Метод резолюций

Разрешимость исчислений

«Извини, Теодор, но это ты очень странно рассуждаешь. Бессмыслица — искать решение, если оно и так есть. Речь идет о том, как поступить с задачей, которая решения не имеет. Это глубоко принципиальный вопрос, который, как я вижу, тебе, прикладнику, к сожалению, не доступен.»

А. и Б. Стругацкие, «Понедельник начинается в субботу»

- Разрешимы: КИВ, ИИВ.
- Неразрешимы: всё остальное (ИП, ФА, ZF(C), ...)

Однако, (1) разрешимости хочется и (2) человек как-то умеет.

Ищем доказательство в исчислении предикатов: упрощение задачи

- По теореме о полноте можем рассматривать ($\models$) вместо ($\vdash$). Напомним: $\models \alpha$, если для всех $M = \langle D, F, P, E \rangle$ выполнено $M \models \alpha$.

- Что мешает:
  1. слишком сложные формулы — кванторы по бесконечным множествам;
  2. слишком больше разнообразие D, включая несчётные;
  3. даже $D = \mathbb{N}$ в формальной арифметике представляет проблему.

- Будем последовательно бороться:
  1. упростим формулу (борьба с кванторами);
  2. заменим произвольное D на какое-то рекурсивно-перечислимое множество, устроенное некоторым фиксированным образом (борьба с разнообразием D);
  3. устроим правильный перебор, позволяющий быстро находить решения, если они есть (борьба с бесконечностью

Упрощаем формулу α. Сколемизация

1. Предварённая форма (поверхностные кванторы) — *для примера* возьмём чередующиеся:

$$\beta := \forall x_1.\exists x_2.\forall x_3.\exists x_4 \ldots \forall x_{n-1}.\exists x_n.\varphi$$

2. Убрать кванторы существования: заменим x_{2k} функциями Сколема $e_{2k}(x_1, x_2, \ldots, x_{2k-1})$. Получим:

$$\gamma := \forall x_1.\forall x_3 \ldots \forall x_{n-1}.\varphi[x_2 := e_2(x_1), x_4 := e_4(x_1, x_3), \ldots, x_n := e_n($$

3. ДНФ (c конъюнктов, в каждом $d(c)$ дизъюнктов):

$$\delta := \forall x_1.\forall x_3 \ldots \forall x_{n-1}.\bigwedge_c \left(\bigvee_{i=\overline{1,d(c)}} (\neg) P_i(\theta_i) \right)$$

4. $\vdash \alpha$ эквивалентно $\models \alpha$ и эквивалентно выполнимости δ при всех D (найдутся e_i, что $[\![\delta]\!] = \text{И}$).

Шаги рассуждения

1. Упростим формулу — поверхностные кванторы всеобщности, сколемизация.
2. Заменяем D.
3. Правильный перебор

Эрбранов универсум.

Определение

$H_0(\varphi)$ — все константы в формуле φ (либо особая константа a, если констант в φ нет)
$H_{k+1}(\varphi)$ — $H_k(\varphi)$ и все функции от значений $H_k(\varphi)$ (как строки)
$H = \cup H_n(\varphi)$ — основные термы.

Пример

$P(a) \vee Q(f(b))$:
$H_0 = \{a, b\}$
$H_1 = \{a, b, f(a), f(b)\}$
$H_2 = \{a, b, f(a), f(b), f(f(a)), f(f(b))\}$
...
$H = \{f^{(n)}(x) \mid n \in \mathbb{N}_0, x \in \{a, b\}\}$

Выполнимость не теряется. Заменяем D на H

Теорема

Формула выполнима тогда и только тогда, когда она выполнима на Эрбрановом универсуме.

Доказательство.

($\Rightarrow$) Пусть $M \models \forall \overline{x}.\varphi$. Тогда построим отображение $\mathrm{eval} : H \to M$ (смысл названия вдохновлён языками программирования: eval("$f(f(b))$") перейдёт в $f(f(b))$, где f и b — из M).

