



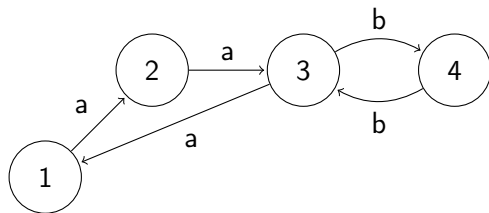
# Эффективные алгоритмы для CFL-reachability

Екатерина Шеметова

JetBrains Research, Programming Languages and Tools Lab  
Санкт-Петербургский Академический Университет

14.12.2019

достижимость в графе + контекстно-свободные ограничения



$S \rightarrow aSb \mid ab$

Решения:

$(S, 2, 4), (S, 1, 3), \dots, (S, 3, 3)$

2-4:  $ab$

1-3:  $aabb$

3-4:  $aaabbb$

...

3-3:  $aaaaaabbabbbb$

- Вход: контекстно-свободная грамматика  $G$  и помеченный ориентированный граф  $D$
- Сферы применения: статический анализ кода, запросы к графовым базам данных

**Цель:** Хотим эффективно решать задачу CFL-reachability

**Проблемы:**

- ❶ Классический алгоритм для решения задачи CFL-reachability работает за  $O(n^3)$  ( $n$  — число вершин в графе).

**Большая открытая проблема:** существует ли алгоритм, работающий за  $O(n^{3-\epsilon})$  (т.н. *субкубический алгоритм*)?

**Цель:** Хотим эффективно решать задачу CFL-reachability

**Проблемы:**

- 1 Классический алгоритм для решения задачи CFL-reachability работает за  $O(n^3)$  ( $n$  — число вершин в графе).

**Большая открытая проблема:** существует ли алгоритм, работающий за  $O(n^{3-\varepsilon})$  (т.н. *субкубический алгоритм*)?

- 2 Параллельный алгоритм — доказано, что для задачи CFL-reachability нельзя построить эффективный параллельный алгоритм

**Цель:** Хотим эффективно решать задачу CFL-reachability

**Проблемы:**

- ❶ Классический алгоритм для решения задачи CFL-reachability работает за  $O(n^3)$  ( $n$  — число вершин в графе).  
**Большая открытая проблема:** существует ли алгоритм, работающий за  $O(n^{3-\varepsilon})$  (т.н. *субкубический алгоритм*)?
- ❷ Параллельный алгоритм — доказано, что для задачи CFL-reachability нельзя построить эффективный параллельный алгоритм

**Подход:** Попробуем найти подклассы грамматик, для которых задача может быть эффективно решена

- Хотим знать, какие задачи эффективно параллелятся, а какие — нет
- Что в теории значит “эффективно параллеляются”?

- Хотим знать, какие задачи эффективно параллелятся, а какие — нет
- Что в теории значит “эффективно параллеляются”?
  - ▶ Для входа длиной  $n$  задача может быть решена:
    - ★ полиномиальным от  $n$  числом процессоров и
    - ★ за полилогарифмическое от  $n$  время  $\text{polylog}(n)$
  - ▶ Если время хотя бы линейно от входа, то параллелить уже неэффективно

- Классический результат: в общем случае для задачи CFL-reachability нельзя построить эффективный параллельный алгоритм
- Но утверждается, что для некоторых фиксированных классов контекстно-свободных грамматик данная задача эффективно параллелится



- Классический результат: в общем случае для задачи CFL-reachability нельзя построить эффективный параллельный алгоритм
- Но утверждается, что для некоторых фиксированных классов контекстно-свободных грамматик данная задача эффективно параллелится
- **Вопросы:** Что объединяет эти эффективные классы? Много ли их? Почему они обладают этим свойством? Можем ли мы получить новые, полезные на практике?

# Эффективные подклассы

- Эффективность зависит от специального параметра языка — рационального индекса  $\rho_L(n)$
- $\rho_L(n) = \max\{\min\{|w| : w \in L \cap K\}, K \in Rat_n, L \cap K \neq \emptyset\}$

$S \rightarrow aSb \mid ab$

Решения:

$(S, 2, 4), (S, 1, 3), \dots, (S, 3, 3)$

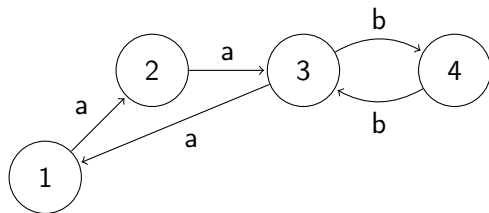
2-4:  $ab$

1-3:  $aabb$

3-4:  $aaabbb$

...

3-3:  $aaaaaabbabbbbbb$



- Эффективность зависит от специального параметра языка — рационального индекса  $\rho_L(n)$
- $\rho_L(n) = \max\{\min\{|w| : w \in L \cap K\}, K \in Rat_n, L \cap K \neq \emptyset\}$
- Если индекс полиномиален от числа вершин в графе — задача может быть эффективно распараллелена, экспоненциален — нет
- Что влияет на величину индекса?

# Эффективные подклассы

- Эффективность зависит от специального параметра языка — рационального индекса  $\rho_L(n)$
- $\rho_L(n) = \max\{\min\{|w| : w \in L \cap K\}, K \in Rat_n, L \cap K \neq \emptyset\}$
- Если индекс полиномиален от числа вершин в графе — задача может быть эффективно распараллелена, экспоненциален — уже нет
- Что влияет на величину индекса?
  - ▶ Некоторые ограничения на стек pushdown автомата делают рациональный индекс соответствующего языка полиномиальным!
  - ▶ Примеры: one-turn, oscillation-boundness, один элемент в стековом алфавите и т.д.
- Также полезно изучить влияние операций над языками на величину индекса

- Ограничения на стек pushdown автомата дают нам эффективные языки
- Найдены новые подклассы, для которых можно построить эффективный параллельный алгоритм: металинейные, суперлинейные, oscillation-bounded языки
- А также известные эффективные языки можно комбинировать с помощью операций конкатенации, union, пересечения с регулярными языками, substitution closure и др. и получать новые эффективные примеры
- Для решения задачи адаптирован параллельный алгоритм Брента–Гольдшляггера–Риттера (на эффективных подклассах число процессоров  $O(n^6)$ , время —  $O(\log^2 n)$ ,  $n$  — число вершин в графе)

А что по поводу субкубического алгоритма?

- Эффективные для распараллеливания подклассы языков более эффективны и для последовательного алгоритма
- Благодаря использованию класса one-counter языков, была получена аппроксимация общей задачи за субкубическое время

- **Журнал “Theory of Computing Systems”**

Ekaterina Shemetova, Semyon Grigorev and Alexander Okhotin.

Parallel efficient variants of context-free language reachability problem.

*Статус: планируется публикация*

Прошрое полугодие:

- **Журнал “Труды Института системного программирования РАН”**

Шеметова Е.Н., Григорьев С.В. Задача поиска путей в ациклических графах с ограничениями в терминах булевых грамматик.

*Статус: опубликовано*

- Построение обобщенного субкубического алгоритма для найденных эффективных подклассов КС-языков
- Планируется найти эффективные подклассы графов путем изучения влияния структурных характеристик графа, таких как *treewidth*, *pathwidth*, *cutwidth* и др., на эффективность решения задачи
- Пробуем использовать найденные эффективные подклассы (как графов, так и грамматик) для получения решения задачи в общем случае
- Подобных проблем много (например APSP), и занимается ими целое сообщество (“fine-grained” complexity). Ждем ответа от Barna Saha (University of California Berkeley, автор субкубического алгоритма для LED).