

Базовые понятия теории формальных языков



Теория формальных языков
2022 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} .

Теоретико-категорный подход

Формальный язык — это категория (т.е. способ задания объектов и их взаимосвязей в форме направленного графа). Отличие от теоретико-множественного подхода — способ описания не синтаксиса слов, а отношения между ними.

Классический пример: A man saw a dog with a telescope.



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, закливающийся, 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, ExpTime-разрешимые...)



Примеры формальных языков

- $\{\underbrace{aa\dots a}_{n \text{ раз}} \underbrace{bb\dots b}_{n \cdot 3 \text{ раз}}\}$ (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



Пример 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:
 $S \rightarrow "a" ++ T \quad S \rightarrow "b" ++ S \quad S \rightarrow \varepsilon$
 $T \rightarrow "a" ++ S \quad T \rightarrow "b" ++ T$

А можно и по-другому:

$$\begin{aligned} S &\rightarrow "a" ++ S ++ "a" & S &\rightarrow "b" ++ S \\ S &\rightarrow S ++ "b" & S &\rightarrow \varepsilon \end{aligned}$$

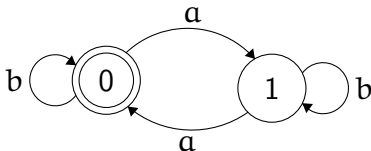
Здесь и далее ε — стандартное обозначения для пустого слова (строки нулевой длины). Поскольку структура данных — слова, то кавычки и знак конкатенации $++$ дальше опускаются.



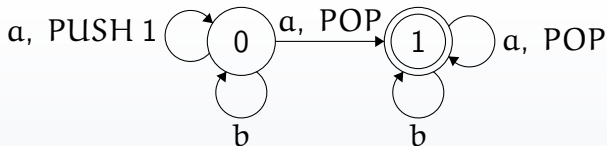
Пример представления

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

- Распознающие машины:



А можно иначе:





Пример 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 $aa \rightarrow \varepsilon$, $b \rightarrow \varepsilon$.

- Формулы логики предикатов: без введения считающих предикатов не выразима в логике предикатов первого порядка (но выразима в логике одноместных предикатов второго порядка).



Анализ свойств языков

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

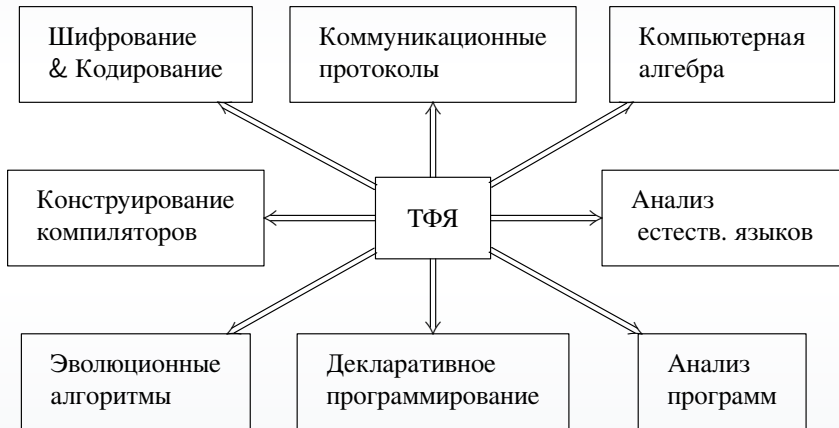
$$S \rightarrow a S a \quad S \rightarrow b S \quad S \rightarrow S b \quad S \rightarrow \varepsilon$$

- 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 $S \rightarrow \varepsilon$ и получится слово с нечётным числом букв a , вывод которого ещё короче.



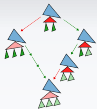
Области применения





Структура курса

- Два рубежных контроля \times 15 баллов.
- Пять лабораторных работ \times 8 баллов.
 - Java, Python, Go, JS — без бонуса
 - C/C++, Kotlin, TypeScript — бонус +1 балл
 - Rust, Dart, все лиспы, Scala — бонус +2 балла
 - Lua, Haskell, Erlang, Рефал — бонус +3 балла
 - 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



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, \dots, t_n — термы, то $f(t_1, \dots, t_n)$ — терм;
- других термов нет.

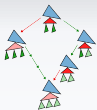


Term Rewriting Systems

Пусть V — множество переменных, F — множество конструкторов (сигнатура); $T(F)$ — множество термов над множеством конструкторов F . TRS — набор правил переписывания вида $\Phi_i \rightarrow \Psi_i$, где Φ_i, Ψ_i — термы в $T(F)$. Правило переписывания $\Phi_i \rightarrow \Psi_i$ 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 $f(g(x)) \rightarrow g(g(f(x)))$ и $g(g(x)) \rightarrow f(x)$, 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 .

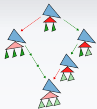
Формально:

$$\forall u, t, s (u \rightarrow^* t \ \& \ u \rightarrow^* s \Rightarrow \exists v (t \rightarrow^* v \ \& \ s \rightarrow^* v))$$

Конфлюэнтные системы поддаются распараллеливанию и легко оптимизируются.

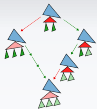
\rightarrow — переписывание за 1 шаг;

\rightarrow^* — переписывание за произвольное число шагов, начиная с 0.



Особенности TRS

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



Особенности TRS

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

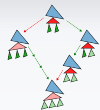
Пример Хетта

$$f(x, x) \rightarrow a$$

$$f(x, g(x)) \rightarrow b$$

$$c \rightarrow g(c)$$

Терм, где нарушается конфлюэнтность?



