Лабораторная работа №1

Вариант 14

$$\begin{cases} ac \to c \\ aa \to ba \\ bc \to c \\ ca \to aa \\ cb \to bc \\ ccc \to c \\ aaa \to aa \\ aab \to ba \\ abb \to aba \\ bbb \to bba \end{cases}$$

Первичный анализ

Слова в данной SRS образованы из алфавита $\{a, b, c\}$. Кроме того, все правила этой системы не увеличивают длину строки. Поэтому можно переписать систему так, чтобы сначала шли правила, уменьшающие длину слова, а затем правила, сохраняющие её. Получаем:

$$\begin{cases} ac \to c & (1) \\ bc \to c & (2) \\ ccc \to c & (3) \\ aaa \to aa & (4) \\ aab \to ba & (5) \\ aa \to ba & (6) \\ ca \to aa & (7) \\ cb \to bc & (8) \\ abb \to aba & (9) \\ bbb \to bba & (10) \end{cases}$$

Пронумеруем каждое преобразование для удобства. Также можно отметить, что все преобразования, кроме (6), образуют фундированную последовательность при сравнении слов сначала по длине, а затем лексикографически (ShortLex). Это может быть полезно в дальнейшем.

Завершимость

Данная система переписывания не является завершимой. Можно найти цикл:

$$bbaaa \xrightarrow{(6)} bbbaa \xrightarrow{(10)} bbaaa$$

Конечность классов эквивалентности по НФ

Система имеет бесконечное число классов эквивалентности по нормальной форме. Рассмотрим слова вида $(ab)^n$, где $n \in \mathbb{N}$. Ни одно из этих слов не поддаётся преобразованию с помощью данной системы, а таких слов бесконечно много. Следовательно, нормальных форм бесконечно много.

Локальная конфлюэнтность

Рассмотрим слово *abbbc*. Применив к нему отдельно правила (9) и (10), получаем слова *ababc* и *abbac*. К слову *ababc* можно применить только правило (2), которое даёт слово *abac* длины 4. Следовательно, для слова *abbac* нужно применить правило, уменьшающее его длину на 1. Для этого подходит только правило (1), которое даёт слово *abbc*. Так как $abac \neq abbc$, SRS не локально конфлюэнтна.

Пополняемость по Кнуту–Бендиксу

Для того чтобы начать применять алгоритм пополнения Кнута–Бендикса, нужно, чтобы система имела терминированный порядок. Но если бы он был, то система переписывания была бы завершаемой, чего быть не может, так как у системы есть цикл. Поэтому данная SRS не пополняема по Кнуту–Бендиксу. Однако, если изменить правило (6) на $ba \to aa$, то будет выполняться ShortLex.

Пополнение

Для проверки пополняемости системы рассмотрим все критические пары, возникающие при наложении левых частей правил. Всего таких пар 34:

ccc–ccc(x2), aaa–aaa(x2), bbb–bbb(x2), ac–ccc, ac–aaa, ac–ba, ac–ca(x2), ac–cb,

bc–ccc, bc–aab, bc–ca, bc–cb, bc–abb, bc–bbb, ccc–ca, ccc–cb, aaa–aab(x2),

aaa–ba, aaa–ca, aaa–abb, aab–ba, aab–ca, aab–bbb, bba–cb, ba–abb(x2), ba–bbb,

ca–abb, cb–bbb.

Пары $ca - ac$ и $cb - bbb$ порождают слова *cac* и *cbbb* нарушающие конфлюэнтность системы, поскольку каждое из них может быть редуцировано более чем одним способом, приводящим к различным нормальным формам: *cac* может переходить в *c* или *cc*, а *cbbb* в *c* или *aa*.

Для устранения проблемы добавим правило $cc \to c$. Его введение создаёт новые критические пары:

cc–cc, *ac–cc*, *bc–cc*, *ccc–cc*, *cc–ca*, *cc–cb*,

все из которых сводятся к нормальной форме *c* или *aa*.

После этого добавляем правило $aa \to c$, чтобы устранить оставшиеся неоднозначности. При этом появляются новые пары:

aa–aa, *aa–ac*, *aa–aaa*, *aa–aab*, *ba–aa*, *ca–aa*, *aa–abb*,

которые также сводятся к нормальной форме *c*.