Предикатам дадим согласованную оценку: $P_H(t_1, \ldots, t_n) = P_M(h(t_1), \ldots, h(t_n))$. Очевидно, любая формула сохранит своё значение, кванторы всеобщности по меньшему множеству также останутся истинными.

($\Leftarrow$) Очевидно. $\square$

Противоречивость системы дизъюнктов

Определение

Система дизъюнктов $\{\delta_1, \ldots, \delta_n\}$ *противоречива, если для каждой интерпретации* M *найдётся* δ_k *и такой набор* $d_1 \ldots d_v$, *что* $[\![\delta_k]\!]^{x_1:=d_1,\ldots,x_v:=d_v} = Л$.

Теорема

Система дизъюнктов противоречива, если она невыполнима на Эрбрановом универсуме.

Доказательство.

Контрапозиция теоремы о выполнимости + разбор определения. □

Основные примеры.

Определение

Дизъюнкт с подставленными основными термами вместо переменных называется основным примером. Системой основных примеров $\mathcal{E}$ назовём множество основных примеров. А именно, рассмотрим $\delta_1 \,\&\, \delta_2 \,\&\, \cdots \,\&\, \delta_n$.

$\mathcal{E} = \{$ *все возможные основные примеры* $\delta_k \mid \mathcal{M} \not\models \delta_k, \mathcal{M}$ *из* $H \}$

Теорема

Система дизъюнктов S противоречива тогда и только тогда, когда система всевозможных основных примеров $\mathcal{E}$ противоречива

Доказательство.

Для некоторой эрбрановой интерпретации дизъюнкт δ_k опровергается тогда и только тогда, когда соответствующая ему подстановка в $\mathcal{E}$ опровергается. □

Теорема Гёделя о компактности

Теорема

Если Γ *— некоторое семейство бескванторных формул, то* Γ *имеет модель тогда и только тогда, когда любое его конечное подмножество имеет модель.*

Доказательство.

($\Leftarrow$): очевидно

($\Rightarrow$): пусть каждое конечное подмножество имеет модель.

Тогда Γ непротиворечиво:

Иначе, для любой σ выполнено $\Gamma \vdash \sigma$. В частности, для $\gamma \in \Gamma$ выполнено $\Gamma \vdash \neg\gamma$. Доказательство имеет конечную длину, и использует конечное количество формул $\gamma_1, \ldots, \gamma_n \in \Gamma$. Тогда рассмотрим $\Sigma = \{\gamma, \gamma_1, \ldots, \gamma_n\}$, и модель $\mathcal{S}$ для неё. Тогда:

1. $\models_S \gamma$ (определение модели)
2. $\models_S \neg\gamma$ (теорема о корректности: $\Sigma \vdash \neg\gamma$, значит $\Sigma \models \neg\gamma$ в любой модели)

Значит, Γ имеет модель (вспомогательная теорема к теореме Гёделя о полноте). $\square$

Теорема Эрбрана

Теорема (Эрбрана)

Система дизъюнктов S противоречива тогда и только тогда, когда существует конечное противоречивое множество основных примеров системы дизъюнктов S

Доказательство.

($\Leftarrow$) Пусть $\delta_1[\overline{x} := \overline{\theta}], \ldots, \delta_k[\overline{x} := \overline{\theta}]$ — противоречивое множество примеров дизъюнктов. Тогда интерпретация $\overline{\theta}$ опровергает хотя бы один из δ_k и система противоречива.
($\Rightarrow$) Если S противоречива, то значит, множество основных примеров S противоречиво (по теореме о выполнимости Эрбранова универсума). Тогда по теореме компактности в нём найдётся конечное противоречивое подмножество. $\square$

Шаги рассуждения

1. Упростим формулу — поверхностные кванторы всеобщности, сколемизация.
2. Упрощаем D — заменили на H, свели к перебору основных примеров.
3. Правильный перебор.

Пример: как проверяем выполнимость формулы?

