Software Engineering Conference Russia 2018

October 12-13
Moscow

Восстановление дерева процессов Linux трансформациями дерева, управляемыми атрибутной грамматикой

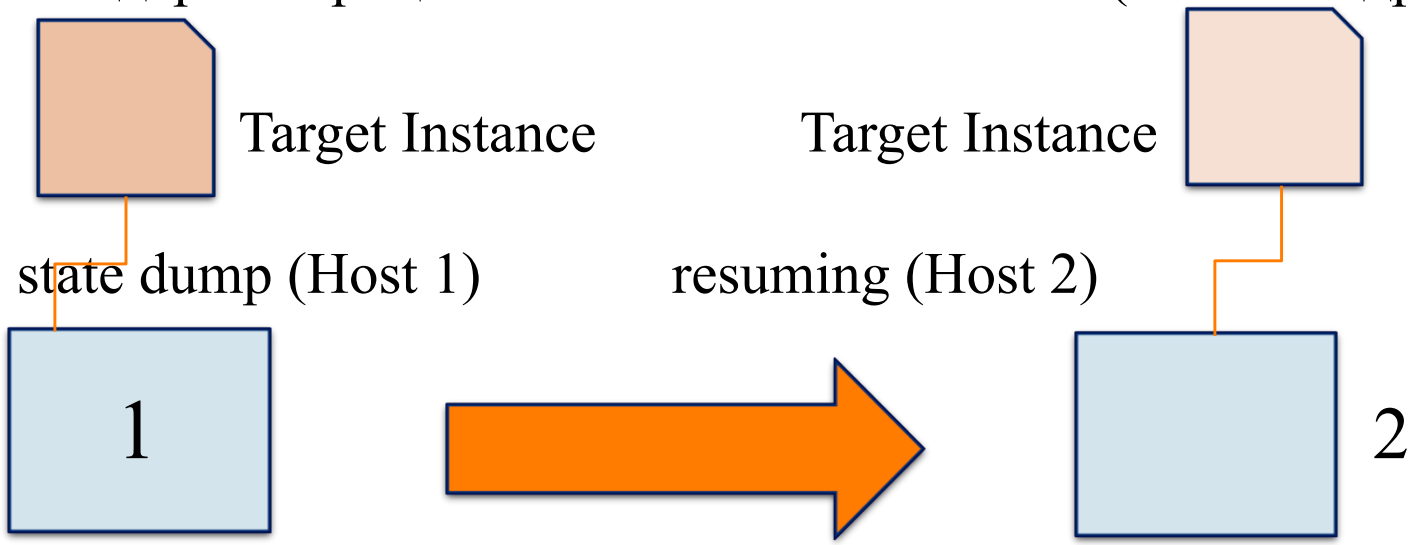
Николай Ефанов

MIPT
MOSCOW INSTITUTE
OF PHYSICS AND TECHNOLOGY

Цель построения модели сохранения/ восстановления и прикладные задачи

Задача: сохранение и восстановление состояния процессов и окружения из пространства пользователя. Строгая модель сможет предоставить:

1. Непосредственный анализ атрибутов в Checkpoint-Restore (CRIU, etc).
2. Сокращение накладных расходов при живой миграции.
3. В идеале: дерево процессов + полезный контекст (память и др.).



Задача и ограничения:

1. Восстановить последовательности системных вызовов, порождающие входное дерево процессов D. Вызовы хранятся в дереве T:

$$D = \{V, E\}, \rightarrow T = \{V^+, E^+\}, V^+, E^+ \text{ - хранят цепочки}$$

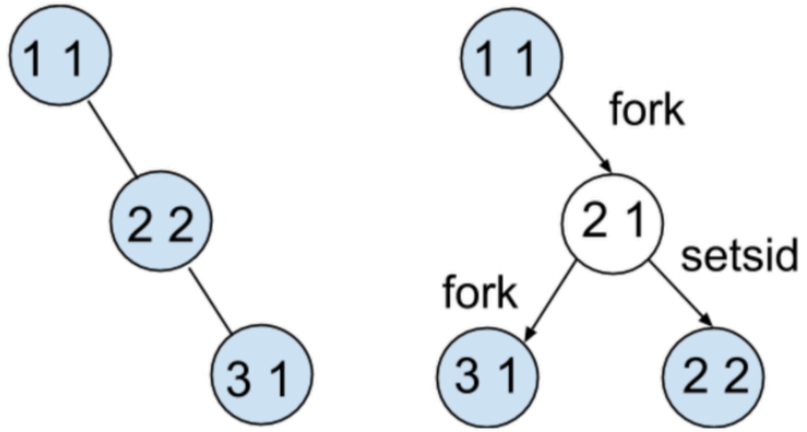
$$|E \backslash (E \cap E^+)| \geq 0$$

$$V = V \cap V^+$$

2. Ограничения:

- Только Linux
- Ввод корректен. Иначе: Ошибка разбора.
- Системные вызовы:
  - ✓ **Fork**: создать процесс-потомок
  - ✓ **Setsid**: создать новую сессию
  - ✓ **Setpgid**: установить группу вызвавшему: setpgid(0,pgid)
  - ✓ **Exit:** завершить процесс -> присвоить потомков Init-процессу

✓ → Базис трансформаций дерева процессов



Анализ задачи: комбинаторные оценки

Число деревьев, полученных с **fork** (из формулы Кейли для деревьев*):

$$F(n) = \sum_{i=2}^{n-1} i^{i-2}$$

И это с учётом ‘нечувствительности’ к перестановкам идентификаторов!

Добавив абстрактный вызов **k call**, изменяющий идентификатор **k:**

$$F(n, k\_call) = F(n) \sum_{m=0}^{n-1} \binom{n}{m} (m+1)^{n-m-1}$$

Прямая генерация – не лучшая идея!

* Ефанов Н.Н. Комбинаторые и групповые свойства деревьев процессов Linux // Сборник трудов XV международной конференции «Алгебра, теория чисел и дискретная геометрия», Тула, 28-31 мая 2018 г., С. 180-183.

Предшествующая работа: грамматика строк

Строчная нотация + Набор правил:

- Процессы представляются как “p, g, s [children] ” и рекурсивно перечисляются
- Набор правил переписывания строк:
  - Пример правила: **fork**(* * * [*]) --> * * *[* \2 \3 [] \4].
  - Левые части могут содержать контексты – «если конфигурация такая, тогда…»
  - Правило **exit** удаляет данные{1 * * [*], **exit**(* * * [*])} --> 1 \1 \2 [\3 \7].

- Грамматика не является контекстно-свободной (**setpgid**)
- Это грамматика Типа 0 по Хомскому (см. **exit**)
- Эвристики в анализе позволяют решить задачу за O(P(n))

*Nikolay Efanov and Pavel Emelyanov. 2017. Constructing the formal grammar of system calls. In Proceedings of the 13th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR '17). ACM, New York, NY, USA, Article 12, 5 pages.

