Обобщённый синтаксический анализ

Generalized parsing

Автор: Григорьев Семён Вячеславович
rsdpisuy@gmail.com

Математико-Механический факультет
Санкт-Петербургского государственного университета

21 ноября 2013г.

LR(k)-анализ

- Управляющие таблицы содержат не более одной инструкции в ячейке.
  - Можно считать, что ровно одно, если все пустые ячейки заполнить инструкцией *ERROR*
- А если впроцессе построения таблицы невозможно однозначно определить очередное действие?
  - Конфликты
    - Shift-Reduce
    - Reduce-Reduce

Конфликты на практике

- В большинстве практических ситуаций нам нужен единственный вывод данной строки в данной грамматике.

Конфликты на практике

- В большинстве практических ситуаций нам нужен единственный вывод данной строки в данной грамматике.
- Или факт его отсутствия.

Конфликты на практике

- Конфликты приводят к недетерминированному поведению распознователя.

Конфликты на практике

- Конфликты приводят к недетерминированному поведению распознователя.
- Для одной входной цепочки можно получить несколько выводов.

Конфликты на практике

- Конфликты приводят к недетерминированному поведению распознователя.
- Для одной входной цепочки можно получить несколько выводов.
- А на сколько это плохо?

Конфликты на практике

- Конфликты приводят к недетерминированному поведению распознователя.
- Для одной входной цепочки можно получить несколько выводов.
- А на сколько это плохо?
- А всегда ли нужен ровно один вывод?

Конфликты на практике

- Конфликты приводят к недетерминированному поведению распознователя.
- Для одной входной цепочки можно получить несколько выводов.
- А на сколько это плохо?
- А всегда ли нужен ровно один вывод?
- Какой из них “правильный”?

Конфликты и неоднозначности

Definition (Неоднозначная грамматика)

Будем называть формальную грамматику неоднозначной, если существует входная последовательность для которой может быть найдено более одного левостороннего вывода в этой грамматике.

Разрешение неоднозначностей

- “Аккуратный” способ.
  - Преобразрвания грамматики.
  - В некоторых случаях можно преобразовать грамматику языка так, чтобы она не содержала неоднозначностей.

Разрешение неоднозначностей

- “Аккуратный” способ.
  - Преобразрвания грамматики.
  - В некоторых случаях можно преобразовать грамматику языка так, чтобы она не содержала неоднозначностей.
- Практический способ.
  - Автоматический детерменированный выбор одного из вариатнов при построении таблиц.

