Удлиняющие грамматики и контекстные вставки/удаления

Тихон Пшеницын

Математический институт им. В. А. Стеклова Российской академии наук

25.6.25

Введение

Определение

Формальная грамматика — структура вида $G = \langle N, T, P, S \rangle$, где

- N — нетерминальные символы, T — терминальные символы,
- P — набор правил вида $\alpha \to \beta$, где α, β — произвольные строки,
- S — стартовый нетерминальный символ.

Применение правила: $x\alpha y \Rightarrow_G x\beta y$.
Язык грамматики: $L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$.

Введение

Определение

Формальная грамматика — структура вида $G = \langle N, T, P, S \rangle$, где

- N — нетерминальные символы, T — терминальные символы,
- P — набор правил вида $\alpha \to \beta$, где α, β — произвольные строки,
- S — стартовый нетерминальный символ.

Применение правила: $x\alpha y \Rightarrow_G x\beta y$.
Язык грамматики: $L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$.

- Произвольными формальными грамматиками можно задать любой перечислимый язык — слишком много.

Введение

Определение

Формальная грамматика — *структура вида* $G = \langle N, T, P, S \rangle$, *где*

- N — *нетерминальные символы,* T — *терминальные символы,*
- P — *набор правил вида* $\alpha \to \beta$, *где* α, β — *произвольные строки,*
- S — *стартовый нетерминальный символ.*

Применение правила: $x\alpha y \Rightarrow_G x\beta y$.
Язык грамматики: $L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$.

- Произвольными формальными грамматиками можно задать любой перечислимый язык — слишком много.
- Контекстно-свободные грамматики: правила вида $A \to \alpha$, где $A \in N$.

Введение

Определение

Формальная грамматика — структура вида $G = \langle N, T, P, S \rangle$, где

- N — нетерминальные символы, T — терминальные символы,
- P — набор правил вида $\alpha \to \beta$, где α, β — произвольные строки,
- S — стартовый нетерминальный символ.

Применение правила: $x\alpha y \Rightarrow_G x\beta y$.
Язык грамматики: $L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$.

- Произвольными формальными грамматиками можно задать любой перечислимый язык — слишком много.
- Контекстно-свободные грамматики: правила вида $A \to \alpha$, где $A \in N$. Алгоритм Кока-Янгера-Касами за полиномиальное время проверяет по контекстно-свободной грамматике G и по слову $w = w_1 \dots w_n$, принадлежит ли w языку $L(G)$.

Введение

Определение

Формальная грамматика — *структура вида* $G = \langle N, T, P, S \rangle$, *где*

- N — *нетерминальные символы,* T — *терминальные символы,*
- P — *набор правил вида* $\alpha \to \beta$, *где* α, β — *произвольные строки,*
- S — *стартовый нетерминальный символ.*

Применение правила: $x\alpha y \Rightarrow_G x\beta y$.
Язык грамматики: $L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$.

- Произвольными формальными грамматиками можно задать любой перечислимый язык — слишком много.
- Контекстно-свободные грамматики: правила вида $A \to \alpha$, где $A \in N$. Алгоритм Кока-Янгера-Касами за полиномиальное время проверяет по контекстно-свободной грамматике G и по слову $w = w_1 \dots w_n$, принадлежит ли w языку $L(G)$.
  Идея — динамическое программирование: храним в памяти тройки (A, i, j) такие, что $A \Rightarrow_G^* w_i \dots w_j$.

Введение-2

- Неукорачивающие грамматики: правила вида $\alpha \to \beta$, где β *не короче, чем* α.

Введение-2

- Неукорачивающие грамматики: правила вида $\alpha \to \beta$, где β *не короче, чем* α.
  - Задают тот же класс языков, что и недетерминированные машины Тьюринга с линейным ограничением на память (NLINSPACE), ...

Введение-2

- Неукорачивающие грамматики: правила вида $\alpha \to \beta$, где β *не короче, чем* α.
  - Задают тот же класс языков, что и недетерминированные машины Тьюринга с линейным ограничением на память (NLINSPACE), ...
  - в том числе PSPACE-полные языки: если L — произвольный PSPACE-язык, распознаваемый машиной Тьюринга с объемом памяти $p(n)$, то $\{w\#^{p(n)} \mid w \in L\}$ распознается машиной с линейной памятью ($\#$ — новый символ).

Введение-2

- Неукорачивающие грамматики: правила вида $\alpha \to \beta$, где β *не короче, чем* α.
  - Задают тот же класс языков, что и недетерминированные машины Тьюринга с линейным ограничением на память (NLINSPACE), ...
  - в том числе PSPACE-полные языки: если L — произвольный PSPACE-язык, распознаваемый машиной Тьюринга с объемом памяти $p(n)$, то $\{w\#^{p(n)} \mid w \in L\}$ распознается машиной с линейной памятью ($\#$ — новый символ).
- Промежуточный класс — удлиняющие грамматики (Dahlhaus, Warmuth 1986): правила вида $\alpha \to \beta$, где β *строго длиннее* α.

