Лекция 4

Алгоритм унификации для термов

(Конспект: Р. Мясников)

Disclaimer: *Конспект приводится "as is", я его не смотрел. – – Э.А.*

4.1 Постановка задачи

Определим понятие терма. Во-первых, термом является любой элемент множества констант C. Во-вторых, термом является любой элемент множества переменных X. В-третьих, термом является любая функция от некоторых термов; множество допустимых функций обозначим F. Множество термов обозначим L.

Пусть есть два терма t и s. Наша задача будет заключаться в нахождении такой подстановки γ, что $t\gamma = s\gamma$.

В качестве формализации понятия подстановки можно рассмотреть распространение отображения γ из X в L на C и F следующим образом: если $c \in C$, то $c\gamma = c$, а если $f \in F, f = f(x_1, ...x_n)$, то $f\gamma = f(x_1\gamma, ..., x_n\gamma)$.

Унификатором термов t и s назовем такую подстановку γ, что $s\gamma = t\gamma$.

Наиболее общим унификатором термов t и s назовем такую подстановку σ, что для любого унификатора γ можно указать такую подстановку α, что $\gamma = \sigma\alpha$.

4.2 Алгоритм унификации

"Лобовой"алгоритм унификации имеет экспоненциальную оценку времени работы. Мы рассмотрим предложенный Эрбраном алгоритм, имеющий линейную оценку времени работы.

Текущей конфигурацией назовем пару (P, S), где P - текущая задача, а S - текущая подстановка. Начальная конфигурация задается задачей $\{t = s\}$ и подстановкой $\emptyset$. Результатом работы алгоритма должна явиться либо конфигурация с пустой задачей, в таком случае финальная подстановка будет соответствовать искомому унификатору, либо заключение о неразрешимости исходной задачи.

Изменение конфигурации в процессе работы алгоритма происходит на каждом шаге по одному из следующих 6 правил:

1) $\{S = S\} \bigcap P; Q => P; Q$

2) $\{f(t_1, ..., t_n) = f(s_1, ..., s_n)\} \bigcap P; Q => \{t_1 = s_1)\} bigcap... \bigcap \{t_n = s_n\} \bigcap P; Q$

3) $\{g(...) = f(...)\} \bigcap P; Q =>$неразрешимая задача

4) $\{t = x\} \bigcap P; Q => \{x = t\} \bigcap P; Q$

5) $\{x = t\} \bigcap P; Q =>$неразрешимая задача, при условии $x \in var(t)$ (и, кроме того, естественно, x отлично от t)

6) $\{x = t\} \bigcap P; Q => P(x- > t); Q(x- > t) \bigcap \{x = t\}$

Здесь через $var(t)$ обозначено множество переменных, участвующих в терме t, а через $P(x- > t)$ - подстановка t вместо x в рамках P.

Лемма 4.1. *независимо от начальной конфигурации, за конечное число шагов алгоритм заканчивает работу в одном из двух специфицированных финальных состояний (либо задача пуста, либо установлена нерешаемость задачи).*

Доказательство. Доказательство проводится по индукции. Введем дополнительную характеристику конфигурации - сложность, определяемую как тройку $< n_1, n_2, n_3 >$, где $n_1 = |var(P)|$ (P - задача), $n_2 = |P|$ - длина строки, n_3 - число "неперевернутых"равенств вида $t = x, t \notin var(p)$ в P. Применение каждого правила уменьшает эту сложность. □

Лемма 4.2. *Пусть на каком-то шаге совершен переход $P; Q => P_1; Q_1$. Тогда для подстановка γ - решение(в смысле унификатора) P при условии Q тогда и только тогда, когда γ - решение P_1 при условии Q_1.*

Доказательство. Доказательство проводится отдельно для каждого правила; в каждом из 6 случаев утверждение леммы очевидно. □

Лемма 4.3. *Рассмотрим начальную конфигурацию* $P;\emptyset$ *и финальную конфигурацию* $\emptyset;Q$*. Тогда подстановка* S *унифицирует любую подзадачу* P*.*

Доказательство. Доказательство леммы получается обратным последовательным применением предыдущей леммы. □

Лемма 4.4. *Пусть* γ *унифицирует любую подзадачу* P*. Тогда алгоритм с начальной конфигурацией* $P;\emptyset$ *заканчивает работу в финальном состоянии, соответствующем пустой задаче.*

Доказательство. По предыдущей лемме алгоритм не может завершиться обнаружением нерешаемости задачи. □

Докажем линейную оценку времени работы алгоритма. Действительно, применение каждого правила работает const время (на RAM-машине или машине Тьюринга, например), при этом каждый шаг обрабатывает, по крайней мере, один символ равенства. Необходимо, однако, заметить, что сказанное справедливо только при организации действий с множествами за линейное время.