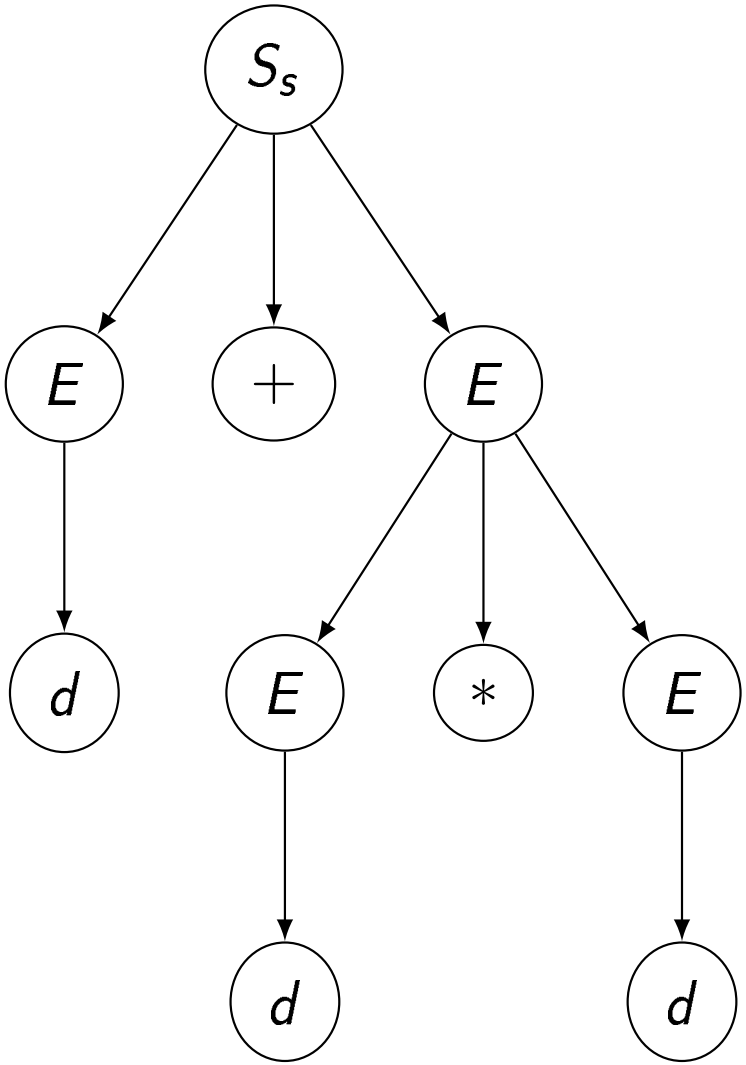
Разрешение неоднозначностей

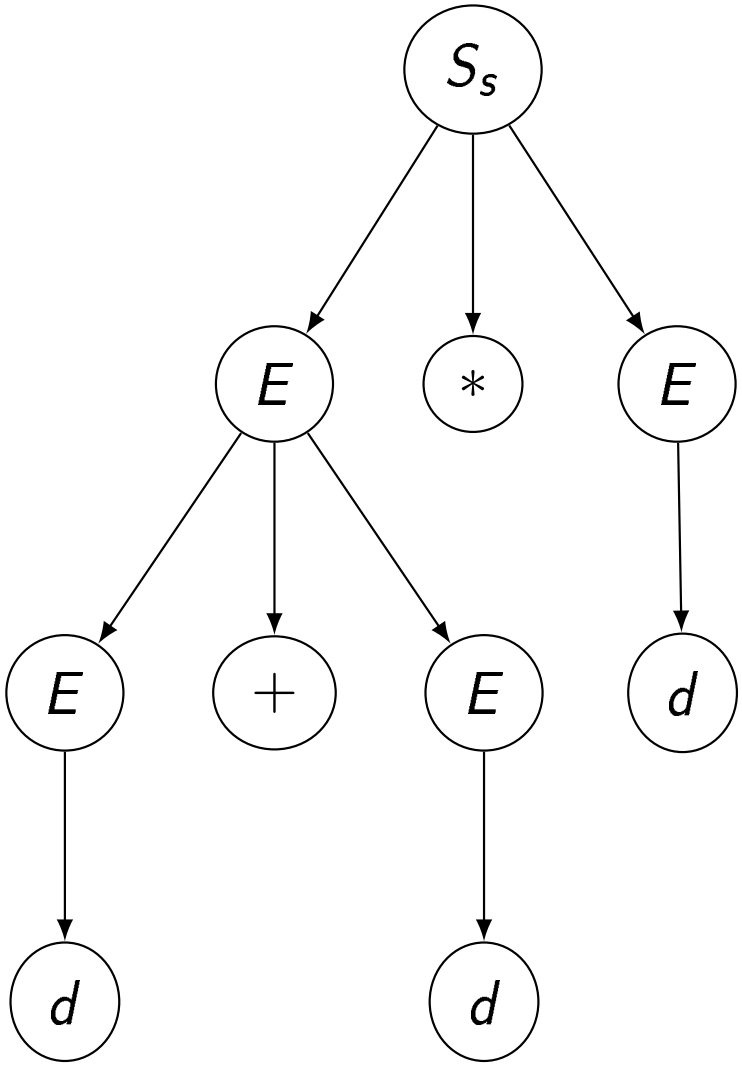
- “Аккуратный” способ.
  - Преобразрвания грамматики.
  - В некоторых случаях можно преобразовать грамматику языка так, чтобы она не содержала неоднозначностей.
- Практический способ.
  - Автоматический детерменированный выбор одного из вариатнов при построении таблиц.
  - Shift-Reduce $\rightarrow$ Shift
  - Reduce-Reduce $\rightarrow$ “magic first” Reduce

Пример преобразования грамматики

Оригинальная грамматика G_1	Преобразованная грамматика G_2
1. $S_s \to E\$$	1. $S_s \to S\$$
2. $E \to E + E$	2. $S \to S + T$
3. $E \to E * E$	3. $S \to T$
4. $E \to d$	4. $T \to T * F$
	5. $T \to F$
	6. $F \to d$

$d + d + d$ в грамматике G_1





Преобразования грамматики

- Не всегда удобны для применения на практике.
  - Не всегда просты.
  - Результирующая грамматика сложна.
  - Не все языки можно описать однозначными грамматиками.

Существенно неоднозначные языки

Definition (Существенно неоднозначный язык)

Язык L называется *существенно неоднозначным* если он может быть задан только неоднозначной грамматикой.

Существенно неоднозначные языки

Definition (Существенно неоднозначный язык)

Язык L называется *существенно неоднозначным* если он может быть задан только неоднозначной грамматикой.

Пример:

$L_1 = \{a^n b^m c^m d^n | n, m > 0\}$

$L_2 = \{a^n b^n c^m d^m | n, m > 0\}$

$L_3 = L_1 \cup L_2$

GLR

- А почему бы не разрешить обрабатывать все варианты?