Введение-2

- Неукорачивающие грамматики: правила вида $\alpha \to \beta$, где β *не короче, чем* α.
  - Задают тот же класс языков, что и недетерминированные машины Тьюринга с линейным ограничением на память (NLINSPACE), ...
  - в том числе PSPACE-полные языки: если L — произвольный PSPACE-язык, распознаваемый машиной Тьюринга с объемом памяти $p(n)$, то $\{w\#^{p(n)} \mid w \in L\}$ распознается машиной с линейной памятью ($\#$ — новый символ).
- Промежуточный класс — удлиняющие грамматики (Dahlhaus, Warmuth 1986): правила вида $\alpha \to \beta$, где β *строго длиннее* α.
  - Каждое правило увеличивает длину слова, поэтому если $S \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n = w$, то $n < |w|$. Задача выводимости для удлиняющих грамматик принадлежит NP.

Введение-2

- Неукорачивающие грамматики: правила вида $\alpha \to \beta$, где β *не короче, чем* α.
  - Задают тот же класс языков, что и недетерминированные машины Тьюринга с линейным ограничением на память (NLINSPACE), ...
  - в том числе PSPACE-полные языки: если L — произвольный PSPACE-язык, распознаваемый машиной Тьюринга с объемом памяти $p(n)$, то $\{w\#^{p(n)} \mid w \in L\}$ распознается машиной с линейной памятью ($\#$ — новый символ).
- Промежуточный класс — удлиняющие грамматики (Dahlhaus, Warmuth 1986): правила вида $\alpha \to \beta$, где β *строго длиннее* α.
  - Каждое правило увеличивает длину слова, поэтому если $S \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n = w$, то $n < |w|$. Задача выводимости для удлиняющих грамматик принадлежит NP.
  - Удлиняющие грамматики задают как минимум все контекстно-свободные языки со словами длины не менее двух (достаточно устранить правила вида $A \to \varepsilon$, $A \to B$ и $A \to a$).

Введение-2

- Неукорачивающие грамматики: правила вида $\alpha \to \beta$, где β *не короче, чем* α.
  - Задают тот же класс языков, что и недетерминированные машины Тьюринга с линейным ограничением на память (NLINSPACE), ...
  - в том числе PSPACE-полные языки: если L — произвольный PSPACE-язык, распознаваемый машиной Тьюринга с объемом памяти $p(n)$, то $\{w\#^{p(n)} \mid w \in L\}$ распознается машиной с линейной памятью ($\#$ — новый символ).
- Промежуточный класс — удлиняющие грамматики (Dahlhaus, Warmuth 1986): правила вида $\alpha \to \beta$, где β *строго длиннее* α.
  - Каждое правило увеличивает длину слова, поэтому если $S \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n = w$, то $n < |w|$. Задача выводимости для удлиняющих грамматик принадлежит NP.
  - Удлиняющие грамматики задают как минимум все контекстно-свободные языки со словами длины не менее двух (достаточно устранить правила вида $A \to \varepsilon$, $A \to B$ и $A \to a$).
  - Что насчёт неконтекстно-свободных языков? NP-полных языков?

Два примера

Пример

Язык $\{a^n b^n c^n \mid n \geq 1\}$ *задается удлиняющей грамматикой. Набросок:*

1. $S \to X_0 S$, $S \to X_1 S$, $S \to abcD$ — *генерируют строки вида* $X_{t_n} \dots X_{t_1} abcD$.
2. *Символ* X_0 *"бежит" слева направо и удваивает каждый терминальный символ.* X_1 *делает то же самое, но дополнительно в конце увеличивает количество каждого из символов на 1.*
   - $X_0 a \to aaX_0$, $X_0 b \to bbX_0$, $X_0 c \to ccX_0$, $X_0 ccD \to ccccD$.
   - $X_1 aa \to aaX_1 a$, $X_1 ab \to aaaX_1 b$, $\dots$
3. *Нужно аккуратно удалять* X_i *удлиняющими правилами, а также не менее аккуратно удалить* D *в конце.*

Два примера

Пример

Язык $\{a^n b^n c^n \mid n \geq 1\}$ *задается удлиняющей грамматикой. Набросок:*

1. $S \to X_0 S$, $S \to X_1 S$, $S \to abcD$ — *генерируют строки вида* $X_{t_n} \dots X_{t_1} abcD$.
2. *Символ* X_0 *"бежит" слева направо и удваивает каждый терминальный символ.* X_1 *делает то же самое, но дополнительно в конце увеличивает количество каждого из символов на 1.*
   - $X_0 a \to aaX_0$, $X_0 b \to bbX_0$, $X_0 c \to ccX_0$, $X_0 ccD \to ccccD$.
   - $X_1 aa \to aaX_1 a$, $X_1 ab \to aaaX_1 b$, ...
3. *Нужно аккуратно удалять* X_i *удлиняющими правилами, а также не менее аккуратно удалить* D *в конце.*

Пример

Язык $\{wcw^R cw \mid w \in \{a, b\}^*\}$ *не задается удлиняющей грамматикой (следует из результата А.В. Гладкого 1964 года, "Алгебра и логика").*

Неравномерная задача выводимости

Теорема

Для каждой фиксированной удлиняющей грамматики G существует полиномиальный алгоритм решения задачи принадлежности. Иными словами, языки удлиняющих грамматик принадлежат P.

