Базовые понятия теории формальных языков

Теория формальных языков

2023 г.

Формальные языки

Традиционный подход

Формальный язык — это множество $\mathcal{M}$ слов над алфавитом Σ (обозначается $\mathcal{M} \subseteq \Sigma^*$, здесь знак * — итерация Клини). Обычно подразумевает наличие формальных правил, определяющих корректность формы (т.е. синтаксиса) слов из $\mathcal{M}$.

Важная характеристика классического формального языка — тот факт, что он является подмножеством свободного моноида, как правило, конечнопорождённого (над конечным Σ).

Происхождение — входные языки вычислительных машин (лента). Сейчас существуют и формальные языки графовых и других неассоциативных структур.

Перечислимость и разрешимость

- Язык $\mathcal{M}$ разрешимый $\Leftrightarrow$ для любого слова w существует алгоритм проверки принадлежности w к $\mathcal{M}$ (всегда завершающийся и дающий точный, либо положительный, либо отрицательный ответ).
- Язык $\mathcal{M}$ перечислимый $\Leftrightarrow$ для любого слова w существует алгоритм, положительно отвечающий на вопрос принадлежности w к $\mathcal{M}$ за конечное время (но, возможно, зацикливающийся, если $w \notin \mathcal{M}$).

Перечислимый, но не разрешимый: язык программ, завершающихся на входе `0` (на любом достаточно мощном ЯП).

Далее разрешимые языки можно классифицировать по минимально необходимой сложности разрешающего алгоритма (P-разрешимые, $\mathsf{ExpTime}$-разрешимые...)

Примеры формальных языков

- $\{\underbrace{aa\ldots a}_{n \text{ раз}}\underbrace{bb\ldots b}_{n\cdot 3 \text{ раз}}\}$ (сокращаем до $\{a^n b^{3n}\}$);
- палиндромы чётной длины в русском языке;
- правильно записанные арифметические выражения с $\cdot$, $+$ над натуральными числами;
- правильные скобочные последовательности;
- язык тождественно истинных формул логики предикатов;
- язык правильно типизированных программ на ЯП со статическими типами;
- язык, описывающий все разрешимые за линейное время формальные языки.

Представления формальных языков

- Свёртки множеств
- Системы переписывания термов
- Распознающие / порождающие машины
- Алгебраические выражения
- Алгебраические структуры
- Формулы логики предикатов

Представление с помощью распознающих машин позволяет оценить вычислительную сложность формального языка (свести к универсальным автоматам — машинам Тьюринга).

Машины Тьюринга

- Потенциально не ограниченная лента (ограничения накладываются классом сложности по памяти);
- Доступные операции чтения (автоматически) и записи на ленту, а также сдвига пишущей головки влево или вправо;
- Конечное множество внутренних состояний машины и конечное число инструкций перехода.

Оценки времени и памяти для машин Тьюринга переносятся и на современные языки программирования с погрешностью $O(n^p)$, где n — длина входных данных. Для субполиномиальных алгоритмов МТ не так универсальны: (теорема Гуревича–Шелла) $n \log n$ алгоритмы эквивалентны во всех формализациях, кроме МТ!

Кодирующие алгоритмы

Погрешность затрат от длины входных данных почти одинакова, зачем потребовалось создавать много языков?

Ключевая проблема теории формальных языков — нахождение короткой (легко анализируемой, легко преобразуемой) кодировки алгоритма.

Минимальный детерминированный автомат-распознаватель слов с n-ой с конца буквой a имеет в $O(2^n)$ раз больше состояний, чем минимальный недетерминированный. При этом оба допускают линейные по длине входных данных алгоритмы разбора слова.

Примеры представления

Язык слов в алфавите $\{a, b\}$ с чётным количеством букв a.

