Дискретная математика

Логическое программирование

Константин Чухарев

Осень 2026

Термы и унификация

> *Алгоритм = логика + управление.*
>
> ***— Роберт Ковальский***

Вычисление как доказательство

- Обычная программа берёт данные, выполняет операции и выдаёт результат.
- Логическая программа описывает, что истинно, а машина по этому описанию ищет доказательство.

Пролог — язык логического программирования.

Вычисление — это построение доказательства.

Пример

`parent(alice, bob).` — факт.

`?- parent(alice, X).` — вопрос: кто ребёнок alice?

Ответ: `X = bob`.

Термы

Данные логической программы — термы.

Определение 1 (Терм)

- Терм — переменная, константа или $f(t_1, ..., t_n)$.
- Константа — функтор с $n = 0$: `alice`, `bob`, `[]`.
- Терм без переменных — **грунтовый** (*ground*).

Примеры термов

Пример

- `parent(alice, bob)` — терм: функтор `parent` и два аргумента-константы.
- `f(X, g(Y))` — вложенные термы: `g(Y)` — аргумент `f`.
- `alice` — константа: функтор с нулём аргументов.

Терм — дерево: функтор и аргументы.

Переменные (заглавные: `X`) — для того, что программа должна найти.

Терм не утверждает ничего.

Утверждение появляется, когда терм становится целью.

Подстановки

Доказательство уточняет, чему равны переменные.

Определение 2 (Подстановка)

Конечное отображение переменных в термы: $\sigma = \{X_1 \to t_1, ...\}$.

Применение $t\sigma$ — одновременная замена всех X_i.

Пример

$\sigma = \{X \to \text{bob}\}$ переводит `ancestor(alice, X)` в `ancestor(alice, bob)`.

Замена одновременная.

Подстановка не знает, что истинно — она лишь соглашение.

Унификация

Чтобы применить правило, голову и цель делают одинаковыми.

Определение 3 (Унификатор)

σ унифицирует s и t, если $s\sigma = t\sigma$.

Наиболее общий унификатор (МГУ) — всякий другой получается из него дополнительной подстановкой.

Алгоритм унификации

Определение 4 (Алгоритм унификации)

1. Одинаковые константы унифицируются, разные — нет.
2. Если обе стороны — одна и та же переменная, уравнение уже решено.
3. Если переменная равна терму, связываем её с ним, при условии что она не входит в этот терм.
4. Две структуры унифицируются, если совпали функторы и попарно унифицировались аргументы.
5. Разные функторы — невозможно.

Пример: унификация

Пример

Термы `p(f(X), Y)` и `p(Z, f(a))` унифицируются.

$f(X) = Z$ связывает $Z \to f(X)$.

$Y = f(a)$ связывает $Y \to f(a)$.

МГУ $-$ $\sigma = \{Z \to f(X), Y \to f(a)\}$. Общий терм $-$ `p(f(X), f(a))`.

X свободен, поэтому σ $-$ наиболее общий унификатор.

Occurs-check

Определение 5

Уравнение $X = t$ решается, только если X не входит в t.

Пример

$X = f(X)$ не имеет решения.

Без проверки получилась бы бесконечная цепочка: $X = f(X) = f(f(X)) = ...$

Теорема 1

Если термы унифицируемы, алгоритм находит МГУ за конечное число шагов: occurs-check не даёт построить бесконечную цепочку вида $X = f(X) = ...$

Унификация одновременно сравнивает и связывает.

Occurs-check [2]

Потому она и возвращает значения.

Факты и правила

Программа

Определение 6 (Клауза)

Определённая клауза: $A :- B_1, ..., B_n$ (в математической записи пишут $A \leftarrow B_1, ..., B_n$).

При $n = 0$ — **факт**, при $n > 0$ — **правило**. Программа — множество клауз. Цель — конъюнкция атомов.

Пример

```
parent(alice, bob).  parent(bob, carol).
ancestor(X, Y) :- parent(X, Y).
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).
```

Факты — база данных, правила — рецепты вывода.

Два прочтения

- Декларативное: «если все B_i истинны, то истинно A».
- Процедурное: «чтобы доказать A, докажи B_1, затем B_2...».

Программа описывает истину.

Машина читает описание как процедуру.

Определения принимаются буквально — задуманный смысл не зашит.