Неравномерная задача выводимости

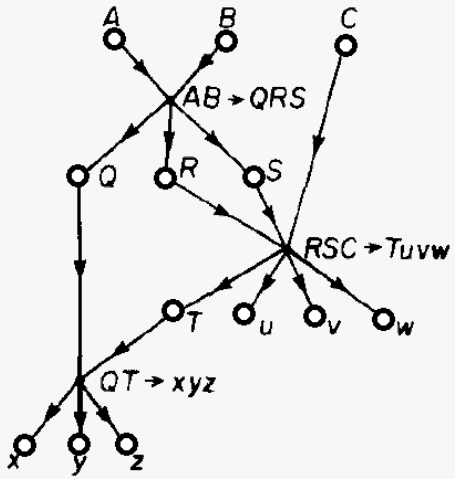
Теорема

Для каждой фиксированной удлиняющей грамматики G существует полиномиальный алгоритм решения задачи принадлежности. Иными словами, языки удлиняющих грамматик принадлежат P.

Граф вывода

Правила: $AB \to QRS$, $RSC \to Tuvw$, $QT \to xyz$.

$$ABC \Rightarrow QRSC \Rightarrow QTuvw \Rightarrow xyzuvw$$



Короткие пути в графе вывода

Коэффициент роста грамматики: $g := \min\{|\beta|/|\alpha| \mid (\alpha \to \beta) \in P\} > 1$.

Лемма

Из каждой вершины-символа v в графе вывода слова w существует путь к некоторому стоку (вершине, из которой не выходят ребра) длины не более $2\log_g(|w|)$.

Короткие пути в графе вывода

Коэффициент роста грамматики: $g := \min\{|\beta|/|\alpha| \mid (\alpha \to \beta) \in P\} > 1$.