Допустим, формула: $(\forall x.P(x)\ \&\ P(x'))\ \&\ \exists x.\neg P(x'''')$

1. Поверхностные кванторы, сколемизация, ДНФ:
   $(\forall x.P(x))\ \&\ (\forall x.P(x'))\ \&\ (\neg P(e))$
2. Строим Эрбранов универсум: $H = \{e, e', e'', e''', \dots\}$
3. Если есть противоречие, то среди основных примеров:

$$\mathcal{E} = \{P(e), P(e'), P(e''), P(e'''), P(e''''), \neg P(e''''), \dots\}$$

Напомним, $\mathcal{E}$ — подстановки элементов H вместо переменных под кванторами. Причём, либо $\models \&\, E$, либо противоречие достигается на конечном подмножестве (т. Эрбрана).
Добавляем по примеру и проверяем. $P(e)$ при $[\![P(e)]\!] = \text{И}$.
$P(e')$ при $[\![P(e')]\!] = \text{И}$.
...
$P(e'''')$ при $[\![P(e'''')]\!] = \text{И}$.
$\neg P(e'''')$ при $[\![P(e'''')]\!] = \text{Л}$. Противоречие.

Правило резолюции (исчисление высказываний)

Пусть даны два дизъюнкта, $\alpha_1 \vee \beta$ и $\alpha_2 \vee \neg\beta$. Тогда следующее правило вывода называется правилом резолюции:

$$\frac{\alpha_1 \vee \beta \qquad \alpha_2 \vee \neg\beta}{\alpha_1 \vee \alpha_2}$$

Теорема

Система дизъюнктов противоречива, если в процессе всевозможного применения правила резолюции будет построено явное противоречие, т.е. найдено два противоречивых дизъюнкта: β *и* $\neg\beta$.

Расширение правила резолюции на исчисление предикатов

Заметим, что правило резолюции для исчисления высказываний не подойдёт для исчисления предикатов.

$$S = \{P(x), \neg P(0)\}$$

Здесь $P(x)$ противоречит $\neg P(0)$, но правило резолюции для исчисления высказываний здесь неприменимо, потому что x можно заменять, это не константа:

$$\frac{P(x) \qquad \neg P(0)}{???}$$

Нужно заменять $P(x)$ на основные примеры, и искать среди них. Модифицируем правило резолюции для этого.

Алгебраические термы

Определение

Алгебраический терм

$$\theta := x | (f(\theta_1, \ldots, \theta_n))$$

где x — переменная, $f(\theta_1, \ldots, \theta_n)$ — применение функции. Напомним, что константы — нульместные функциональные символы, собственно переменные будем обозначать последними буквами латинского алфавита.

Определение

Система уравнений в алгебраических термах

$$\begin{cases} \theta_1 = \sigma_1 \\ \vdots \\ \theta_n = \sigma_n \end{cases}$$

где θ_i и σ_i — термы

Уравнение в алгебраических термах

Определение

$\{x_i\} = X$ — множество переменных, $\{\theta_i\} = T$ — множество термов.

Определение

Подстановка — отображение вида: $\pi_0 : X \to T$, тождественное почти везде.
$\pi_0(x)$ может быть либо $\pi_0(x) = \theta_i$, либо $\pi_0(x) = x$.

Доопределим $\pi : T \to T$, где

1. $\pi(x) = \pi_0(x)$
2. $\pi(f(\theta_1, \ldots, \theta_k)) = f(\pi(\theta_1), \ldots, \pi(\theta_k))$

Определение

Решить уравнение в алгебраических термах — найти такую наиболее общую подстановку π, что $\pi(\theta_1) = \pi(\theta_2)$. Наиболее общая подстановка — такая, для которой другие подстановки являются её частными случаями.

Задача унификации

Определение

Пусть даны формулы α и β. Тогда решением задачи унификации будет такая наиболее общая подстановка $\pi = \mathcal{U}\big[\alpha, \beta\big]$, что $\pi(\alpha) = \pi(\beta)$.
Также, η назовём наиболее общим унификатором.