- Свёртка: $\{w \in \{a, b\}^* \mid 2 \text{ делит } |w|_a\}$. $|w|_t$ — количество вхождений терма t в слово w.
- Система переписывания термов:

$$S \to "a" +\!+ T \quad S \to "b" +\!+ S \quad S \to \varepsilon$$
$$T \to "a" +\!+ S \quad T \to "b" +\!+ T$$

  А можно и по-другому:

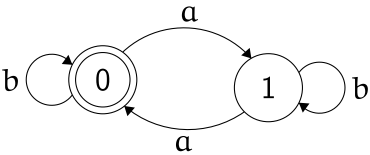
$$S \to "a" +\!+ S +\!+ "a" \quad S \to "b" +\!+ S$$
$$S \to S +\!+ "b" \quad S \to \varepsilon$$

  Здесь и далее ε — стандартное обозначения для пустого слова (строки нулевой длины). Поскольку структура данных — слова, то кавычки и знак конкатенации $+\!+$ дальше опускаются.

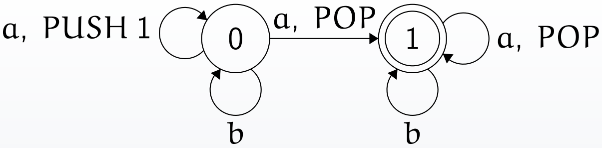
Примеры представления

Язык слов в алфавите $\{a, b\}$ с чётным количеством букв a.

- Распознающие машины:



А можно иначе:



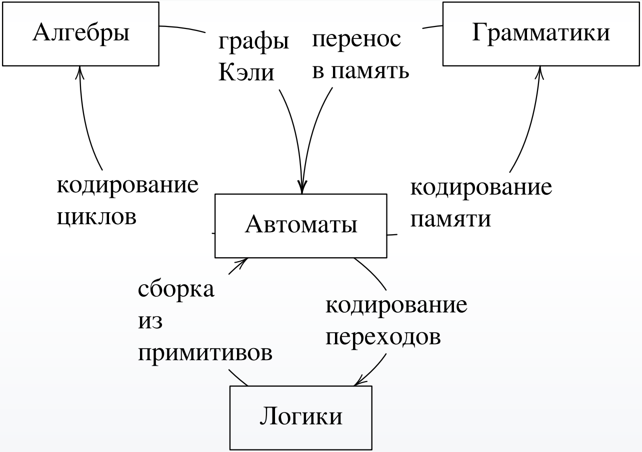
Примеры представления

Язык слов в алфавите $\{a, b\}$ с чётным количеством букв a.

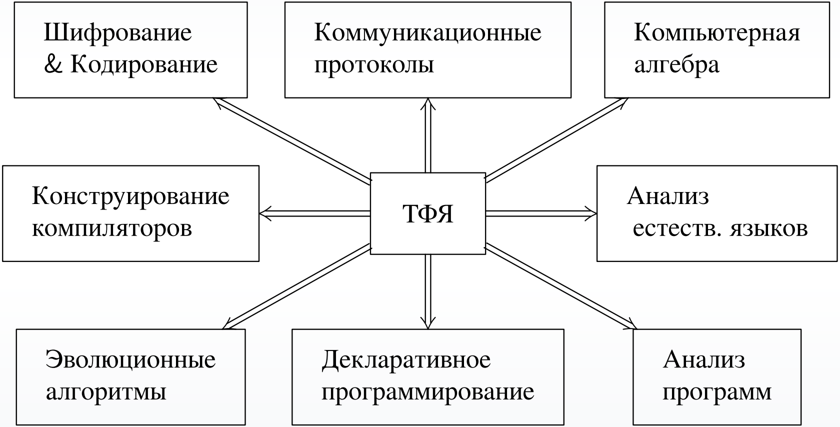
- Алгебраические выражения:
  $(b^*ab^*ab^*)^*$
  А можно и так:
  $(b^*ab^*a)^*b^*$
- Алгебраические структуры:
  Класс эквивалентности слова ε в полугруппе с соотношениями $aa \to \varepsilon$, $b \to \varepsilon$.
- Формулы логики предикатов: без введения считающих предикатов не выразима в логике предикатов первого порядка (но выразима в логике одноместных предикатов второго порядка).