Предшествующая работа: грамматика строк

- «BPSF»: Двухстадийный O(Nlog(N)log(S)log(P))-time анализатор:
  - Стадия 1: бесконтекстный анализ --> промежуточное состояние в стеке
  - Стадия 2: разбор контекста --> ответ
    - +AVL-структуры поиска параметров: логгирование p, g, s в ходе работы

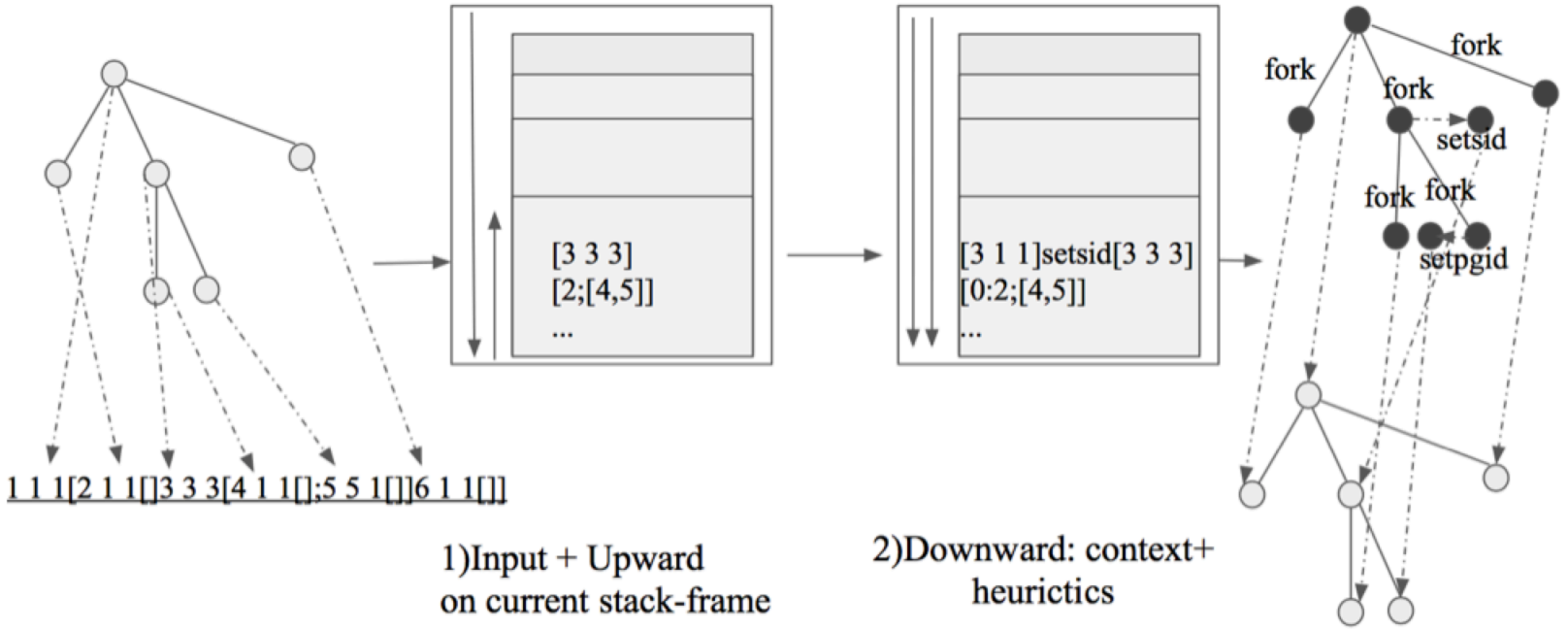


Схема «Дерево процессов→Строка→Стек с кадрами+Структуры поиска→Дерево»

Предшествующая работа: грамматика строк

- «BPSF»: Двухстадийный O(Nlog(N)log(S)log(P))-time анализатор:
  - Стадия 1: бесконтекстный анализ --> промежуточное состояние в стеке
  - Стадия 2: разбор контекста --> ответ
    - +AVL-структуры поиска параметров: логгирование p, g, s в ходе работы