GLR

- А почему бы не разрешить обрабатывать все варианты?
- Запсывать в ячейки наблицы больше одной инструкции.
- Во время анализа в момент возникновения неоднозначности дклать нужное количество копий стека и работать с каждым из них независимо.

GLR

- А почему бы не разрешить обрабатывать все варианты?
- Запсывать в ячейки наблицы больше одной инструкции.
- Во время анализа в момент возникновения неоднозначности дклать нужное количество копий стека и работать с каждым из них независимо.
- Много лишних копий данных.

Объединение общих частей стека

- Копирование появляется при возникновении конфликтов (Reduce/Reduce, Shift/Reduce).
- Его можно заменить ветвлением.

Объединение общих частей стека

- Копирование появляется при возникновении конфликтов (Reduce/Reduce, Shift/Reduce).
- Его можно заменить ветвлением.
- Если вершины двух веток соответствуют одному состоянию, то их можно объединить.

“Историческая сравка”

Идеи и алгоритмы известны достаточно давно.

- 1974. Bernard Lang. Не ризнали.
- 1984. Masaru Tomita. Книга в 200 стракиц.

Graph Structuref Stack (GSS)

- Стек можно организовать в виде графа.
- В общем случае граф произвольный.

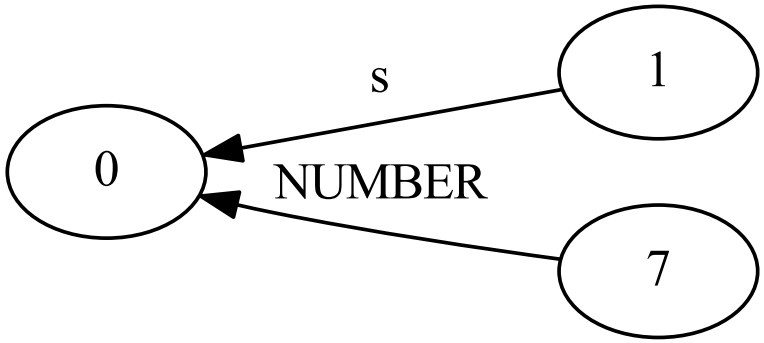
Пример

- Грамматика

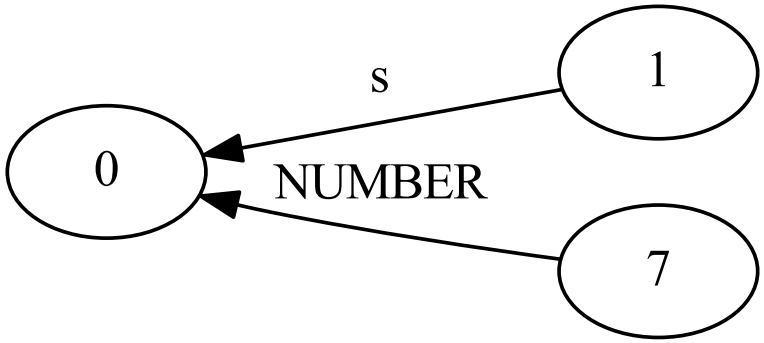
  `s: (s(+|*)s)|NUM`

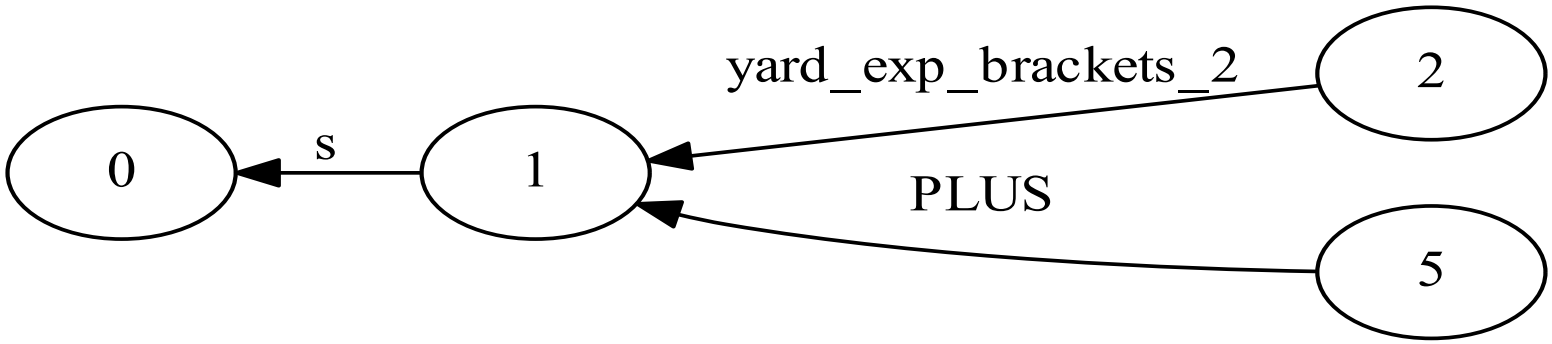
- Входная строка: NUM+NUM+NUM

Пример: GSS

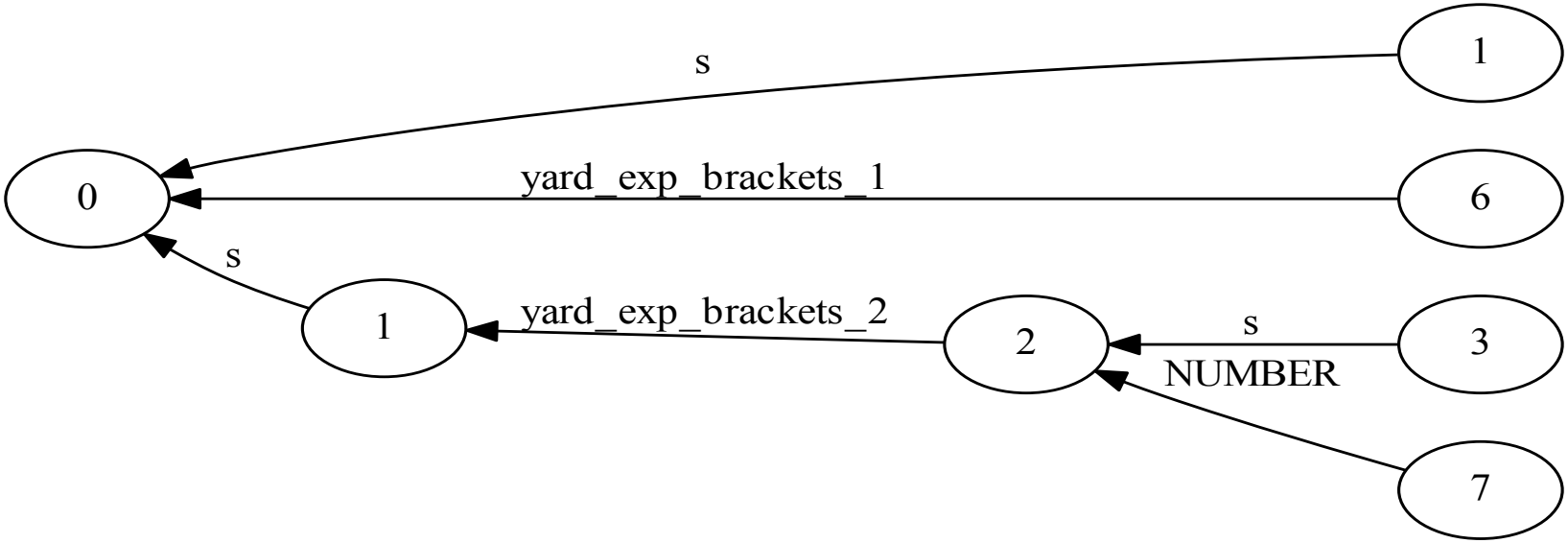


Пример: GSS

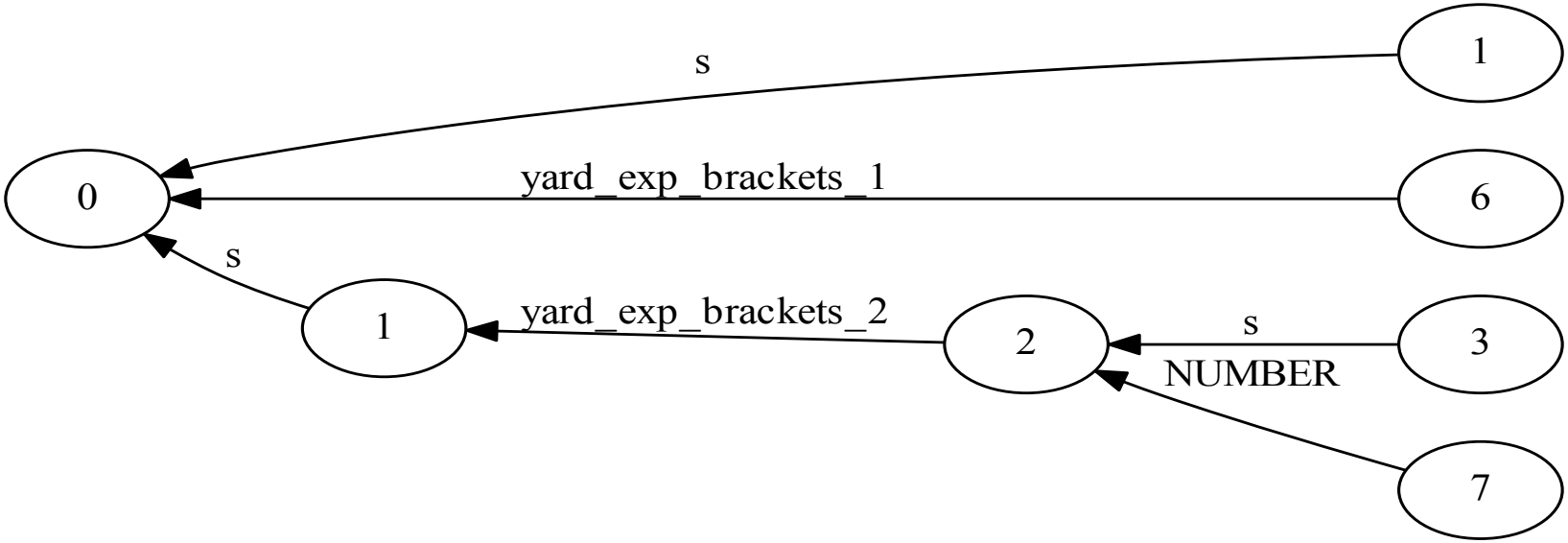


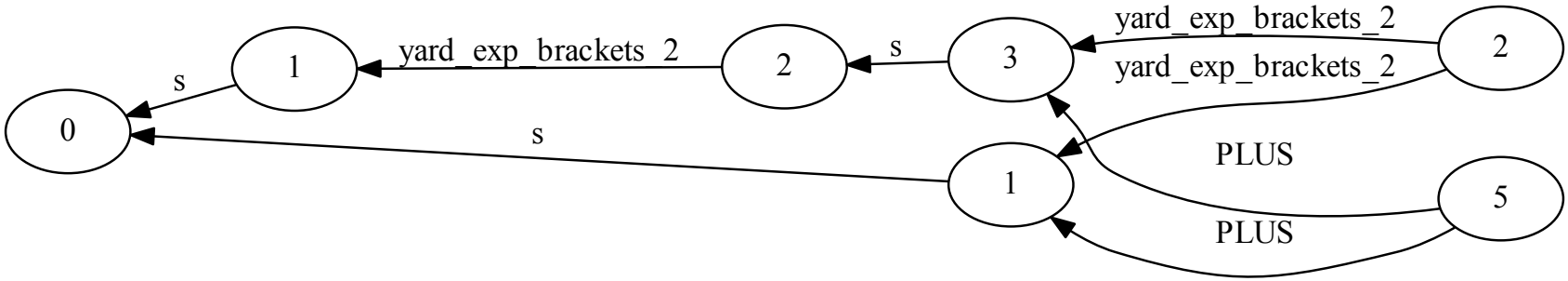


Пример: GSS

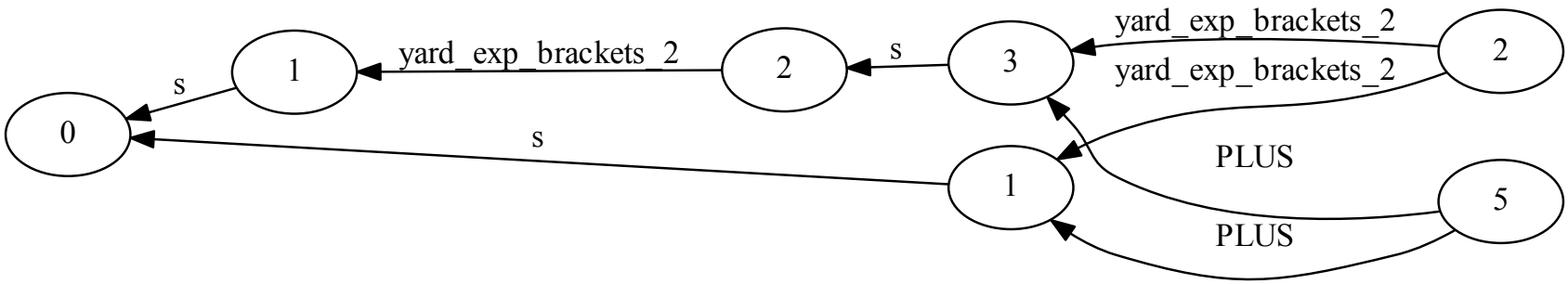


Пример: GSS

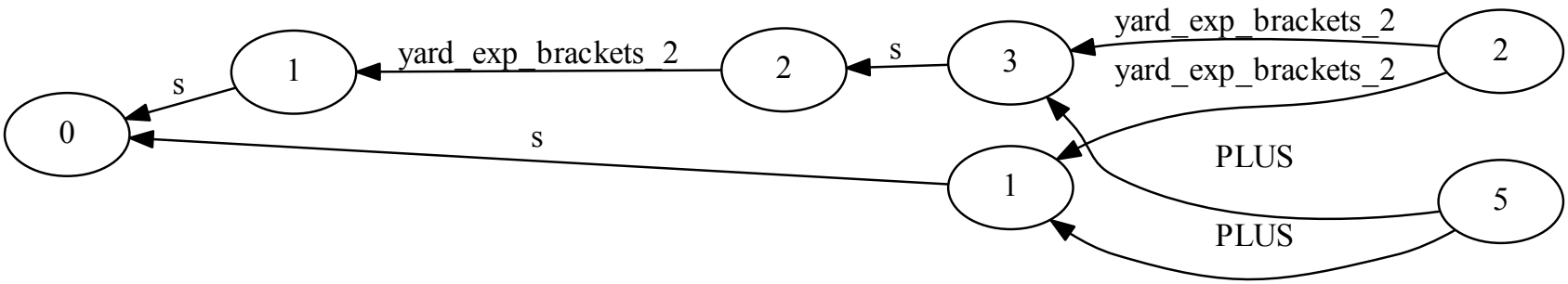

1
s
0
yard_exp_brackets_1
6
s
1
yard_exp_brackets_2
2
s
3
NUMBER
7