В результате получаем пополненную систему переписываний, которая является завершаемой и конфлюэнтной:

$$\begin{cases} \mathbf{cc} \to \mathbf{c} & (1) \\ \mathbf{aa} \to \mathbf{c} & (2) \\ ac \to c & (3) \\ bc \to c & (4) \\ ccc \to c & (5) \\ aaa \to aa & (6) \\ aab \to ba & (7) \\ ba \to aa & (8) \\ ca \to aa & (9) \\ cb \to bc & (10) \\ abb \to aba & (11) \\ bbb \to bba & (12) \end{cases}$$

Нормальные формы

Так как все слова длины 3 редуцируются, то к любым словам большей длины тоже применима полученная система. Следовательно, множество нормальных форм конечно.

У полученной системы количество нормальных форм сократилось по сравнению с исходной:

– в первой системе нормальные формы: $\{\varepsilon, a, b, c, aa, ab, bb, cc\}$;
– во второй системе: $\{\varepsilon, a, b, c, ab, bb\}$.

Минимизация

Цель минимизации системы переписывания заключается в том, чтобы уменьшить длину слов и количество правил в системе, не изменяя при этом множество достижимых нормальных форм.

Правило (5) можно удалить, так как его действие полностью воспроизводится с помощью двух последовательных применений правила (1).

Правило (6) также является избыточным, поскольку результат его применения можно получить при помощи уже существующих правил: aaa $\xrightarrow{(2)} ca \xrightarrow{(9)} aa$.

Если для нас важно сохранить все промежуточные слова, возникающие в процессе применения правил переписывания, то на этом этапе минимизацию можно считать завершённой, поскольку каждое оставшееся правило выполняет уникальную функцию и не может быть выражено через остальные.

Однако, если целью является лишь получение конечного результата, то есть нормальной формы слова, то систему можно упростить ещё больше.

Поскольку все слова длины 2, за исключением ab и bb, являющихся нормальными формами, редуцируются к c, а также все слова длины 3 в итоге тоже переходят в c, при этом длина слова на каждом шаге не уменьшается более чем на единицу, систему переписывания можно минимизировать сильнее:

$$\begin{cases} cc \to c \\ aa \to c \\ ac \to c \\ bc \to c \\ ba \to c \\ ca \to c \\ cb \to c \\ abb \to c \\ bbb \to c \end{cases}$$

В такой системе все слова, не являющиеся изначально нормальными формами, непосредственно переходят в c, что полностью соответствует свойству итогого результата алгоритма Кнута–Бендикса.

Инварианты

1. Количество символов b не возрастает.

2. Длина строки не увеличивается.

3. Слово всегда убывает по правилу ShortLex.

4. Суммарное расстояние символов b до начала строки не увеличивается.

5. Слово либо уже является нормальной формой, либо после применения любого правила переписывания в нём всегда остаётся хотя бы один символ a или c.

6. Если в слове отсутствует символ b, то в процессе преобразований символ b никогда не появится.

7. Пусть дана функция
$$f(w) = |w|_a + |w|_b + 2 \cdot |w|_c,$$
где $|w|_x$ обозначает количество символов x в слове w. Тогда при любом применении правила значение функции $f(w)$ не увеличивается.

7′. Пусть дана функция
$$f(w) = 2^{|w|_a + |w|_b} * 3^{|w|_c}.$$
Значение этой функции также не увеличивается при применении любого правила переписывания.

8. Сопоставим каждому символу матрицу:
$$a = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}, \quad c = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$
Обозначим матрицу, соответствующую левой части правила, через L, а матрицу, соответствующую правой части через R. Для каждого преобразования выполняется поэлементное неравенство $L \geq R$. Следовательно, при замене подстроки, соответствующей матрице L, на подстроку, соответствующую матрице $L - R$, получаем

выражение, являющееся матрицей без отрицательных элементов, поскольку все элементы матрицы $L - R$ неотрицательны.

Таким образом, применение такого преобразования не может привести к появлению отрицательных элементов в произведении всех матриц строки после преобразования.

Для следующего преобразования все символы становятся какими и были до этого.

Итог

$$\boldsymbol{SRS\tau}: \begin{cases} ac \to c \\ bc \to c \\ ccc \to c \\ aaa \to aa \\ aab \to ba \\ aa \to ba \\ ca \to aa \\ cb \to bc \\ abb \to aba \\ bbb \to bba \end{cases}$$

$$\boldsymbol{SRS\tau'}: \begin{cases} cc \to c \\ aa \to c \\ ac \to c \\ bc \to c \\ aab \to ba \\ ba \to aa \\ ca \to aa \\ cb \to bc \\ abb \to aba \\ bbb \to bba \end{cases}$$