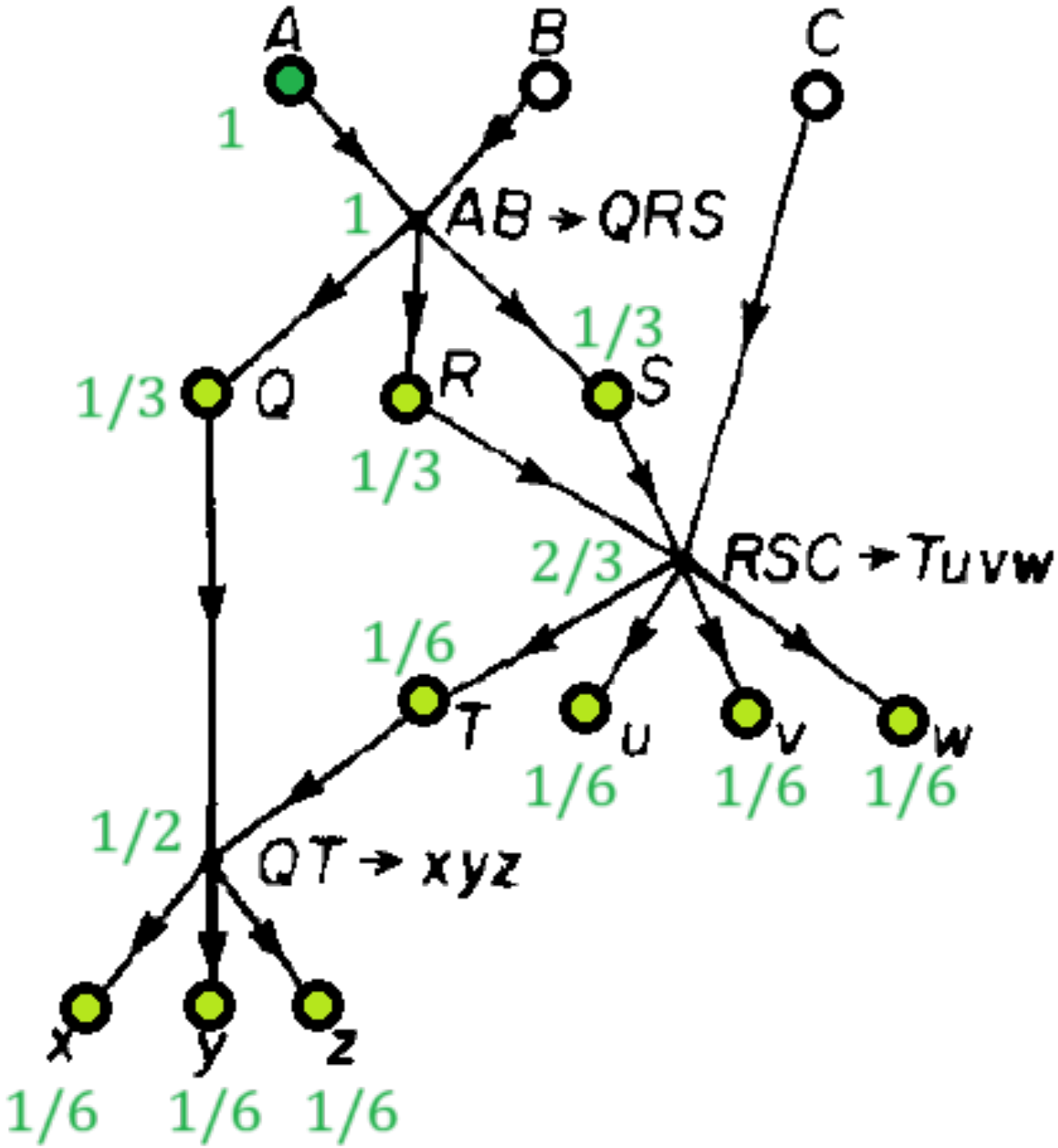
Лемма

Из каждой вершины-символа v в графе вывода слова w существует путь к некоторому стоку (вершине, из которой не выходят ребра) длины не более $2\log_g(|w|)$.

Доказательство.

Пусть D_v — все вершины, достижимые из v в графе вывода.
Присвоим вершинам из D_v веса, чтобы выполнялись условия:

1. вес v равен 1;
2. вес вершины-правила равен сумме весов ее непосредственных предков;
3. если у вершины-правила вес W и у нее k непосредственных потомков, то каждый из них получает вес W/k.

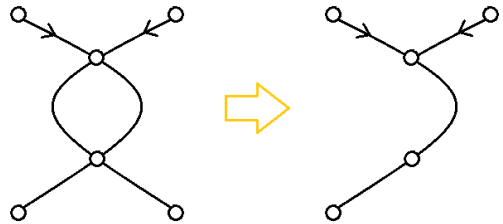

A
B
C
1
1
AB → QRS
1/3
1/3
Q
R
S
1/3
1/3
RSC → Tuvw
2/3
1/6
T
u
v
w
1/6
1/6
1/6
1/2
QT → xyz
x
y
z
1/6
1/6
1/6

Доказательство.

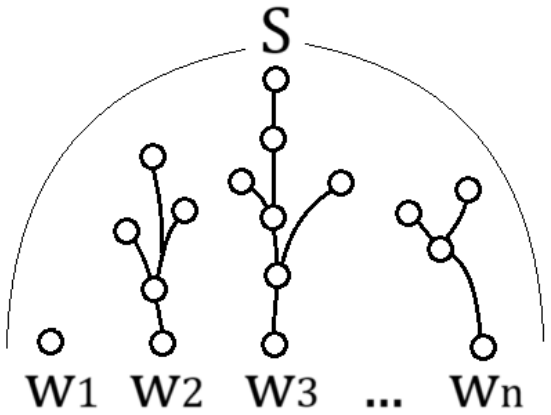
- Для нахождения весов перебираем вершины-правила из D_v сверху вниз и вычисляем их веса и веса их непосредственных потомков.
- На каждом шаге верно свойство: сумма весов стоков равна 1.
- В конце процедуры стоки в D_v — это какие-то из символов итогового слова w; их не более $|w|$ штук, значит, есть сток весом не менее $1/|w|$ ("тяжелый сток").
- У каждой вершины-правила потомков хотя бы в g раз больше, чем предков. Значит, если вес каждого из потомков равен x, то вес хотя бы одного из предков не меньше $g \cdot x$ ("тяжелый предок").
- От тяжелого стока будем каждый раз подниматься к тяжелому предку. Если смогли сделать N шагов, то $\frac{1}{|w|} \cdot g^N \leq 1$.
- Итого: можем добраться до v за $\leq \log_g(|w|)$ шагов; длина пути в 2 раза больше, потому что проходим через вершины-правила.

Совместное множество кратчайших путей

- Зафиксируем кратчайший путь из каждой вершины-символа к какому-нибудь стоку. *Его длина* $\leq 2\log_g(|w|)$.
- Если два пути из разных вершин-символов к стокам в какой-то момент пересеклись, объединим их:



- Итого: каждый сток — корень дерева, состоящего из кратчайших путей. Высота каждого дерева не больше $2\log_g(|w|)$. Каждая вершина принадлежит ровно одному такому дереву.



Полиномиальный алгоритм Дальхауса-Вармута

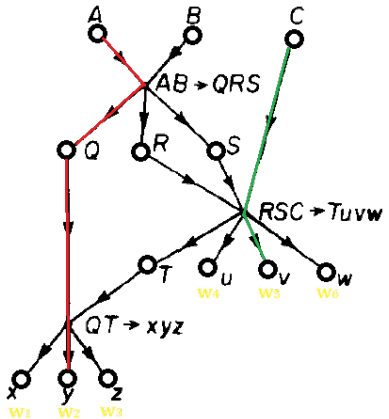
Определение

Разметка — *последовательность меток вершин на пути от вершины-символа к стоку длины не более* $2\log_g(|w|)$.