Обзор представлений

- Поиск алгоритмов конвертации между классами языков
- Поиск оптимального представления внутри класса



Области применения



Структура курса

- Два рубежных контроля $\times$ 15 баллов.
- Пять лабораторных работ $\times$ 8 баллов.
  - Java, Python, Go, JS — без бонуса
  - C/C++, Kotlin, TypeScript, Lua — бонус +1 балл
  - Rust, Dart, все лиспы, Scala, Julia — бонус +2 балла
  - Haskell, Erlang — бонус +3 балла
  - Рефал — бонус +4 балла за первый раз и +3 за прочие
  - Agda, Idris (с доказательствами) — 5 баллов за курс
- С момента выдачи лабораторной работы:
  - 0-14 дней — сдача за полный балл
  - 15-21 день — сдача со штрафом -1 балл
  - 22-28 дней — сдача со штрафом -2 балла
  - 29-∞ — сдача со штрафом -3 балла

Система репутации

Все начинают с репутации 100 баллов (максимум).

- Сдача в последний день до смены штрафа: -2 балла.
- Сдача в последнюю ночь до смены штрафа: -5 баллов.
- Сдача в последнюю ночь до обнуления: -10 баллов.
- Очевидные артефакты чужого кода: -15 баллов.
- Неспособность объяснить свой код: -25 баллов.
- Сдача в последний день экзаменом: -35 баллов.
- Сдача в последнюю ночь перед экзаменом: -50 баллов.
- Сдача не своего варианта: -75 баллов.
- Прочие заслуги (нарушение порядка на биг-фарме, и т.п.): ситуативно.

Посылки проверяются в порядке убывания репутации.

Анализ свойств языков

Проверить, действительно ли данная система переписывания термов порождает язык $\{w \mid |w|_a$ делится на $2\}$, если начальным состоянием является S.

$S \to a\,S\,a \quad S \to b\,S \quad S \to S\,b \quad S \to \varepsilon$

- Необходимо доказать, что все указанные слова порождаются системой (например, по индукции).
- А также что никакие другие не порождаются.

Предположим, что система порождает слова с нечётным числом букв a. Выберем из них такое, которое выводится из S за самое малое число шагов. Покажем, что каким бы ни был предпоследний шаг вывода, его можно поменять на $S \to \varepsilon$ и получится слово с нечётным числом букв a, вывод которого ещё короче.

Проверка корректности рекурсивных доказательств

1. Завершаемость — индуктивные переходы должны завершаться за конечное число шагов. Соответствует фундированности множеств относительно выбранного упорядочения.

2. Корректность — способ доказательства «*minimal bad sequence*» — пусть существуют элементы $k_i \in M$, которые не находятся рекурсивным алгоритмом. Выберем тот из них, до которого минимальный путь из S (варианты — из которого минимальный путь до Σ^*; до ε). Покажем, что есть ещё какой-то с путём вывода ещё короче.

Системы переписывания термов

Определение

Сигнатура — множество пар $\langle f, n \rangle$ из имени конструктора f и его местности n.

Определение

Пусть V — множество переменных, F — множество конструкторов; множество термов $T(F)$ над F определяется рекурсивно:

- все элементы V — термы;
- если $\langle f, n \rangle$ — конструктор и $t_1, \ldots, t_n$ — термы, то $f(t_1, \ldots, t_n)$ — терм;
- других термов нет.

Term Rewriting Systems

Пусть V — множество переменных, F — множество конструкторов (сигнатура); $T(F)$ — множество термов над множеством конструкторов F. TRS — набор правил переписывания вида $\Phi_i \to \Psi_i$, где Φ_i, Ψ_i — термы в $T(F)$. Правило переписывания $\Phi_i \to \Psi_i$ применимо к терму t, если t содержит подтерм, который можно сопоставить (унифицировать) с Φ_i.

Если к терму t не применимо ни одно правило переписывания TRS, терм называется нормализованным.

Имея правила переписывания вида $f(g(x)) \to g(g(f(x)))$ и $g(g(x)) \to f(x)$, каждое из них можно применить к терму $f(g(g(f(g(f(g(g(g(Z)))))))))$ тремя разными способами.

Конфлюэнтность

Определение

TRS называется конфлюэнтной, если для любых двух термов t, s, которые получаются переписыванием одного и того же терма u, существует терм v такой, что t, s оба переписываются в v.
Формально:
$\forall u, t, s(u \to^* t \;\&\; u \to^* s \Rightarrow \exists v(t \to^* v \;\&\; s \to^* v))$

Конфлюэнтные системы поддаются распараллеливанию и легко оптимизируются.
$\to$ — переписывание за 1 шаг;
$\to^*$ — переписывание за произвольное число шагов, начиная с 0.