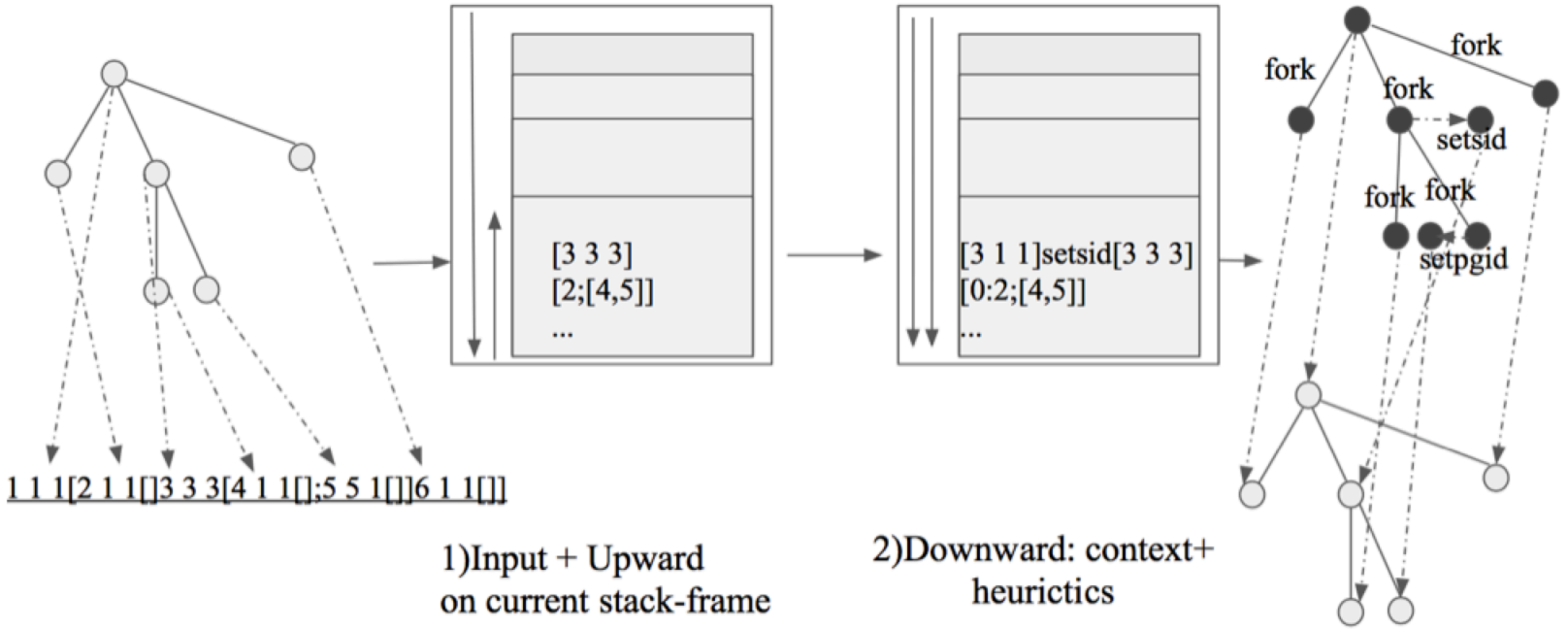


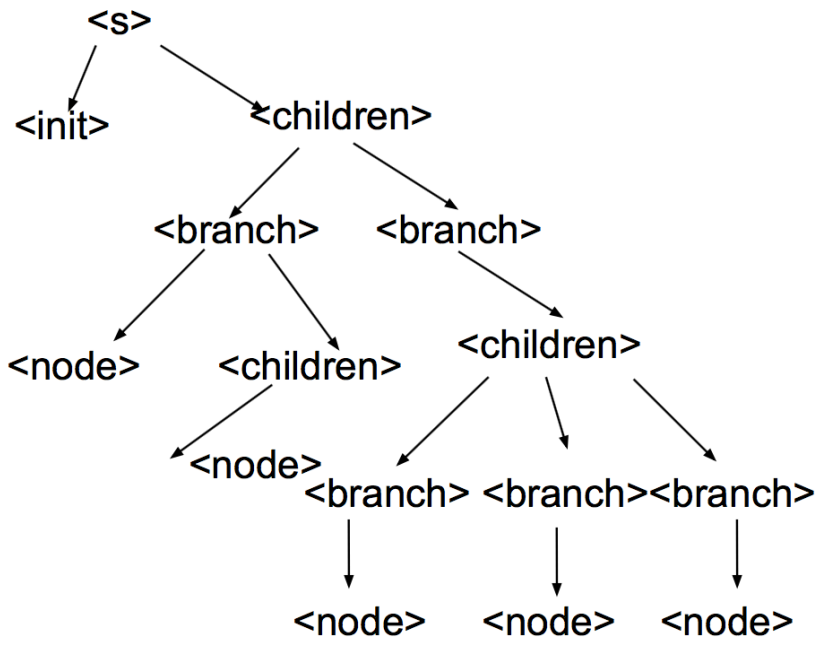
Схема «Дерево процессов→Строка→Стек с кадрами+Структуры поиска→Дерево»

! Некоторая избыточность преобразований. Какой выход?

КС-грамматика {A, N, P,<s>}

- **A** – терминалы
- **N** = {<s>, <init>, <children>, <branch>,<node>}
- **<s>** - стартовый нетерминал
- **P** – набор правил:

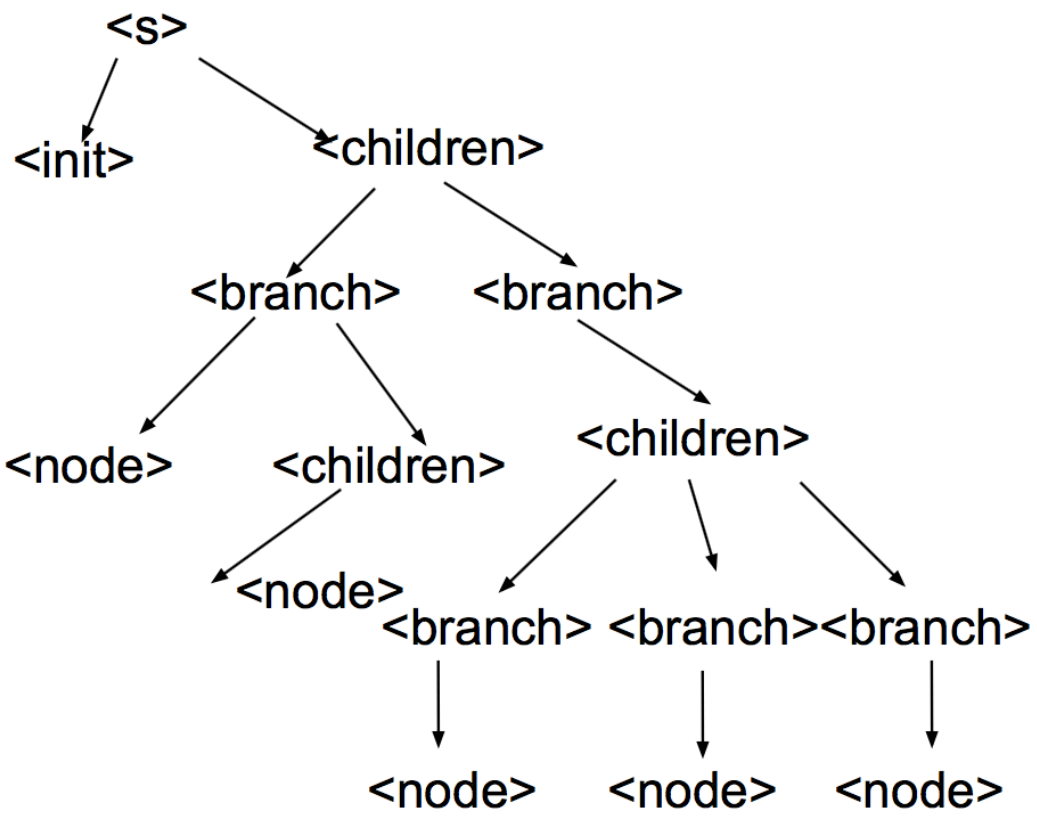
  <s>=<init><children>|<init>
  <init>=”1..1”
  <children>={<branch>}
  <branch>=<node><children>|<node>
  <node>= строка терминалов



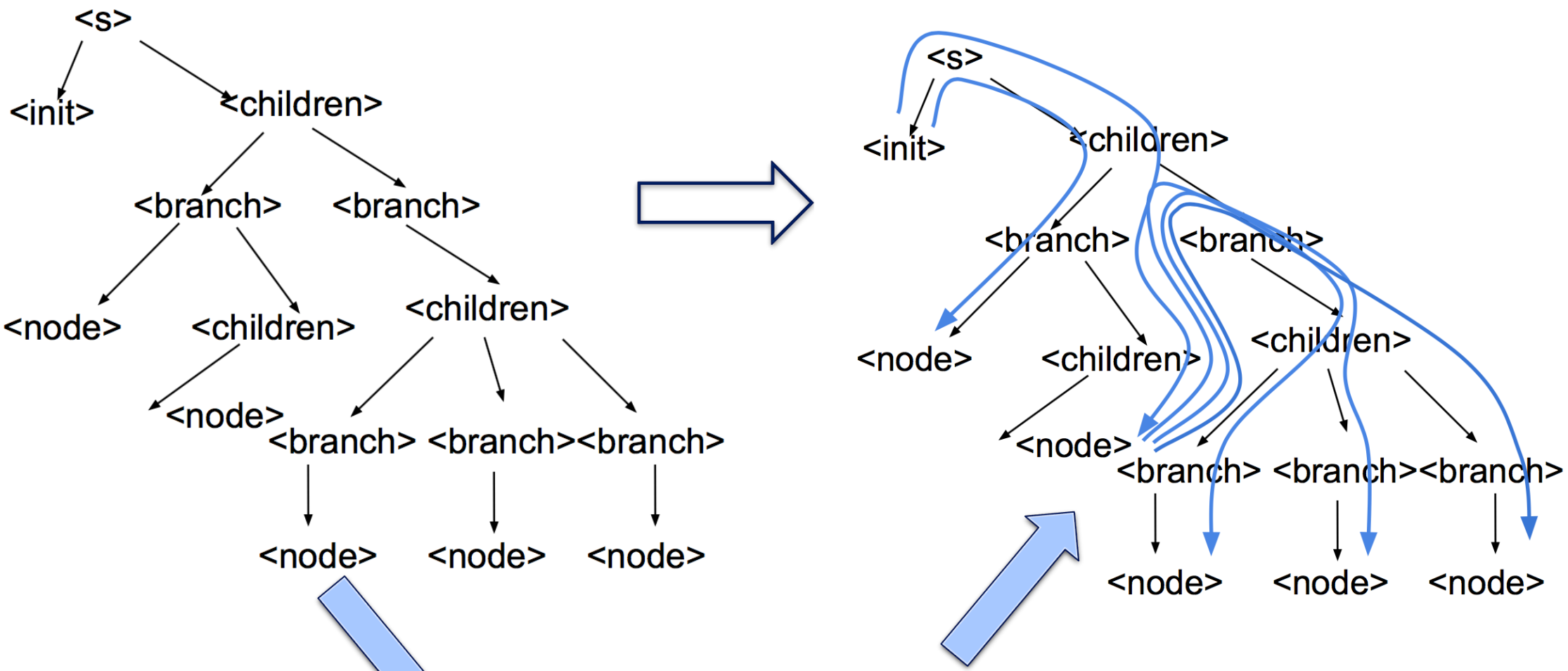
Такая грамматика задаёт язык деревьев с выделенной корневой вершиной – то что нужно !