Полиномиальный алгоритм Дальхауса-Вармута

Храним в памяти фреймы — наборы вида

1. $(AB, \lambda, \rho, l, r)$, где $A, B \in N \cup T$, $1 \leq l \leq r \leq |w|$ и где λ [ρ] — разметка, начинающаяся с A [с B соответственно] и заканчивающаяся w_l [w_r соответственно];
2. (A, σ, k), где $A \in N \cup T$, $1 \leq k \leq |w|$ и где σ — разметка, начинающаяся с A и заканчивающаяся w_k.



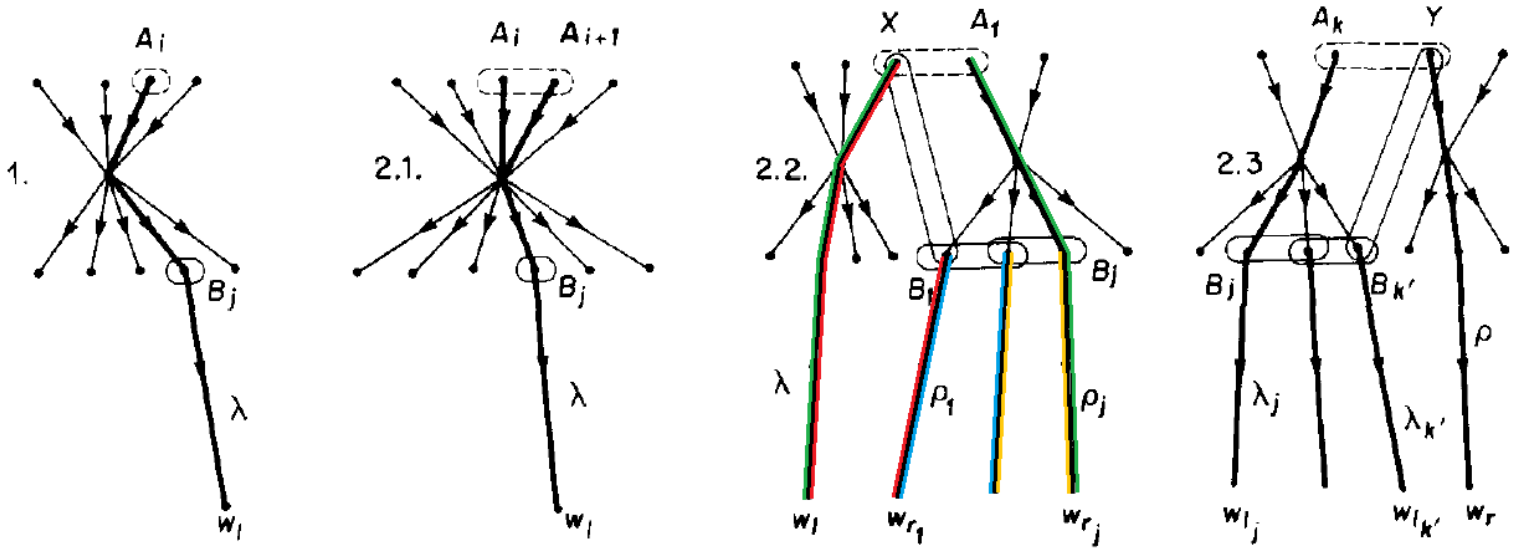
$\lambda = (A; AB \to QRS; Q; QT \to xyz; y)$; $\rho = (C; RSC \to Tuvw; v)$; $l = 2, r = 5$. Тогда $(AC, \lambda, \rho, l, r)$ — корректный бинарный фрейм.

Полиномиальный алгоритм Дальхауса-Вармута

Храним в памяти фреймы — наборы вида

1. $(AB, \lambda, \rho, l, r)$, где $A, B \in N \cup T$, $1 \leq l \leq r \leq |w|$ и где λ [ρ] — разметка, начинающаяся с A [с B соответственно] и заканчивающаяся w_l [w_r соответственно];
2. (A, σ, k), где $A \in N \cup T$, $1 \leq k \leq |w|$ и где σ — разметка, начинающаяся с A и заканчивающаяся w_k.

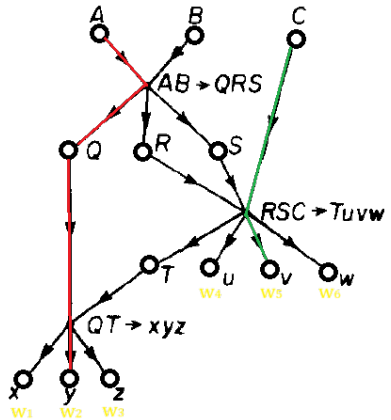
Далее — динамическое программирование.



