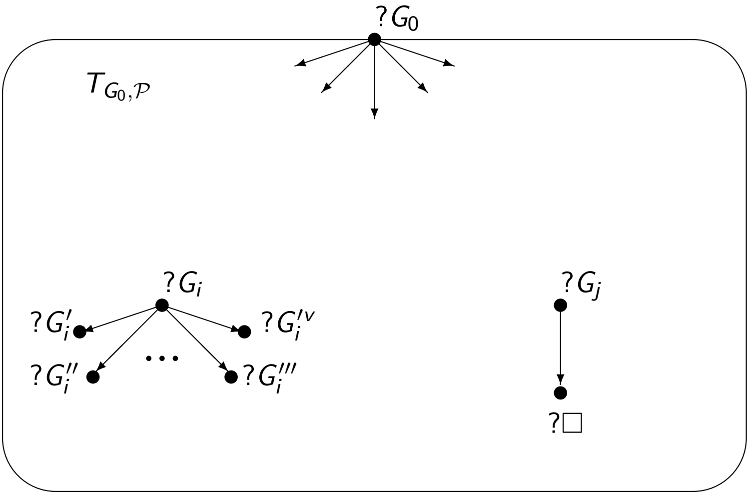
Определение

Деревом SLD-резолютивных вычислений запроса G_0 к логической программе $\mathcal{P}$ называется помеченное корневое дерево $T_{G_0,\mathcal{P}}$, удовлетворяющее следующим требованиям:

1. Корнем дерева является исходный запрос G_0;
2. Потомками каждой вершины G являются всевозможные SLD-резольвенты запроса G (при фиксированном стандартном правиле выбора подцелей);
3. Листовыми вершинами являются пустые запросы (завершающие успешные вычисления) и запросы, не имеющие SLD-резольвент (завершающие тупиковые вычисления).

ДЕРЕВЬЯ ВЫЧИСЛЕНИЙ ЛОГИЧЕСКИХ ПРОГРАММ

Иллюстрация



СТРАТЕГИИ ВЫЧИСЛЕНИЙ ЛОГИЧЕСКИХ ПРОГРАММ

Определение

Стратегией вычисления запросов к логическим программам называется алгоритм построения (обхода) дерева SLD-резолютивных вычислений $T_{G_0,\mathcal{P}}$ всякого запроса G_0 к произвольной логической программе $\mathcal{P}$

Стратегия вычислений называется **вычислительно полной**, если для любого запроса G_0 и любой логической программы $\mathcal{P}$ эта стратегия строит (обнаруживает) **все** успешные вычисления запроса G_0 к программы $\mathcal{P}$

СТРАТЕГИИ ВЫЧИСЛЕНИЙ ЛОГИЧЕСКИХ ПРОГРАММ

Фактически, стратегия вычисления — это одна стратегий обхода корневого дерева. Как известно, таких стратегий существует много, но среди них выделяются две наиболее характерные:

- **стратегия обхода в ширину** , при которой дерево строится (обходится) поярусно — вершина i-го не строится, до тех пор пока не будут построены все вершины $(i-1)$-го яруса;
- **стратегия обхода в глубину с возвратом** , при которой ветви дерева обходятся поочередно — очередная ветвь дерева не обходится, до тех пор пока не будут пройдены все вершины текущей ветви.

Стратегии вычислений логических программ

Стратегия обхода в ширину является вычислительно полной, поскольку

- каждый запрос имеет конечное число SLD-резольвент, и поэтому в каждом ярусе дерева SLD-резолютивных вычислений имеется конечное число вершин;
- каждое успешное вычисление завершается на некотором ярусе;
- и поэтому каждое успешное вычисление будет рано или поздно полностью построено.

Но строить интерпретатор логических программ на основе стратегии обхода в ширину **нецелесообразно**. При обходе дерева в ширину нужно обязательно хранить в памяти **все** вершины очередного яруса. Это требует большого расхода памяти. Например, в 100-м ярусе двоичного дерева содержится 2^{99} вершин. Вычислительных ресурсов всего земного шара не хватит, чтобы хранить информацию обо всех этих вершинах.

Стратегии вычислений логических программ

Стратегия обхода в глубину с возвратом основана на следующих принципах:

1. все программные утверждения упорядочиваются;
2. на каждом шаге обхода из текущей вершины G осуществляется переход
   - либо в новую вершину-потомок G', которая является SLD-резольвентой запроса G и первого по порядку программного утверждения D, ранее не использованного для этой цели;
   - либо в ранее построенную родительскую вершину G'' (откат), если все программные утверждения уже были опробованы для построения SLD-резольвент запроса G.

СТРАТЕГИИ ВЫЧИСЛЕНИЙ ЛОГИЧЕСКИХ ПРОГРАММ

Стратегия обхода в глубину с возвратом

- имеет эффективную реализацию: в памяти нужно хранить лишь запросы той ветви, по которой идет обход, и каждый запрос должен вести учет использованных программных утверждений;
- является, к сожалению, вычислительно неполной.

Стратегия обхода в глубину чувствительна к порядку расположения программных утверждений в логических программах. Результат вычисления запроса может измениться при перестановке программных утверждений. Поскольку соображения эффективности превалируют над требованиями вычислительной полноты, в качестве **стандартной стратегии** вычисления логических программ выбрана стратегия обхода в глубину. Программист должен сам выбрать нужный порядок расположения программных утверждений, чтобы стандартная стратегия вычисления отыскала все вычисленные ответы.

КОНЕЦ ОТВЕТА НА БИЛЕТ 2.