Лекция 3

Моделирование секущих плоскостей в системах Фреге. Оптимальные системы (30.09.2010)

(Конспект: А. Бешенов)

3.1 Моделирование секущих плоскостей в системах Фреге

Теорема 3.1. *Система секущих плоскостей полиномиально моделируется в системах Фреге.*

Основные шаги доказательства. Можно считать, что система секущих плоскостей оперирует неравенствами вида $\sum_t c_t\, y_t \geq c$, где c_t, c — положительные целые числа, а y_t — переменные либо отрицания переменных (в любом случае, они по существу принимают значения из $\{0,1\}$). (От отрицательных коэффициентов в нетривиальных неравенствах всегда можно избавиться, перейдя к отрицаниям: например, $\cdots - c_t\, x_t + \cdots \geq c$ заменяется на $\cdots + c_t\,(1 - x_t) + \cdots \geq c + c_t$.)

Будем кодировать числа логическими формулами. Именно, числу соответствует вектор формул: по одной формуле (предикату) на каждый бит.

Умножение на 0 и 1 записывается очевидным образом: вектор формул для $c_t\, y_t$ имеет вид $(Y_0, \ldots, Y_k)$, где $Y_i = y_t$, если i-й бит константы c_t равен единице, и $Y_i =$ false в противном случае.

Научимся вычислять сумму $\sum_t c_t\, y_t$ и сравнивать результат с c. Можно попробовать вычислять сумму наивным образом:

$$Add((F_0, \ldots, F_k), (G_0, \ldots, G_k)) = \left(F_i \oplus G_i \oplus \bigvee_{0 \leq j < i} \left(F_j \wedge G_j \wedge \bigwedge_{i < l < j} (F_l \oplus G_l) \right) \right)_i .$$

Но тогда, просуммировав (даже двоичным деревом) n членов суммы $\sum_t c_t\, y_t$, мы получим формулы размера порядка $k^{\log n}$, что превосходит любой полином для любого

растущего количества битов $k = k(n)$.

Используем «продвинутое суммирование», которое по трем числам $\vec{F}$, $\vec{G}$, $\vec{H}$ выдает два числа $SAdd$ и $CAdd$ (неполную сумму и перенос):

$$SAdd(\vec{F}, \vec{G}, \vec{H}) = (F_i \oplus G_i \oplus H_i)_i,$$
$$CAdd(\vec{F}, \vec{G}, \vec{H}) = ((F_i \wedge G_i) \vee (F_i \wedge H_i) \vee (G_i \wedge H_i))_i .$$

При таком суммировании $\sum_t c_t\, y_t$ (деревом глубины $\sim \log_{3/2} n$) в конечном счете у нас останется два числа — последние значения $SAdd$ и $CAdd$. Их уже можно сложить стандартным способом. Полученная запись $\sum_t c_t\, y_t$ будет полиномиального размера от количества переменных и количества битов в коэффициентах c_t; обозначим такое суммирование Sum.

Сравнение чисел в битовой записи производится так:

$$\vec{F} > \vec{G} \iff \bigvee_j \left(F_i \wedge \neg G_i \wedge \bigwedge_{j>i} (F_j = G_j) \right).$$

В качестве числа битов k достаточно выбрать такое число, чтобы большее количество битов не понадобилось при переписывании доказательства; легко видеть, что можно ограничиться $\lceil \log(2Mn) \rceil$, где M — наибольший коэффициент, появляющийся в исходном доказательстве.

Аксиомы натуральных чисел, которые нам понадобятся в дальнейшем, остаются в качестве упражнения: например, нужно доказать $Add(\vec{F}, 0) = \vec{F}$ и $Sum(\vec{0}) < 1$.

Теперь будем переписывать исходное доказательство в системе секущих плоскостей в систему Фреге. Рассмотрим правило сложения

$$\frac{\sum_t c_t\, y_t \geq c \qquad \sum_t d_t\, y_t \geq d}{\sum_t (c_t + d_t)\, y_t \geq c + d}$$

Нам нужно доказать[1] по индукции, что

$$Add(Sum((c_t\, y_t)_t), Sum((c_t\, y_t)_t)) = Sum(((c_t + d_t)\, y_t)_t).$$

[1] Речь о доказательстве в системе Фреге, которая моделирует секущие плоскости, а не о доказательстве в метаязыке.

Равенство $Add(c_t\,y_t, d_t\,y_t) = (c_t + d_t)\,y_t$ доказывается через разбор случаев $y_t = 0$ и $y_t = 1$ (понятно, что корректность суммирования констант доказывается за количество шагов, сравнимое с количеством шагов в этом суммировании). Это база индукции, а для доказательства индукционного перехода надо вычленять из Sum по одному слагаемому и суммировать их аналогично базе (вычленение требует, в частности, доказательства того, что $CAdd$ и $SAdd$ эквивалентны обычному суммированию:

$$Add(CAdd(\vec{F},\vec{G},\vec{H}), SAdd(\vec{F},\vec{G},\vec{H})) = Add(Add(\vec{F},\vec{G}),\vec{H}),$$

которое, конечно, следует из их определений).