Полиномиальный алгоритм Дальхауса-Вармута

Храним в памяти фреймы — наборы вида

1. $(AB, \lambda, \rho, l, r)$, где $A, B \in N \cup T$, $1 \le l \le r \le |w|$ и где λ [ρ] — разметка, начинающаяся с A [с B соответственно] и заканчивающаяся w_l [w_r соответственно];
2. (A, σ, k), где $A \in N \cup T$, $1 \le k \le |w|$ и где σ — разметка, начинающаяся с A и заканчивающаяся w_k.



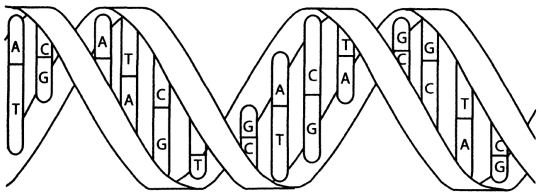
Количество фреймов $\le 2|G|^2 \cdot |G|^{4\log_g(|w|)+2}|w|^2 = 2|G|^4|w|^{4\log_g(|G|)+2}$.

Операции на молекулах ДНК

- Молекула ДНК описывается последовательностью нуклеотидов в алфавите A, C, G, T.

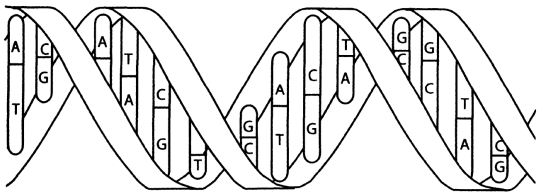
Операции на молекулах ДНК

- Молекула ДНК описывается последовательностью нуклеотидов в алфавите A, C, G, T.
- A и T комплементарны, как и C и G. Комплементарные нуклеотиды способны образовывать водородные связи; так формируется двойная спираль ДНК.

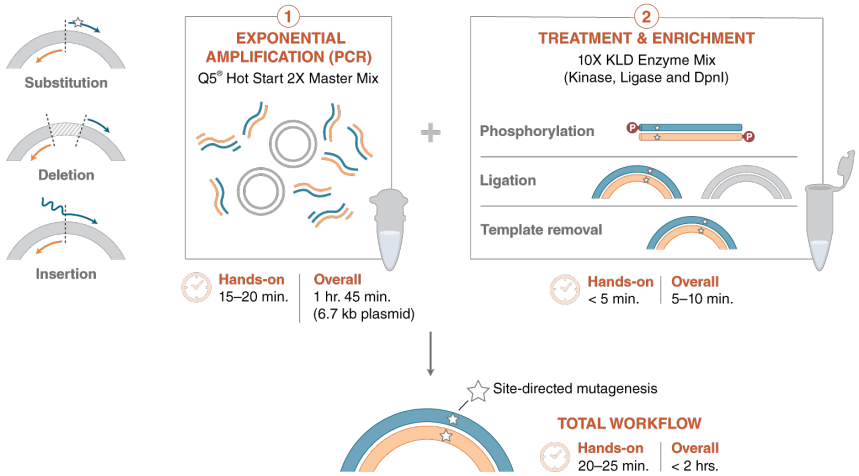


Операции на молекулах ДНК

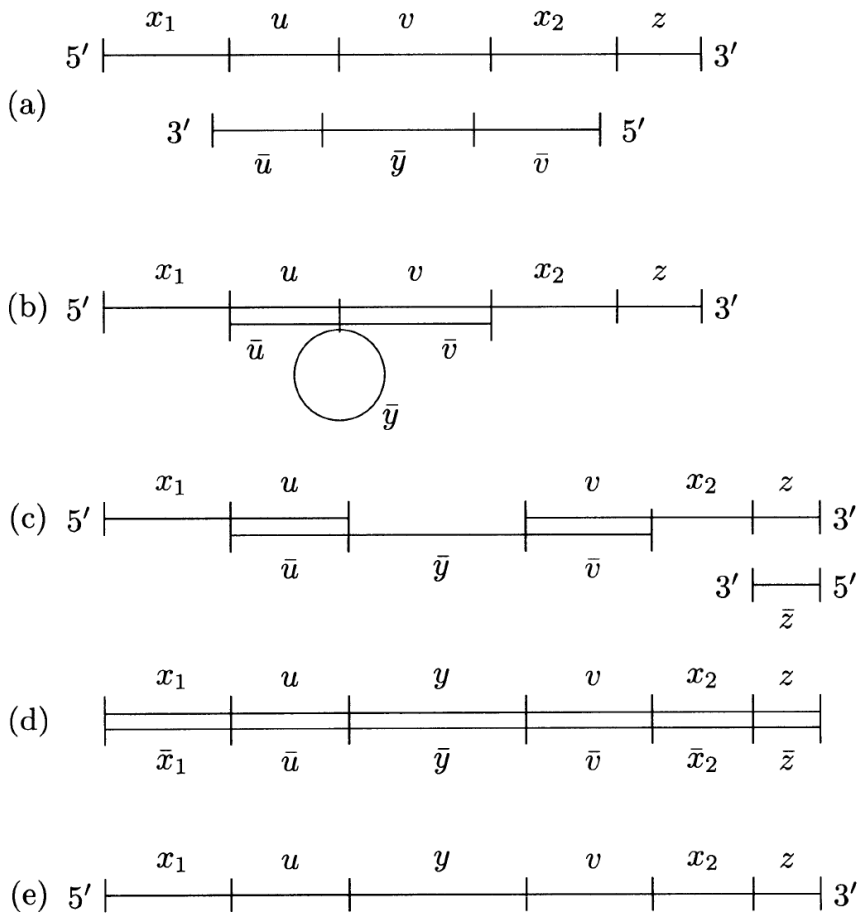
- Молекула ДНК описывается последовательностью нуклеотидов в алфавите A, C, G, T.
- A и T комплементарны, как и C и G. Комплементарные нуклеотиды способны образовывать водородные связи; так формируется двойная спираль ДНК.