- Хранение параметров процессов в терминалах
- Извлечение параметров для анализа

Дерево разбора →Дерево процессов



Дерево разбора →Дерево процессов

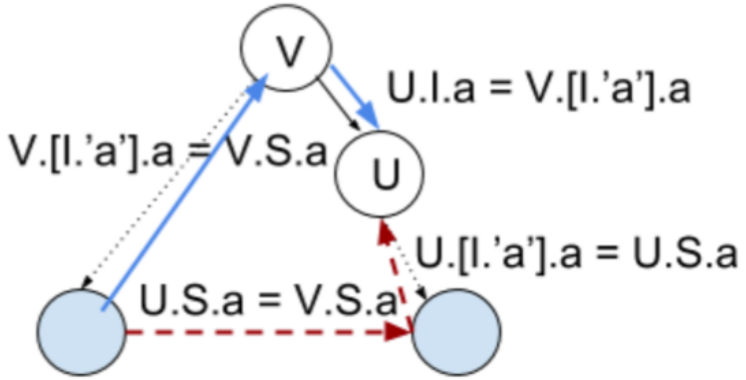


$$E = E^{<branch>}_{<node>} \cup E^{<children>}_{<branch>} \cup E^{<branch>}_{<children>} \cup E^{<node>}_{<branch>},$$

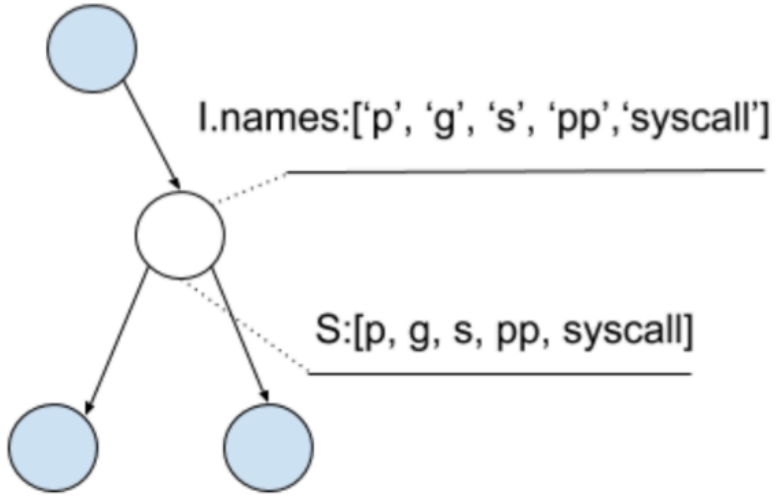
$$E = E^{<s>}_{<init>} \cup E^{<children>}_{<s>} \cup E^{<branch>}_{<children>} \cup E^{<node>}_{<branch>}$$

Атрибутная грамматика деревьев процессов: {A, N, P, <s>, **AT, SA**}

- **AT** – Набор атрибутов:
  - .I – наследуемые – ‘переменные’: p, pp, g, s, ‘syscall’.
  - .S – синтезируемые – значения переменных.
- **SA** – Семантические действия:
  - Переписывание атрибутов
  - Добавление вершин
  - Добавление и удаление рёбер



a)



b)

Attributed Grammar Tree Transformation Method: анализатор

- Последовательные трансформации дерева на основании проверок атрибутов
- **В результате: неявное построение графа зависимостей текущей вершины обходом дерева**
- **→ Трансформации**
- **O(n^2)-Time** в худшем случае (доказательство см. в статье).

Attributed Grammar Tree Transformation Method: анализатор

- Последовательные трансформации дерева на основании проверок атрибутов
- **В результате: неявное построение графа зависимостей текущей вершины обходом дерева**
- **→ Трансформации**
- **O(n^2)-Time** в худшем случае (доказательство см. в статье).

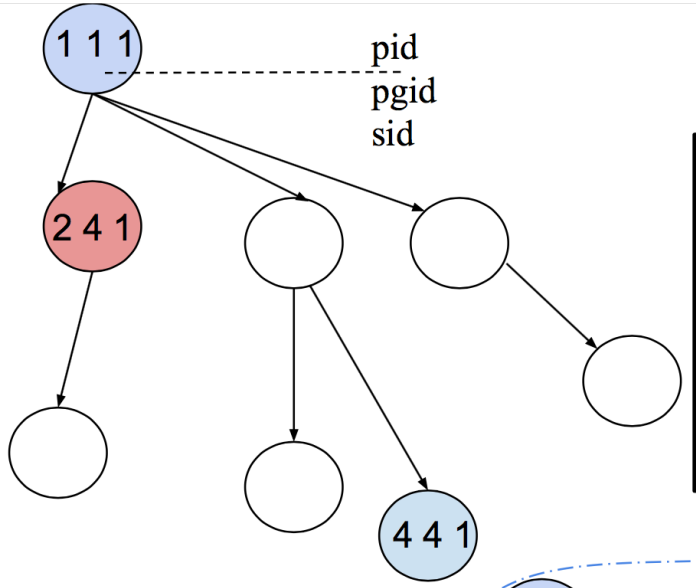
Theorem 2 (on time complexity). The worst-case time complexity of Algorithm 1 *is* $O(n^2)$ for n nodes in the input tree, if all of traverse type-dependent checks can be performed via single *upbranch* and optionally single *dfs* routines.

```
Subprogram 1:
context_checking():
        cond = upbranch({expr(Node, Current)})
    if not cond in IR:
            cond = dfs ({expr(Node, Current)})
        return cond
```