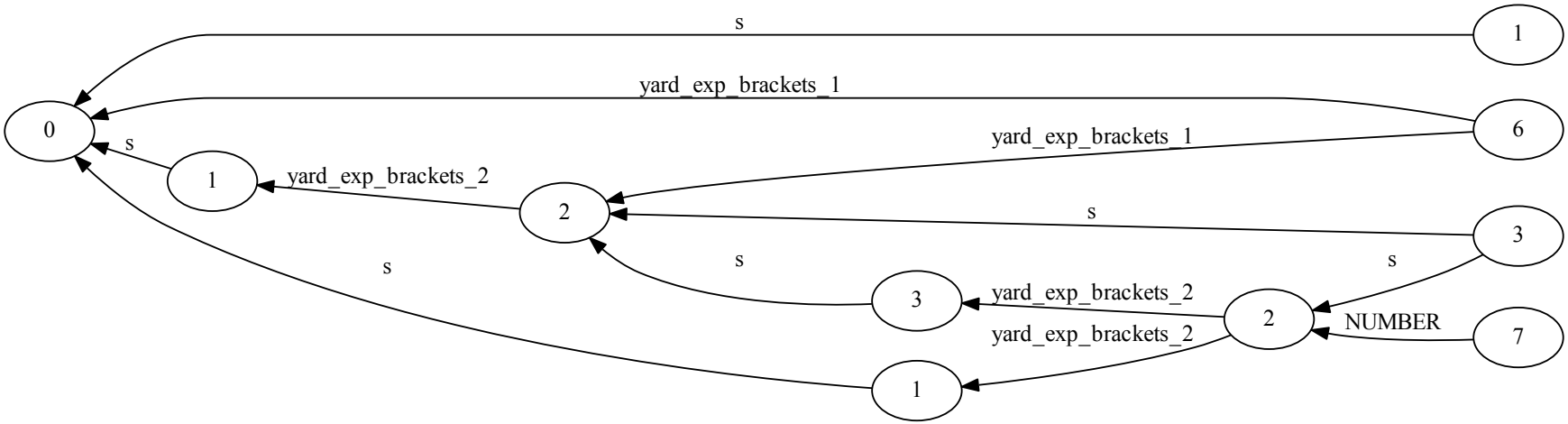

yard_exp_brackets_2
2
yard_exp_brackets_2
1
yard_exp_brackets_2
2
s
3
s
0
s
1
PLUS
5
PLUS

Пример: GSS


0
1
2
3
1
2
5
s
s
s
yard_exp_brackets_2
yard_exp_brackets_2
yard_exp_brackets_2
PLUS
PLUS

Пример: GSS


0
1
2
3
1
2
5
s
s
yard_exp_brackets_2
s
yard_exp_brackets_2
yard_exp_brackets_2
s
PLUS
PLUS


0
1
2
3
1
2
3
6
7
s
s
yard_exp_brackets_2
s
yard_exp_brackets_1
yard_exp_brackets_1
s
s
yard_exp_brackets_2
yard_exp_brackets_2
NUMBER

Обобщение GLR-анализа

Немного про то, когда нам могут понадобиться все варианты разбора.