Приведённое выше рассуждение отражает лишь ситуацию, когда в суммах присутствует действительно один и тот же набор y_t. Однако некоторые из этих членов могут относиться к противоположным литералам: y_t и $\neg y_t$. В этом случае база индукции для $c_t y_t + d_t \neg y_t$ (пусть, не умаляя общности, $c_t \geq d_t$) потребует доказательства $c_t y_t + d_t \neg y_t = (c_t - d_t) y_t + d_t$, которое также можно проделать разбором случаев $y_t = 0$ и $y_t = 1$, сложением и вычитанием в столбик. Впоследствии "отщепившиеся" константы (в данном случае d_t) должны быть учтены при сравнении с тем, насколько по сравнению с $c_t + d_t$ должна измениться правая часть. Выглядит это рассуждение громоздко, но, к счастью, мы не должны его проделывать "в метаязыке", нам достаточно убедиться, что для всякого выбора натуральных чисел c_t, d_t, c, t и расстановки знаков в y_t такое доказательство возможно проделать в системе Фреге (доказательство будет выглядеть по-разному для каждого варианта, главное, чтобы оно было полиномиальной длины!).

Далее докажем, что $\vec{F} \geq \vec{G} \wedge \vec{F'} \geq \vec{G'} \supset Add(\vec{F},\vec{F'}) \geq Add(\vec{G},\vec{G'})$ (в битовом представлении легко разбираются все возможные случаи — в каком именно бите имеется строгое неравенство (если оно есть)). Это завершает доказательство правила сложения.

Для моделирования умножения и деления на константу достаточно ограничиться реализацией умножения. Чтобы не иметь дело с делением, мы просто записываем в другом виде правило округления:

$$\frac{\sum_t (a\,c_t)\,y_t \geq a\,c + r}{\sum_t c_t\,y_t \geq c + 1} \qquad (0 < r < a)$$

Чтобы доказать его корректность, снова воспользуемся разбором случаев: перебрав все варианты, в каком бите два натуральных числа различаются (за счёт чего сумма $< c$), заключим, что сумма $\leq c$; умножив на a, получим противоречие с посылкой правила. □

3.2 Оптимальные системы

Определение 3.1. Говорят, что A — *оптимальный полуалгоритм* (*optimal acceptor*) для языка L, если

- $\forall x \in L\ A(x) = 1$,
- $\forall x \notin L\ A(x) \neq 1$ (например, может зацикливаться),

- для всякого алгоритма A', удовлетворяющего первым двум условиям, найдётся полином p, такой что для всех $x \in L$

$$\mathrm{time}_A(x) \leq p(\mathrm{time}_{A'}(x) + |x|).$$

В отличие от оптимального аксептора, *оптимальный алгоритм* для языка должен всегда выдавать правильный ответ, а условие оптимальности должно выполняться для всех входов $x \in \{0,1\}^*$ без ограничений.

Факт 3.1. *Существует* оптимальный алгоритм для задачи поиска *выполняющего набора для булевой формулы (тут речь идёт, естественно, об алгоритмах, выдающих выполняющий набор, а не один бит; на невыполнимых формулах можно зацикливаться).*

Напомним доказательство. Этот алгоритм (оптимальный алгоритм Левина) работает следующим образом:

- параллельно запускаются все возможные алгоритмы;
- если какой-то алгоритм останавливается и выдает строку, то проверяем, является ли она выполняющим набором, и выдаем, если она подошла.

"Параллельность" достигается моделированием на "большом" шаге номер $k \cdot 2^l$ шага номер k алгоритма номер l. В терминах номеров ("больших") шагов замедление относительно каждого алгоритма определяется только порядковым номером этого алгоритма, а это константа. Общее замедление зависит от вычислительной модели, но остаётся полиномиальным даже для машин Тьюринга. Итак, с точностью до полинома, алгоритм Левина является самым быстрым. □

Замечание 3.1. Для языка TAUT алгоритм Левина не является оптимальным (на невыполнимых формулах он ничего путного не делает). Он не является оптимальным даже для SAT, если рассматривать её как задачу распознавания, а не задачу поиска, ведь чтобы "вытащить" выполняющий набор из очередного моделируемого алгоритма, его надо промоделировать не только на исходной формуле F, но и на формулах, полученных из неё подстановками в духе $F|_{x_1=1,\, x_2=0,\, x_3=1}$, а время его работы на какой-то их них может быть неожиданно гораздо больше, чем на F.