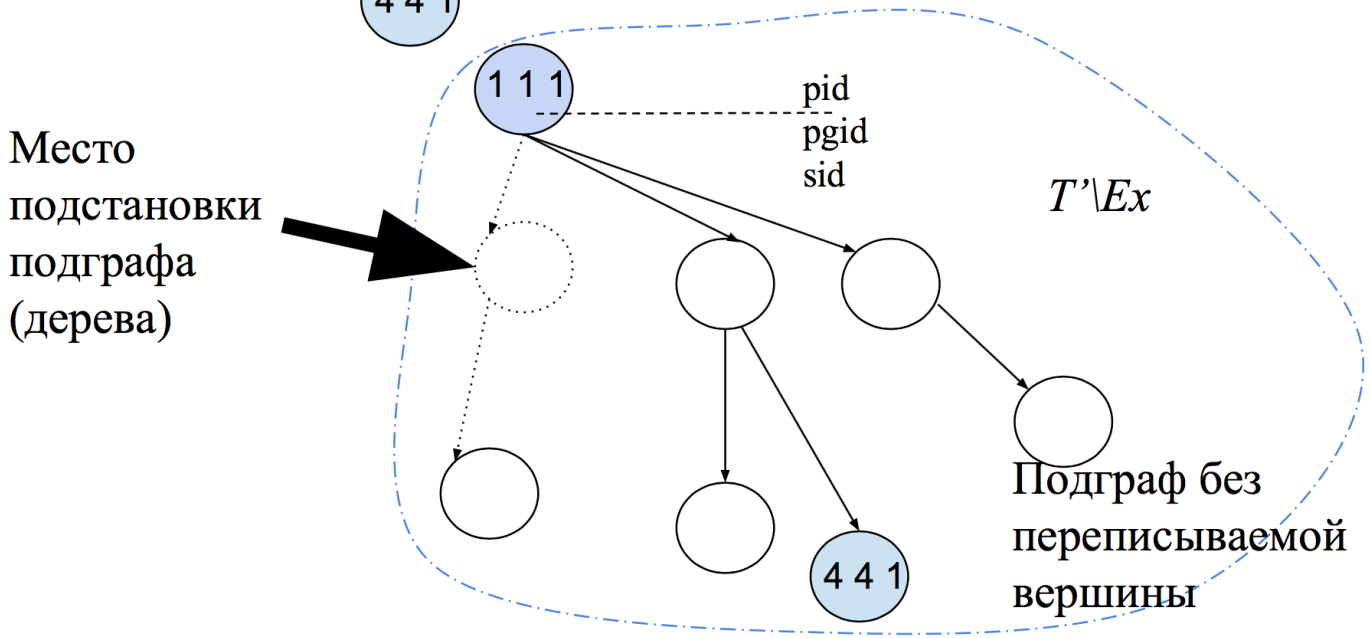
Q.E.D.

```
Algorithm 1:
(Input: IR, D, expr; Output: T)
begin:
    for any Node in dfs(T.root):
    cond = f(expr(Node.S,I [1..m])) # m – attributes
    if cond not in IR: # context check
            cond=context_checking()
            ...
     D= TR(D, Ex, IR, k(cond))
     for child in Node.children:
            dfs(child)
    return D
```