Особенности TRS

- Недетерминированные.
- Нет ограничений на порядок применения правил.
- Не обязательно конфлюэнтны.
- Могут порождать бесконечные цепочки.

Фундированность

Определение

Частичный порядок $\preceq$ является фундированным (wfo) на множестве M, если в M не существует бесконечных нисходящих цепочек относительно $\preceq$ (говоря о множестве термов, иногда такой $\preceq$ называют нётеровым).

Частичный порядок $\preceq$ является монотонным в алгебре A, если $\forall f, t_1, ..., t_n, s, s'(s \preceq s' \Rightarrow f(t_1, \dots, s, \dots, t_n) \preceq f(t_1, \dots, s', \dots, t_n))$ (строго монотонным, если при этом неверно обратное).

Завершаемость

Определение

Фундированная монотонная алгебра (ФуМА) над множеством функциональных символов F — это фундированное множество $\langle A, > \rangle$ такое, что для каждого функционального символа $f \in F$ существует функция $f_A : A^n \to A$, строго монотонная по каждому из аргументов.

Определим расширение произвольного отображения σ из множества переменных в A следующим образом:

- $[x, \sigma] = \sigma(x)$;
- $[f(t_1, \ldots, t_n), \sigma] = f_A([t_1, \sigma], \ldots, [t_n, \sigma])$.

Завершаемость

Совместность

TRS $\{l_i \to r_i\}$ совместна с ФуМА $A \Leftrightarrow$ для всех i и для всех σ выполняется условие $[l_i, \sigma] > [r_i, \sigma]$.

Теорема

TRS не порождает бесконечных вычислений (завершается), если и только если существует совместная с ней ФуМА.

ФуМА, совместные с TRS

Стандартные способы определения f_A:

- лексикографический порядок на множестве имён F + отношение подтерма;
- построение монотонно возрастающей (по каждому аргументу) числовой функции на $\mathbb{N}$, соответствующей f_A.

Оба случая подразумевают, что в построенной модели целое больше части, т.е. всегда выполняется $f(t) > t$.

Лексикографический порядок $> \mathrm{lo}$

Определение

$f(t_1, \ldots, t_n) >_{\mathrm{lo}} g(u_1, \ldots, u_m)$ (этот порядок также называют порядком Кнута–Бендикса) если и только если выполнено одно из условий:

1. $\exists i (1 \leqslant i \leqslant n \ \& \ t_i = g(u_1, \ldots, u_m))$;
2. $\exists i (1 \leqslant i \leqslant n \ \& \ t_i >_{\mathrm{lo}} g(u_1, \ldots, u_m))$;
3. $(f > g) \ \& \ \forall i (1 \leqslant i \leqslant m \Rightarrow f(t_1, \ldots, t_n) >_{\mathrm{lo}} u_i)$;
4. $(f = g) \ \& \ \forall i (1 \leqslant i \leqslant n \Rightarrow f(t_1, \ldots, t_n) >_{\mathrm{lo}} u_i)$ и n-ка $(t_1, \ldots, t_n)$ лексикографически больше, чем $(u_1, \ldots, u_n)$ (т.е. первый её не совпадающий с u_i элемент t_i удовлетворяет условию $t_i >_{\mathrm{lo}} u_i$).

Примеры

Проверить завершаемость TRS методом $>_{lo}$:

$$f(g(x)) \to g(h(x,x))$$
$$g(f(x)) \to h(g(x),x)$$

- Первое правило переписывания вынуждает либо $g(x) >_{lo} g(h(x,x))$ (по условию 1 или 2) — что невозможно, потому что x должно лексикографически оказаться больше $h(x,x)$ (по условию 4); либо $f > g$ и $f(g(x)) > h(x,x)$ (по условию 3). В этом случае можно взять также $f > h$. Неравенство $f(g(x)) > x$ выполняется тривиально.
- Второе правило переписывания удовлетворяет условию завершаемости по условию 2, например, если показать, что $f(x) >_{lo} h(g(x),x)$. Уже имеем $f > h$, поэтому достаточно показать $f(x) >_{lo} g(x)$ и $f(x) >_{lo} x$. Оба условия тривиально выполняются из допущений выше.