- Последовательность нуклеотидов в ДНК можно модифицировать: вставлять, удалять или изменять фрагменты.

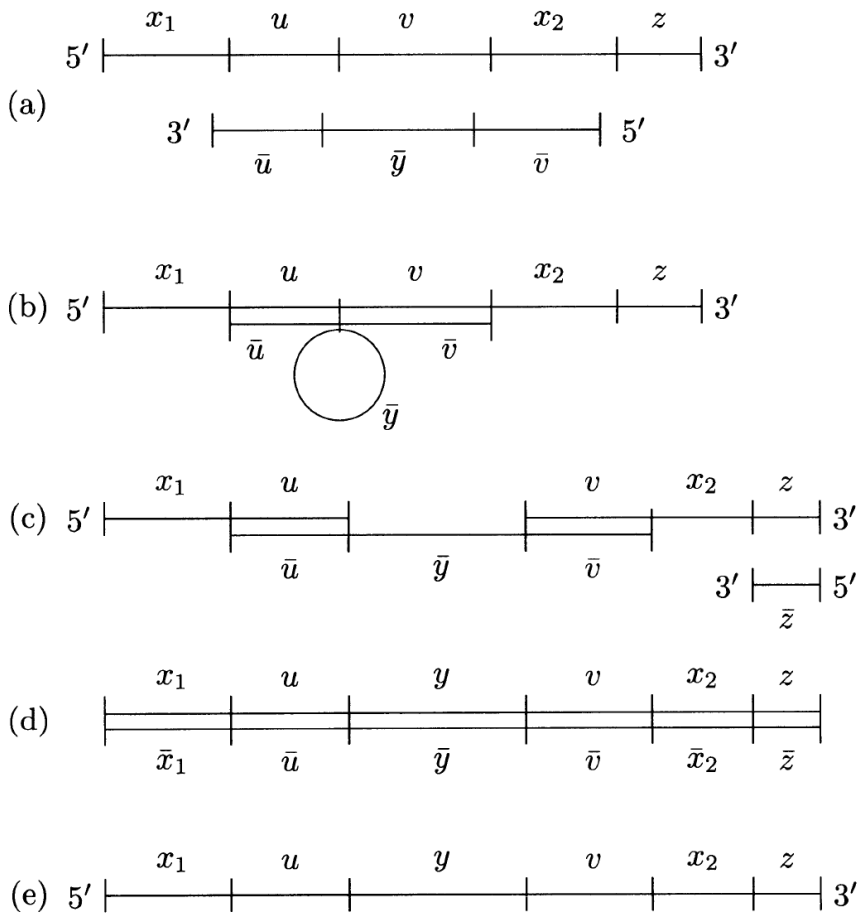


Контекстная вставка



(a) Даны две молекулы ДНК, у которых есть комплементарные фрагменты u и $\overline{u}$, а также v и $\overline{v}$.
(b) Между комплементарными фрагментами образуется связь. Средняя часть $\overline{y}$ остается без пары.
(c) Разрывается связь между u и v в первой молекуле.
(d) Наращивается фрагмент y, комплементарный $\overline{y}$.
(e) Разрывается связь между молекулами.

Контекстная вставка



(a) Даны две молекулы ДНК, у которых есть комплементарные фрагменты u и $\overline{u}$, а также v и $\overline{v}$.
(b) Между комплементарными фрагментами образуется связь. Средняя часть $\overline{y}$ остается без пары.
(c) Разрывается связь между u и v в первой молекуле.
(d) Наращивается фрагмент y, комплементарный $\overline{y}$.
(e) Разрывается связь между молекулами.

Контекстная вставка: $uv \to uwv$. Применение: $x_1uvx_2 \to x_1uwvx_2$.

Грамматики вставок

Определение

Грамматика вставок — формальная грамматика, все правила которой имеют вид $uv \to uwv$.

- A.k.a. полуконтекстные грамматики, системы вставок, грамматики Галюкшова.

Грамматики вставок

Определение

Грамматика вставок — формальная грамматика, все правила которой имеют вид $uv \to uwv$.