Пример

- *Формулы $P(a, g(b))$ и $P(c, d)$ не имеют унификатора (мы считаем, что a, b, c, d — нульместные функции, а f — одноместная функция).*
- *Проверим формулу на соответствие 11 схеме аксиом:*

$$(\forall x.P(x)) \to P(f(t, g(t), y))$$

Пусть $\pi = \mathcal{U}\big[P(x), P(f(t, g(t), y))\big]$, тогда $\pi(x) = f(t, g(t), y)$.

Правило резолюции для исчисления предикатов

Определение

Пусть σ_1 и σ_2 — подстановки, заменяющие переменные в формуле на свежие. Тогда правило резолюции выглядит так:

$$\frac{\alpha_1 \vee \beta_1 \qquad \alpha_2 \vee \neg\beta_2}{\pi(\sigma_1(\alpha_1) \vee \sigma_2(\alpha_2))}\ \pi = \mathcal{U}\left[\sigma_1(\beta_1), \sigma_2(\beta_2)\right]$$

σ_1 и σ_2 разделяют переменные у дизъюнктов, чтобы π не осуществила лишние замены, ведь
$\vdash (\forall x.P(x)\ \&\ Q(x)) \leftrightarrow (\forall x.P(x))\ \&\ (\forall x.Q(x))$, но
$\not\vdash (\forall x.P(x) \vee Q(x)) \rightarrow (\forall x.P(x)) \vee (\forall x.Q(x))$.

Пример

$$\frac{Q(x) \vee P(x) \quad \neg P(a) \vee T(x)}{Q(a) \vee T(x'')}$$ *подстановки:* $\sigma_1(x) = x', \sigma_2(x) = x'', \pi(x$

Метод резолюции

Ищем $\vdash \alpha$.

1. будем искать опровержение $\neg\alpha$.
2. перестроим $\neg\alpha$ в ДНФ.
3. будем применять правило резолюции, пока получаем новые дизъюнкты и пока не найдём явное противоречие (дизъюнкты вида β и $\neg\beta$).

Если противоречие нашлось, значит, $\vdash \neg\neg\alpha$. Если нет — значит, $\vdash \neg\alpha$. Процесс может не закончиться.

SMT-решатели

Обычно требуется не логическое исчисление само по себе, а теория первого порядка. То есть, «Satisfability Modulo Theory», «выполнимость в теории» — вместо SAT, выполнимости.

- Иногда можно вложить теорию в логическое исчисление, даже в исчисление высказываний: $\overline{S_2 S_1 S_0} = \overline{A_1 A_0} + \overline{B_1 B_0}$

$$
\begin{array}{ll}
S_0 = A_0 \oplus B_0 & C_0 = A_0 \,\&\, B_0 \\
S_1 = A_1 \oplus B_1 \oplus C_0 & C_1 = (A_1 \,\&\, B_1) \vee (A_1 \,\&\, C_0) \vee (B_1 \,\&\, C_0) \\
S_2 = C_1 &
\end{array}
$$

- А можно что-то добавить прямо на уровень унификации / резолюции: Например, можем зафиксировать арифметические функции — и производить вычисления в правиле резолюции вместе с унификацией.
  Тогда противоречие в $\{x = 1 + 3 + 1, \neg x = 5\}$ можно найти за один шаг.

Уточнённые типы (Refinement types), LiquidHaskell

Определение

(Неформальное) Уточнённый тип — тип вида $\{\tau(x) \mid P(x)\}$*, где* P *— некоторый предикат.*

Пример на LiquidHaskell:

```
data [a] <p :: a -> a -> Prop> where
   | []  :: [a] <p>
   | (:) :: h:a -> [a<p h>]<p> -> [a]<p>
```

- `h:a` — голова (h) имеет тип a
- `[a<p h>]<p>` — хвост состоит из значений типа a, уточнённых p — $\{t : a \mid p\ h\ t\}$ (карринг: `a <p h>`).

```
{-@ type IncrList a = [a] <{\xi xj -> xi <= xj}> @-}
{-@ insertSort    :: (Ord a) => xs:[a] -> (IncrList a) @-}
insertSort []     = []
insertSort (x:xs) = insert x (insertSort xs)
```