Встроенные языки

- Динамический SQL

```
IF @X = @Y
    SET @TABLE = '#table1'
ELSE
    SET @TABLE = 'table2'
EXECUTE
    ('SELECT x FROM' + @TABLE + ' WHERE ISNULL(n,0) > 1')
```

- JavaScript в Java

```
String script =
    "function hello(name)  print('Hello, ' + name); ";
engine.eval(script);
Invocable inv = (Invocable) engine;
inv.invokeFunction("hello", "Scripting!!!" );
```

Проблема

- Ошибки в динамически формируемых выражениях обнаруживаются лишь во время выполнения.
- Динамически формируемые выражения – код на некотором языке. Его нужно соответствующим образом поддерживать и обрабатывать.
  - Поддержка в IDE.
  - Реинжиниринг ПО, разработанного с использованием встроенных языков.

Предлагаемое решение

- Статический анализ для встроенных языков. Абстрактный лексический и синтаксический анализы.
  - Поддержка в IDE:
    - ★ Многие ошибки можно искать без запуска программы.
    - ★ Автодополнение, рефакторинги.
    - ★ Возможны трансформации.
  - Реинжиниринг:
    - ★ Статический анализ.
    - ★ Трансформация (трансляция).

Как это работает

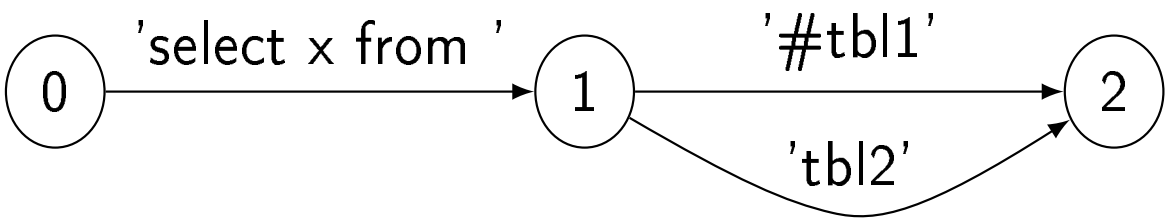
- Для каждого выражения строится конструкция, аппроксимирующая множество его возможных значений.
  - Data-flow уравнение
  - **Граф**
  - Регулярное выражение
- Выполнение лексического, синтаксического анализа над графом – абстрактный анализ.

Пример

- Код:

```
var tbl1 = “#tbl1”
var tbl2 = “tbl2”
execute (“select x from ” + if cond then tbl1 else tbl2)
```

- Входной “поток” для анализатора:



Shift-Shift псевдоконфликты

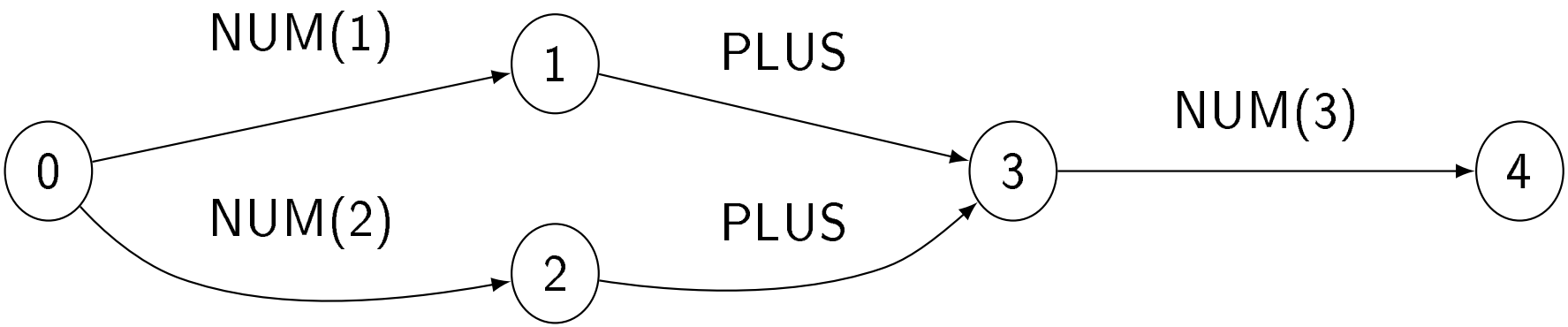
- Ветвление во входном потоке соответсвует возможности в текущем состоянии сделать несколько разных переносов (Shift).
- Конечно, это не конфликты в строгом смысле слова, но давайте по аалогии называть их Shift-Shift конфликтами.