Примеры

Проверить завершаемость TRS методом построения монотонной функции:

$$f(g(x,y)) \to g(h(y),x)$$
$$h(f(x)) \to f(x).$$

- Завершаемость по второму правилу переписывания автоматически выполняется по свойству подтерма. Поэтому то, что функция f стоит на двух его сторонах, не дает никаких указаний относительно того, стоит ли делать f_A быстро растущей или медленно. Все подсказки содержатся только в первом правиле переписывания.
- По первому правилу переписывания видно, что f_A надо делать большой (f стоит только слева), а h_A нет (h есть только справа). Положим $f_A(x) = 10 \cdot (x+1)$, $h_A(x) = x+1$. Тогда должно выполняться $10 \cdot (g_A(x,y)+1) > g_A(y+1,x)$. Этому неравенству удовлетворяет, например, $g_A(x,y) = x+y$.

Общие комментарии

- Не обязательно добиваться выполнения неравенства на образах f_A на всём множестве $\mathbb{N}$. Поскольку любой отрезок $\mathbb{N}$ от k и до бесконечности фундирован, а все образы f_A монотонны, они замкнуты на этом отрезке. Поэтому, если неравенство не выполняется для нескольких первых чисел натурального ряда, этим можно пренебречь.

- Если не получается применить $>_{lo}$ или подобрать числовую функцию, это ещё не значит, что TRS не завершается. См. пример Зантемы: $f(g(x)) \to g(f(f(x)))$.

Терминалы и нетерминалы

Если TRS определена над алфавитом Σ, а нас интересует порождаемый ею язык в $\Sigma' \subset \Sigma$, то элементы Σ' обычно называются терминалами, а элементы $\Sigma \setminus \Sigma'$ — нетерминалами.

В этом случае значащие (порождающие) нетерминалы обязательно должны встречаться хотя бы в одной левой части правила переписывания (иначе такой нетерминал не сможет быть переписан в слово над Σ').

Терминалы также могут встречаться в левых частях правил (это не так только для некоторых классов систем переписывания термов).

Грамматики

Определение

Грамматика — это четвёрка $G = \langle N, \Sigma, P, S \rangle$, где:

- N — алфавит нетерминалов;
- Σ — алфавит терминалов;
- P — множество правил переписывания $\alpha \to \beta$ типа $\langle (N \cup \Sigma)^+ \times (N \cup \Sigma)^* \rangle$;
- $S \in N$ — начальный символ.

$\alpha \Rightarrow \beta$, если $\alpha = \gamma_1 \alpha' \gamma_2$, $\beta = \gamma_1 \beta' \gamma_2$, и $\alpha' \to \beta' \in P$.
$\Rightarrow^*$ — рефлексивное транзитивное замыкание $\Rightarrow$.

Определение

Язык $L(G)$, порождаемый G — множество $\{u \,|\, u \in \Sigma^* \ \&\ S \Rightarrow^* u\}$.

α-преобразование

По-разному воспринимают переименовку:

- Переменные vs конструкторы в TRS;
- Нетерминалы vs терминалы в грамматиках.

α-преобразование

По-разному воспринимают переименовку:

- Переменные vs конструкторы в TRS;
- Нетерминалы vs терминалы в грамматиках.

Для любой инъективной переименовки σ применение σ к правилам грамматики/trs для переменных и нетерминалов также называется α-преобразованием.

- α-преобразование не меняет терминальный язык;
- обычно термы различаются с точностью до α-преобразования.

α-преобразование

По-разному воспринимают переименовку:

- Переменные vs конструкторы в TRS;
- Нетерминалы vs терминалы в грамматиках.

Для любой инъективной переименовки σ применение σ к правилам грамматики/trs для переменных и нетерминалов также называется α-преобразованием.

- α-преобразование не меняет терминальный язык;
- обычно термы различаются с точностью до α-преобразования.

Неформально: контейнеры определяются не именем, а содержимым (см. экстенсиональность в логике).