Пример

Запрос `?- ancestor(alice, Y).` даёт `Y = bob`, затем `Y = carol`.

SLD-резолюция

Выполнение — построение доказательства шаг за шагом.

Определение 7 (SLD-шаг)

Цель $\leftarrow A_1, ..., A_k$, клауза $A \leftarrow B_1, ..., B_m$ (свежие переменные).

θ — МГУ A_1 и головы A.

Новая цель: $\leftarrow (B_1, ..., B_m, A_2, ..., A_k)\theta$.

Пример: два шага

Пример

Запрос `?- ancestor(alice, bob)` — два SLD-шага.

1. Правило `ancestor(X, Y) :- parent(X, Y)` переименовано в `ancestor(_1, _2) :- parent(_1, _2)`. МГУ `{_1 -> alice, _2 -> bob}`. Новая цель `parent(alice, bob)`.
2. Факт `parent(alice, bob)`. Унификация пуста. Новая цель пуста.

Пустая цель — опровержение: ответ `yes`.

Синтаксис ведёт вывод, не заглядывая в смысл.

Поэтому доказательство становится вычислением.

Корректность и полнота

Теорема 2 (Корректность)

Каждый вычисленный ответ — логическое следствие программы.

Теорема 3 (Полнота)

Если цель следует из определённой программы, существует SLD-опровержение.

Доказательство

Каждый SLD-шаг — это шаг резолюции, а резолюция полна, поэтому корректность и полнота наследуются. Найти опровержение — задача поиска.

Поиск в глубину может уйти в бесконечную ветвь.

Поиск и бэктрекинг

Определение 8

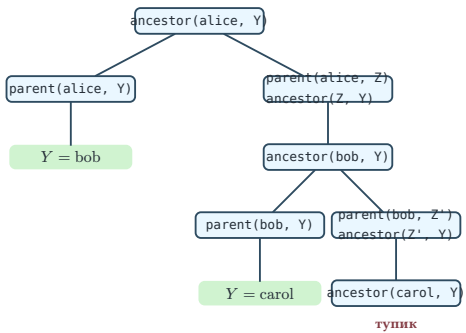
SLD-дерево — дерево, узлы которого — цели, а рёбра — SLD-шаги: из цели растёт по ветви на каждую подходящую клаузу.

Пример: дерево вывода

Пример

Запрос `?- ancestor(alice, Y).` даёт две ветви: правило `ancestor(X, Y) :- parent(X, Y)` и рекурсивное правило.

Каждая клауза, чья голова унифицируется с целью, — своя ветвь дерева.



Выбор ветви

SLD-дерево содержит все возможные выводы. Пролог обходит его в глубину:

- выбирает первую подходящую клаузу.
- при неуспехе откатывается и пробует следующую.

Порядок клауз и целей — программа: от него зависит, какой вывод будет найден.

Рекурсия и списки

- Списки — термы: `[1, 2, 3]` — сокращение для вложенных `.(1, .(2, .(3, [])))`.
- Рекурсия идёт по структуре списка: обрабатываем голову, а хвост отдаём рекурсивному вызову.
- Корректность доказывается индукцией по длине списка.

Терминация требует базового случая.

Как индукция: база + шаг.

Пример

```
member(X, [X|_ ]).
member(X, [_|T]) :- member(X, T).
```

Что мы поняли?

- Логическая программа описывает, что истинно, а вычисление — это построение доказательства.
- Терм — дерево из функтора и аргументов. Унификация находит МГУ, а occurs-check отсекает $X = f(X)$.
- Клауза — факт или правило. SLD-резолюция ведёт вывод синтаксически, не заглядывая в смысл.
- Корректность и полнота не гарантируют успех поиска: обход в глубину может уйти в бесконечную ветвь.

Вопросы для самопроверки

- Что такое терм и чем переменная отличается от константы?
- Что такое подстановка и почему замена одновременная?
- Унифицируются ли $p(f(X), Y)$ и $p(Z, f(a))$? Найдите МГУ.
- Почему $X = f(X)$ не имеет решения?
- Что добавляет унификация к сравнению термов?
- Запишите программу «родство» фактами и правилами.
- Объясните SLD-шаг на примере.
- Почему полнота не гарантирует, что поиск найдёт ответ?