Теорема 3.2 (Krajíček, Pudlak, 1989)**.** *Существует p-оптимальная система доказательств* $\iff$ *существует оптимальный полуалгоритм для языка* TAUT *булевых тавтологий.*

Доказательство. $\Leftarrow$ Пусть существует оптимальный полуалгоритм O для TAUT.

Оптимальное доказательство формулы F размера n будет включать номер системы доказательств П, а также П-доказательство. Однако за отсутствием видимого способа перечислить только корректные системы доказательств, нам необходимо как-то подтвердить корректность П.

Оптимальный полуалгоритм O обязан выдавать ответ за полиномиальное время на любом подмножестве тавтологий из $\mathbf{P}$. Из доказательства теоремы Кука-Левина

ясно, что для любой системы доказательств П легко (за полиномиальное время) записывается тавтология

$$\mathrm{Con}_{\Pi,n}\colon \text{П корректна для формул размера } n$$

(поскольку некорректность системы проверяется полиномиальным вычислением, если в качестве подсказки даны формула и выполняющий набор), поэтому множество $C_\Pi = \{\mathrm{Con}_{\Pi,n}\}_{n\in\mathbb{N}}$, состоящее из таких тавтологий для всех возможных n, принадлежит $\mathbf{P}$.

Итак, полуалгоритм O полиномиален на C_Π.

Оптимальное доказательство формулы F размера n будет содержать

- номер системы доказательств П,
- протокол работы полуалгоритма O на $\mathrm{Con}_{\Pi,n}$ (точнее говоря, достаточно записать 1^t, где t — время работы полуалгоритма),
- П-доказательство π формулы F.

Система доказательств оптимальна (т.к. полуалгоритм O полиномиален на C_Π) и корректна. Доказательство конструктивно, поэтому система p-оптимальна.

$\Leftarrow$ Пусть система доказательств П является p-оптимальной.

Оптимальный полуалгоритм будет "параллельно" (см. алгоритм Левина) запускать все алгоритмы O_i, считая их претендующими на выдачу П-доказательств. Всякое выданное O_i «доказательство» проверяется П; если оно правильное, то выдается 1.

По определению p-оптимальности П протокол работы любого алгоритма A может быть за полиномиальное время преобразован в П-доказательство при помощи некоторого отображения f. Поскольку множество $\{O_i\}$ содержит композицию A и f, приведённый выше полуалгоритм действительно оптимален. □

Эту теорему можно обобщить, перейдя от TAUT к т. н. *дополняемым* языкам. Впрочем, более, чем общность, важен альтернативный метод доказательства, продемонстрированный в следующей теореме.

Определение 3.2. Язык L называется *дополняемым* (*paddable*), если имеется инъективная, не уменьшающая длину входов и полиномиальная по времени *дополняющая функция*

$$\mathrm{pad}_L\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*,$$

которая обратима за полиномиальное время на своем образе и не меняет принадлежности языку — то есть, для всех строк x и w справедливо

$$x \in L \iff \mathrm{pad}_L(x,w) \in L.$$

Первый аргумент функции pad_L выражает какую-то "формулу", а второй аргумент — это «приклеенная записка», которую потом можно отклеить.

Пример 3.1. Язык TAUT является дополняемым: $F \in \mathrm{TAUT} \iff F \vee (l_1 \wedge l_2 \ldots) \in \mathrm{TAUT}$, где $l_1, l_2, \ldots$ — новые переменные $x_1, x_2, \ldots$ или их отрицания.

Теорема 3.3 (Messner, 1999)**.** *Для всякого дополняемого языка L p-оптимальная система доказательств для L существует тогда и только тогда, когда существует оптимальный полуалгоритм для L.*

Доказательство. В одну сторону доказательство полностью повторяет доказательство Крайчека-Пудлака. Докажем существование p-оптимальной системы доказательств. Доказательство для неё будет содержать

- П-доказательство π для формулы F,
- 1^t — количество времени, которое можно потратить на проверку.

(удивительно, но описание системы доказательств П нам не потребуется).

Для проверки доказательства достаточно запустить оптимальный полуалгоритм на $\mathrm{pad}_L(x, \pi)$, ограничив время его работы t шагами (если не успел вернуть ответ, вернуть 0).

Для корректного доказательства π корректной системы П полуалгоритм принимает такой вход за полиномиальное время, поскольку

$$\left\{\mathrm{pad}_L(x, \pi) \;\middle|\; x \in L,\ \Pi(x, \pi) = 1\right\} \in \mathbf{P}$$

(полиномиальный алгоритм состоит в инвертировании функции pad_L и моделировании проверяющего алгоритма системы П). Эти полиномы для разных систем разные, но число t в доказательстве также можно выбрать в зависимости от системы. □