- A.k.a. полуконтекстные грамматики, системы вставок, грамматики Галюкшова.
- Борис Семенович Галюкшов: выпускник НГУ, работал в Тверском государственном университете (диссертация 1986 года защищена в Калинине).

Грамматики вставок

Определение

Грамматика вставок — формальная грамматика, все правила которой имеют вид $uv \to uwv$.

- A.k.a. полуконтекстные грамматики, системы вставок, грамматики Галюкшова.
- Борис Семенович Галюкшов: выпускник НГУ, работал в Тверском государственном университете (диссертация 1986 года защищена в Калинине).
- Частный случай удлиняющих грамматик $\Rightarrow$ задают только полиномиальные языки.

Грамматики вставок

Определение

Грамматика вставок — *формальная грамматика, все правила которой имеют вид* $uv \to uwv$.

- A.k.a. полуконтекстные грамматики, системы вставок, грамматики Галюкшова.
- Борис Семенович Галюкшов: выпускник НГУ, работал в Тверском государственном университете (диссертация 1986 года защищена в Калинине).
- Частный случай удлиняющих грамматик $\Rightarrow$ задают только полиномиальные языки.
- Не задают контекстно-свободный язык $\{a^n c b^n \mid n \geq 0\}$ (хотя при этом задают язык $\{a^n b^n \mid n \geq 0\}$).

Грамматики вставок

Определение

Грамматика вставок — *формальная грамматика, все правила которой имеют вид* $uv \to uwv$.

- A.k.a. полуконтекстные грамматики, системы вставок, грамматики Галюкшова.
- Борис Семенович Галюкшов: выпускник НГУ, работал в Тверском государственном университете (диссертация 1986 года защищена в Калинине).
- Частный случай удлиняющих грамматик $\Rightarrow$ задают только полиномиальные языки.
- Не задают контекстно-свободный язык $\{a^n c b^n \mid n \geq 0\}$ (хотя при этом задают язык $\{a^n b^n \mid n \geq 0\}$).
- Способны задавать неконтекстно-свободные языки (Păun 1985).

Грамматики вставок/удалений

Определение

Грамматика вставок/удалений — формальная грамматика, все правила которой имеют вид $uv \to uwv$ *либо* $uwv \to uv$.

- Порождают все перечислимые языки.

Грамматики вставок/удалений

Определение

Грамматика вставок/удалений — формальная грамматика, все правила которой имеют вид $uv \to uwv$ либо $uwv \to uv$.

- Порождают все перечислимые языки.
- Что интереснее, это верно и для бесконтекстных грамматик вставок/удалений (Margenstern, Păun, Rogozhin, Verlan 2005): можно породить любой перечислимый язык грамматиками с правилами вида $\varepsilon \to \alpha$ и $\beta \to \varepsilon$.

Грамматики удалений

Определение

Грамматика удалений — *формальная грамматика G, все правила которой имеют вид $uwv \to uv$.*

- Очевидно, такие грамматики порождают только конечные языки.
- Однако что если вместо стартовой строки брать бесконечный язык с простой структурой, скажем, регулярный?

Грамматики удалений

Определение

Грамматика удалений — *формальная грамматика G, все правила которой имеют вид $uwv \to uv$.*

- Очевидно, такие грамматики порождают только конечные языки.
- Однако что если вместо стартовой строки брать бесконечный язык с простой структурой, скажем, регулярный?
- $\mathrm{Cl}_G(L_0) = \{w \mid \exists w_0 \in L_0 \; w_0 \Rightarrow_G^* w\}$.

Грамматики удалений

Определение

Грамматика удалений — *формальная грамматика G, все правила которой имеют вид $uwv \to uv$.*

- Очевидно, такие грамматики порождают только конечные языки.
- Однако что если вместо стартовой строки брать бесконечный язык с простой структурой, скажем, регулярный?
- $\mathrm{Cl}_G(L_0) = \{w \mid \exists w_0 \in L_0\ w_0 \Rightarrow_G^* w\}$.

Теорема (П., Чо 2025)

Существуют регулярный язык L_0 и грамматика удалений G такие, что $\mathrm{Cl}_G(L_0)$ — Σ_1^0-полный язык.

Грамматики удалений

Определение

Грамматика удалений — формальная грамматика G, все правила которой имеют вид $uwv \to uv$.

- Очевидно, такие грамматики порождают только конечные языки.
- Однако что если вместо стартовой строки брать бесконечный язык с простой структурой, скажем, регулярный?
- $\mathrm{Cl}_G(L_0) = \{w \mid \exists w_0 \in L_0\ w_0 \Rightarrow_G^* w\}$.

Теорема (П., Чо 2025)

Существуют регулярный язык L_0 и грамматика удалений G такие, что $\mathrm{Cl}_G(L_0)$ — Σ_1^0-полный язык.

- Ср. с тем, что если все правила в G имеют вид $\alpha \to \beta$, где $|\beta| \leq 1$ и L_0 регулярный, то $\mathrm{Cl}_G(L_0)$ тоже регулярный.