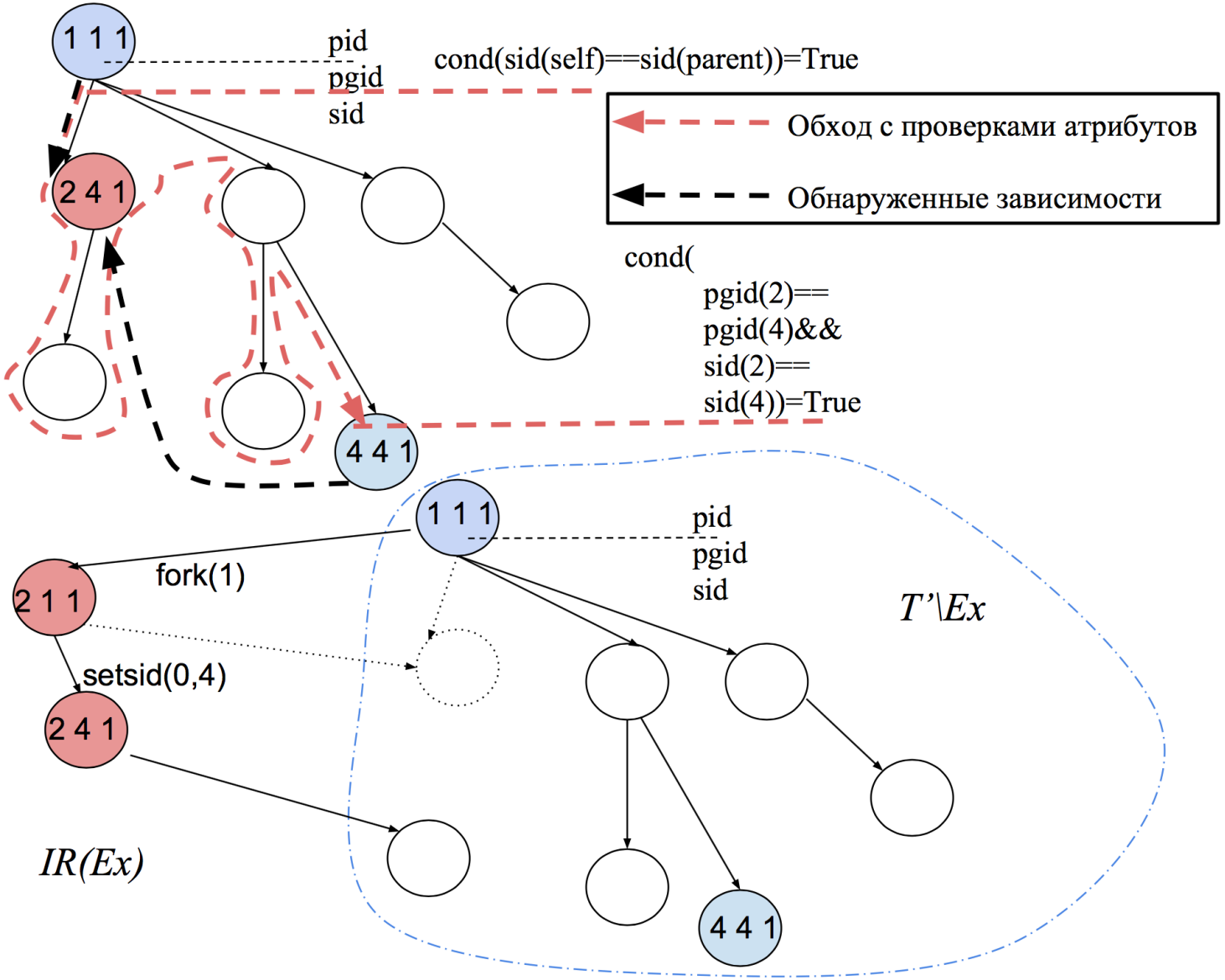
AGTTM - анализатор: пример



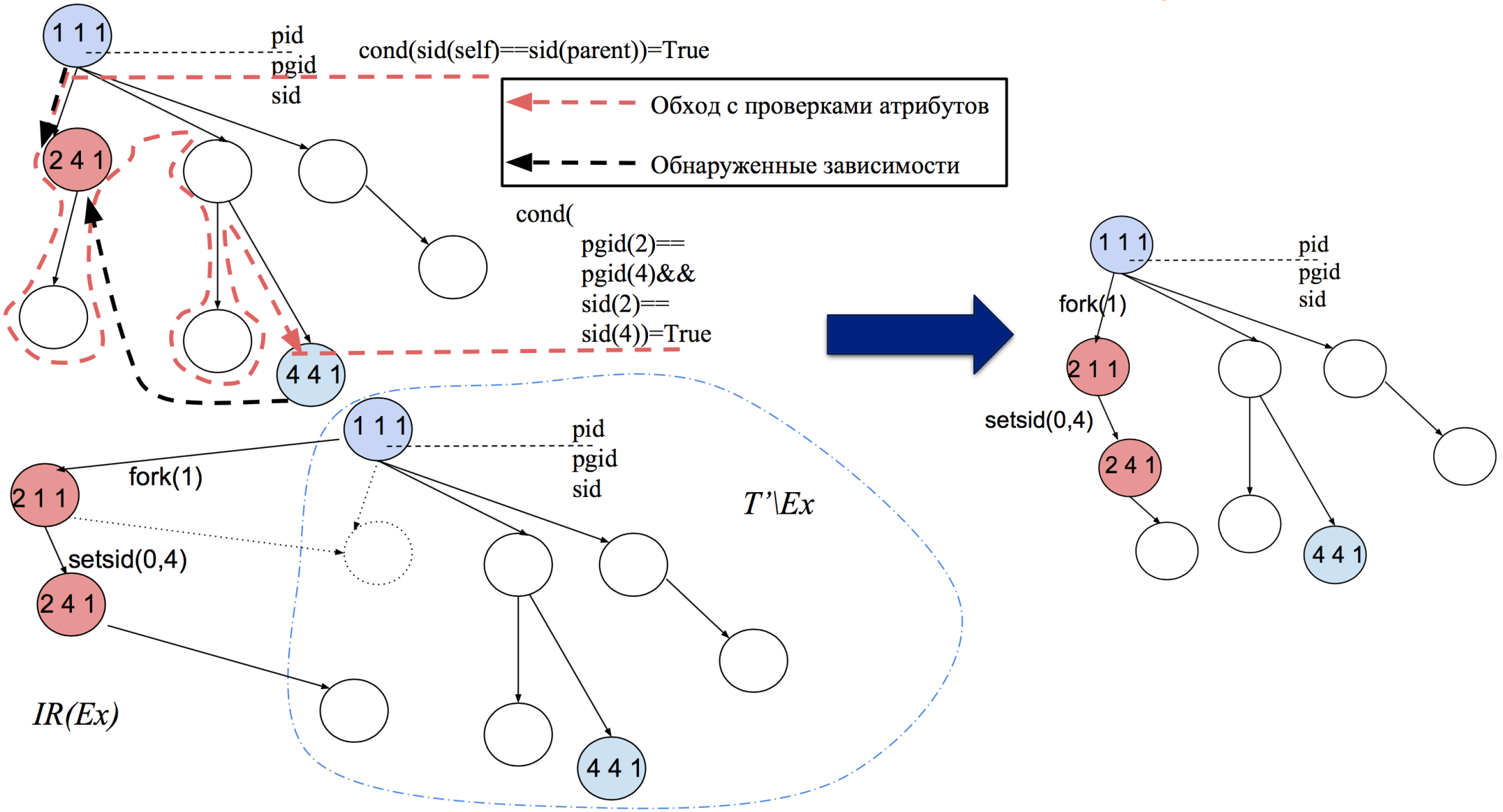
Задача: найти атрибуты в вершинах, от которых зависят атрибуты в текущей. “Безопасно” исключить вершину или подграф *Ex*. Сформировать или выбрать переписываемый подграф *IR(Ex)*.