Особенности TRS

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

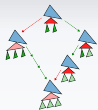
Пример Клопа

$$A \rightarrow CA$$

$$Cz \rightarrow Dz(Cz)$$

$$Dzz \rightarrow E$$

Способы преобразовать A?



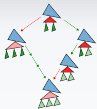
Особенности TRS

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

Пример Тойямы

- TRS 1:
 $f(0, 1, x) \rightarrow f(x, x, x)$
- TRS 2:
 $g(x, y) \rightarrow x$
 $g(x, y) \rightarrow y$

Как можно вычислить $f(g(0, 1), g(0, 1), g(0, 1))$?



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 $\forall f, t_1, \dots, t_n, s, s' (s \preceq s' \Rightarrow f(t_1, \dots, s, \dots, t_n) \preceq f(t_1, \dots, s', \dots, t_n))$ (строго монотонным, если при этом неверно обратное).



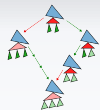
Завершаемость

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 $f_A : A^n \rightarrow A$, строго монотонная по каждому из аргументов.

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

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



Завершаемость

Совместность

TRS $\{l_i \rightarrow r_i\}$ 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, соответствующей f_A .

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



Лексикографический порядок $>_{lo}$

Определение

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

- ❶ $\exists i(1 \leq i \leq n \ \& \ t_i = g(u_1, \dots, u_m))$;
- ❷ $\exists i(1 \leq i \leq n \ \& \ t_i >_{lo} g(u_1, \dots, u_m))$;
- ❸ $(f > g) \ \& \ \forall i(1 \leq i \leq m \Rightarrow f(t_1, \dots, t_n) >_{lo} u_i)$;
- ❹ $(f = g) \ \& \ \forall i(1 \leq i \leq n \Rightarrow f(t_1, \dots, t_n) >_{lo} u_i)$ и n -ка (t_1, \dots, t_n) лексикографически больше, чем (u_1, \dots, u_n) (т.е. первый её не совпадающий с u_i элемент t_i удовлетворяет условию $t_i >_{lo} u_i$).

Примеры

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

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

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

- 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.

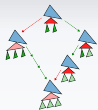
Примеры

Проверить завершаемость TRS методом построения монотонной функции:

$$f(g(x, y)) \rightarrow g(h(y), x)$$

$$h(f(x)) \rightarrow f(x).$$

- 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 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 $>_{l_0}$ или подобрать числовую функцию, это ещё не значит, что TRS не завершается. См. пример Зантемы:
 $f(g(x)) \rightarrow g(f(f(x)))$.



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 $\alpha \rightarrow \beta$ 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 $\alpha' \rightarrow \beta' \in P$.

\Rightarrow^* — рефлексивное транзитивное замыкание \Rightarrow .

Определение

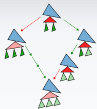
Язык $L(G)$, порождаемый G — множество $\{u \mid u \in \Sigma^* \text{ \& } S \Rightarrow^* u\}$.



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

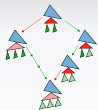
По-разному воспринимают переименовку:

- 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. экстенциональность в логике).



Иерархия Хомского без ε -правил

$A, B \in N, a \in \Sigma^*, \alpha, \beta \in (N \cup \Sigma)^*, \gamma \in (N \cup \Sigma)^+$

Иерархия грамматик

Тип 0	Рекурсивно-перечислимые	\forall
Тип 1	Контекстно-зависимые	$\alpha A \beta \rightarrow \alpha \gamma \beta, \gamma \neq \varepsilon$
Тип 2	Контекстно-свободные	$A \rightarrow \gamma$
Тип 3	Праволинейные (регулярные)	$A \rightarrow a, A \rightarrow aB$



Иерархия Хомского без ε -правил

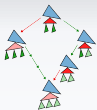
$A, B \in N, a \in \Sigma^*, \alpha, \beta \in (N \cup \Sigma)^*, \gamma \in (N \cup \Sigma)^+$

Иерархия грамматик

Тип 0	Рекурсивно-перечислимые	\forall
Тип 1	Контекстно-зависимые	$\alpha A \beta \rightarrow \alpha \gamma \beta, \gamma \neq \varepsilon$
Тип 2	Контекстно-свободные	$A \rightarrow \gamma$
Тип 3	Праволинейные (регулярные)	$A \rightarrow a, A \rightarrow aB$

Примеры языков

Тип 0	$\{u \mid L(u) = L(r)\}, r \text{ — фикс. regex, } u \text{ — regex;}$
Тип 1	$\{ww \mid w \in \Sigma^+\}$
Тип 2	непустые палиндромы в алфавите $\{a, b\}$
Тип 3	$\{w \mid w = aw_1 \ \& \ (w_1 = a^{2k} \vee w = a^{3k} \vee w_1 \neq a^{5k})\}$



Иерархия Хомского с ε -правилами

$A, B \in N, a \in \Sigma^*, \alpha, \beta \in (N \cup \Sigma)^*, \gamma \in (N \cup \Sigma)^+.$

Иерархия грамматик

Тип 0	Рекурсивно-перечислимые	\forall
Тип 1	Контекстно-зависимые	$\alpha A \beta \rightarrow \alpha \gamma \beta, \gamma \neq \varepsilon$ $\forall S \rightarrow \varepsilon \ \& \ \forall p : \alpha \rightarrow \beta \in P \forall \beta_1, \beta_2 (\beta \neq \beta_1 S \beta_2)$
Тип 2	Контекстно-свободные	$A \rightarrow \alpha$
Тип 3	Регулярные	$A \rightarrow a, A \rightarrow aB, A \rightarrow \varepsilon$