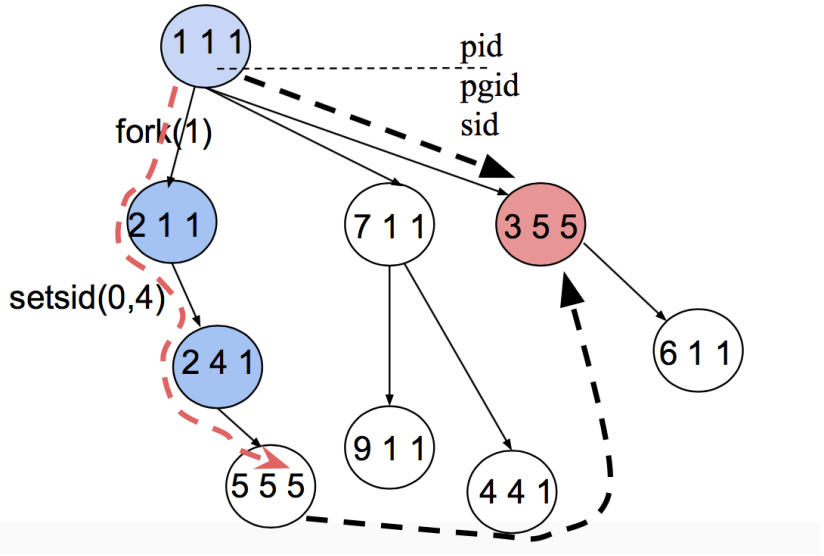
AGTTM - анализатор: пример



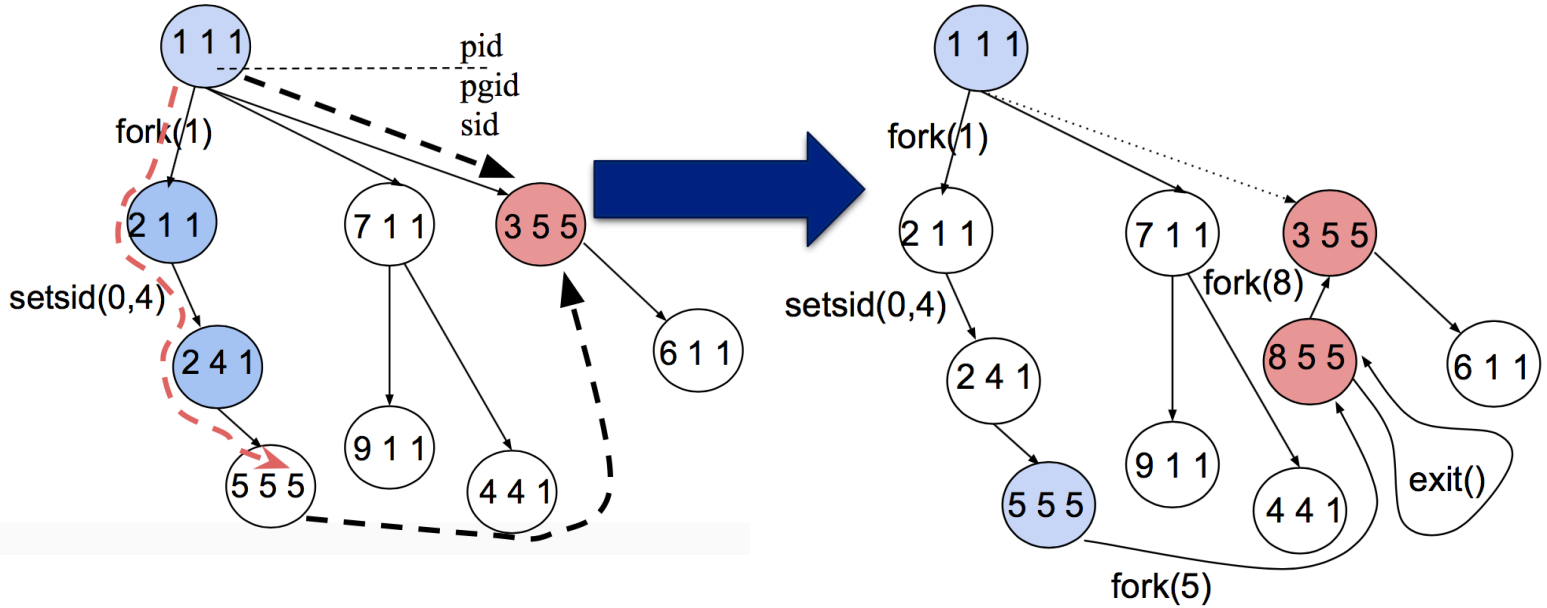
AGTTM - анализатор: пример



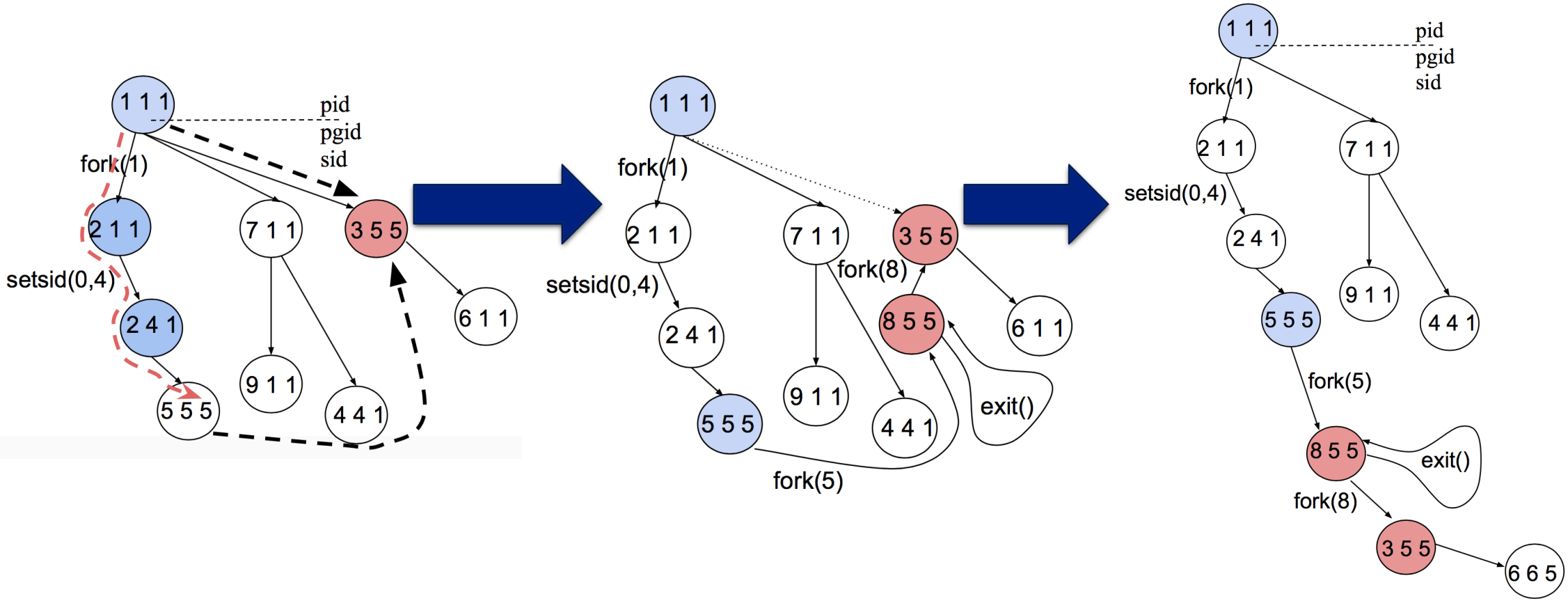
AGTTM - анализатор: пример с **exit()**



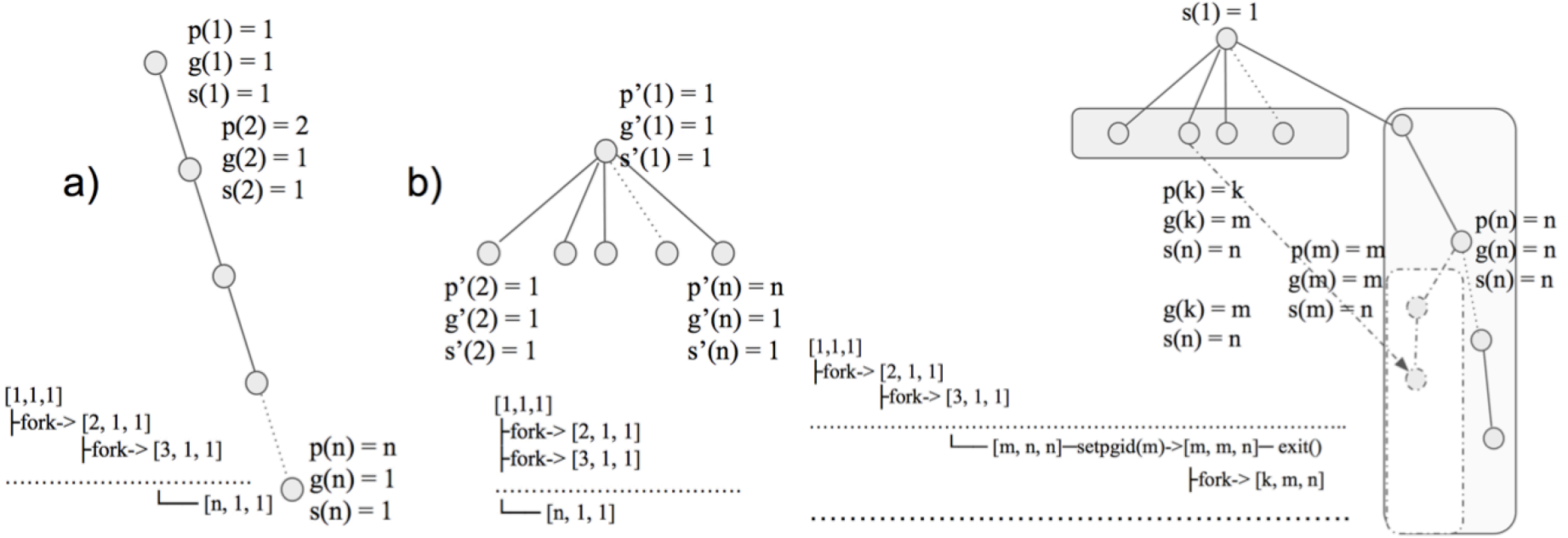
AGTTM - анализатор: пример с **exit()**



AGTTM - анализатор: пример с **exit()**



Тесты и метрика затрат:



"Simple-load" profile of tests

Reverse reparent-highloaded "heuristic" test

$$C_{overall} = C_s + C_r$$

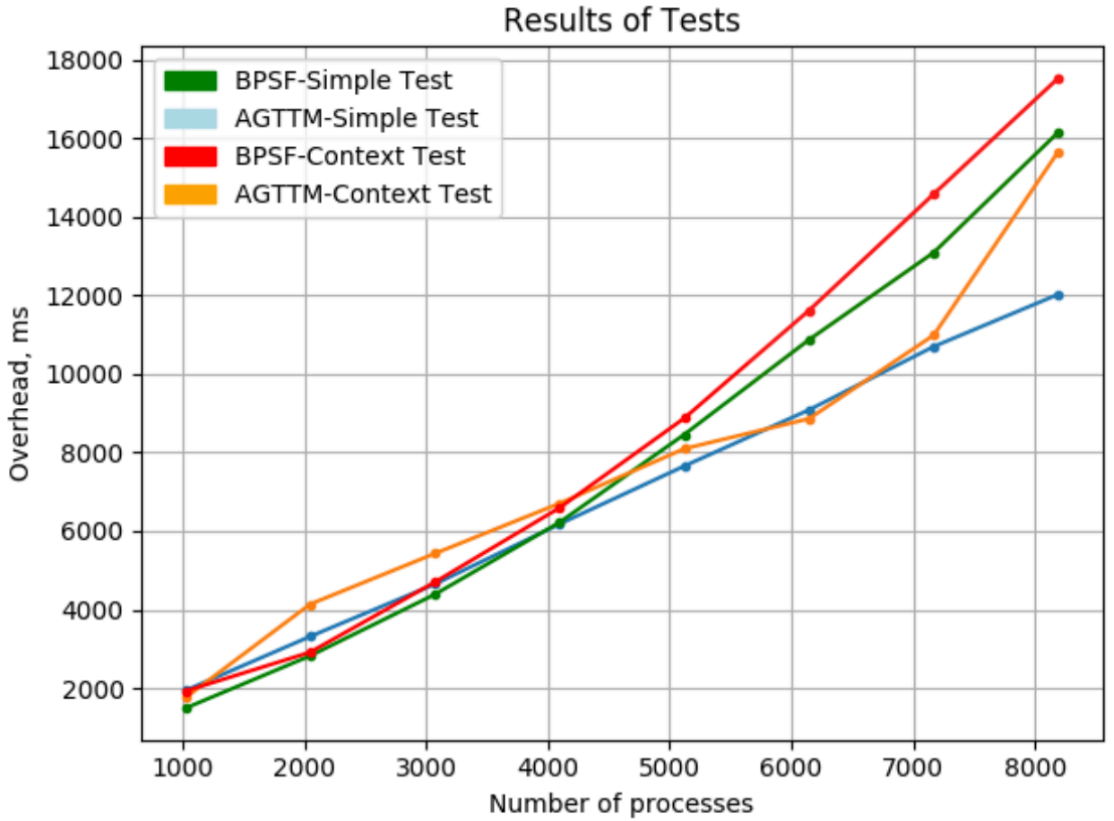
Результаты экспериментов

Table 1: The "Simple-load" test results

Number of Processes	AGTTM	BPSF
1024	1938ms	1488ms
2048	3314	2821
3072	4645	4385
4096	6176	6218
5120	7651	8453
6144	9080	10865
7168	10692	13085
8192	12023	16149

Table 2: The "Heuristic-load" test results

Number of Processes	AGTTM	BPSF
1024	1755ms	1925ms
2048	4135	2913
3072	5425	4696
4096	6701	6588
5120	8092	8868
6144	8856	11610
7168	10981	14573
8192	15659	17527



Выводы и дальнейшие действия

1. Метод на основе атрибутных грамматик применим к поставленной задаче

2. Метод эффективнее, чем строчный, при числе процессов > 4000-5000

3. Правила трансформации ограничены только применяемым методом
   → негативная черта строчного метода устранена!

•Дальнейшие действия:
- ➢Поддержка новых системных вызовов/функций/аспектов
- ➢Разработка других методов: «семантика как синтаксис» – грамматики с контекстами, мультикомпонентные грамматики надстройки деревьев
- ➢Статистический анализ деревьев процессов, выработка метрик, оптимизация последовательностей системных вызовов.
- ➢Интеграция с системами сохранения-восстановления (CRIU иди др.)

Благодарю за внимание!

Контактная информация:

- Николай Ефанов
- Email: nefanov90@gmail.com
- Phone:+7(916)0911108

Hidden2: The AGTTM analyzer: DFS routine with extra DFS / upbranch traversal checks

Algorithm 1:
(Input: *IR*, *D*, *expr*; Output: *T)*
begin:

```
for any Node in dfs(T.root):
cond = f(expr(Node.S,I [1..m])) # m – attr
if cond not in IR: # context check
        cond=context_checking()
        …
 D= TR(D, Ex, IR, k(cond))
 for child in Node.children:
        dfs(child)
return D
```

Theorem 2 (on time complexity). The worst-case time complexity of Algorithm 1 *is* $O(n^2)$ for n nodes in the input tree, if all of traverse type-dependent checks can be performed via single *upbranch* and optionally single *dfs* routines.

Subprogram 1:

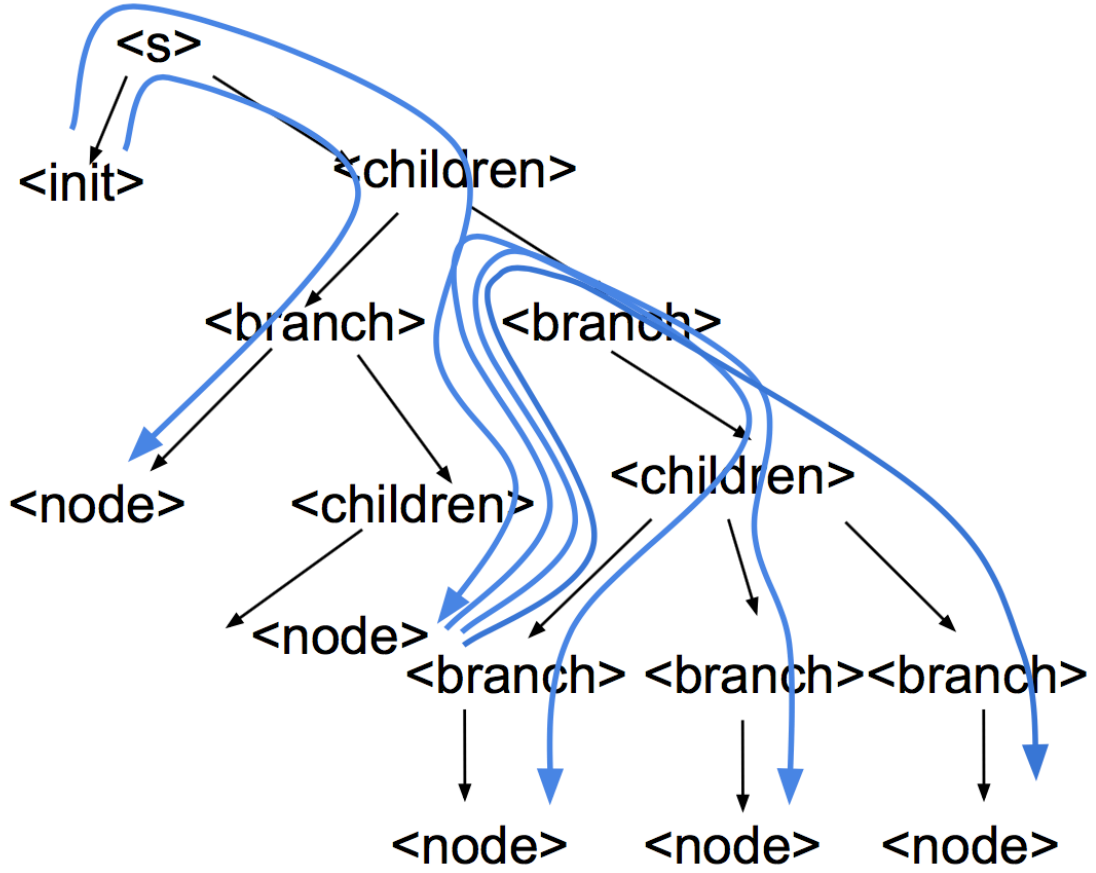
```
context_checking():
        cond = upbranch({expr(Node, Current)})
    if not cond in IR:
            cond = dfs ({expr(Node, Current)})
        return cond
```

Q.E.D.

The main recap:

Total time complexity is quasi-square of nodes number *n*

Дерево разбора {A, N, P,<s>}



$$E = E^{<branch>}_{<node>} \cup E^{<children>}_{<branch>} \cup E^{<branch>}_{<children>} \cup E^{<node>}_{<branch>},$$

$$E = E^{<s>}_{<init>} \cup E^{<children>}_{<s>} \cup E^{<branch>}_{<children>} \cup E^{<node>}_{<branch>}$$

Дерево разбора {A, N, P,<s>}