Пример

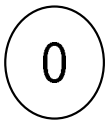
- Грамматика:

```
s: NUM PLUS e
e: NUM
```

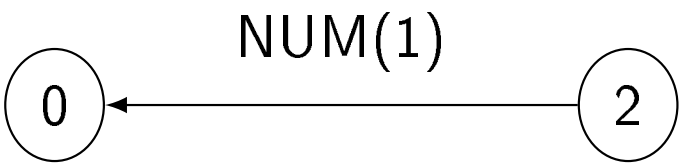
- Входной граф:



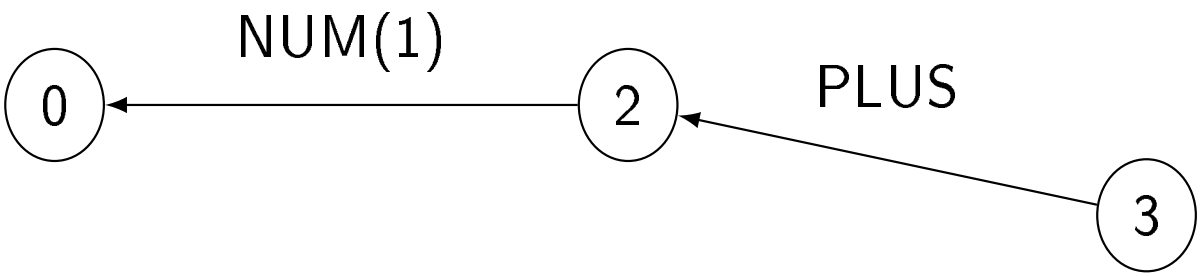
Пример: стек



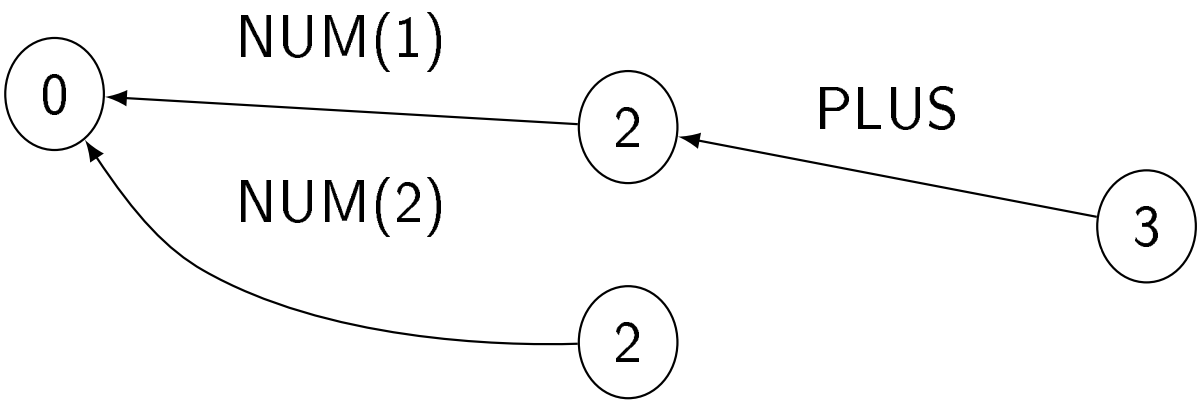
Пример: стек



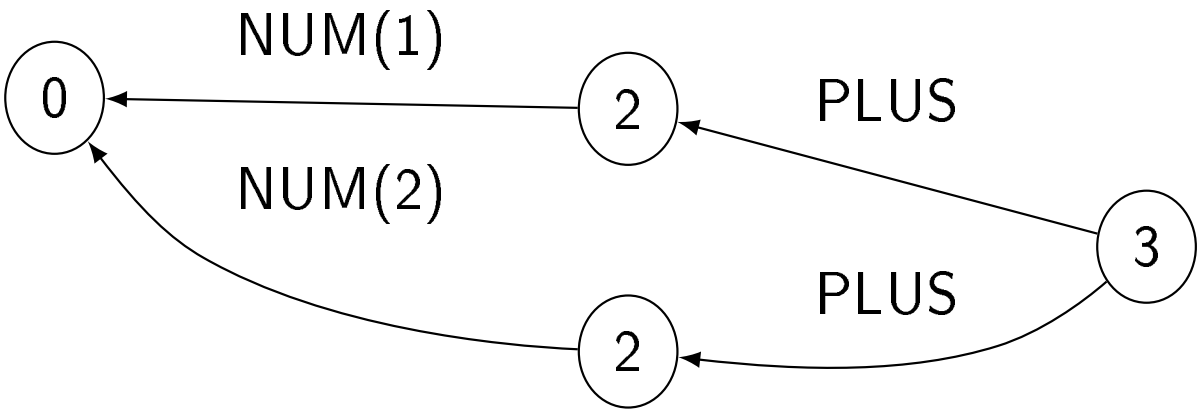
Пример: стек



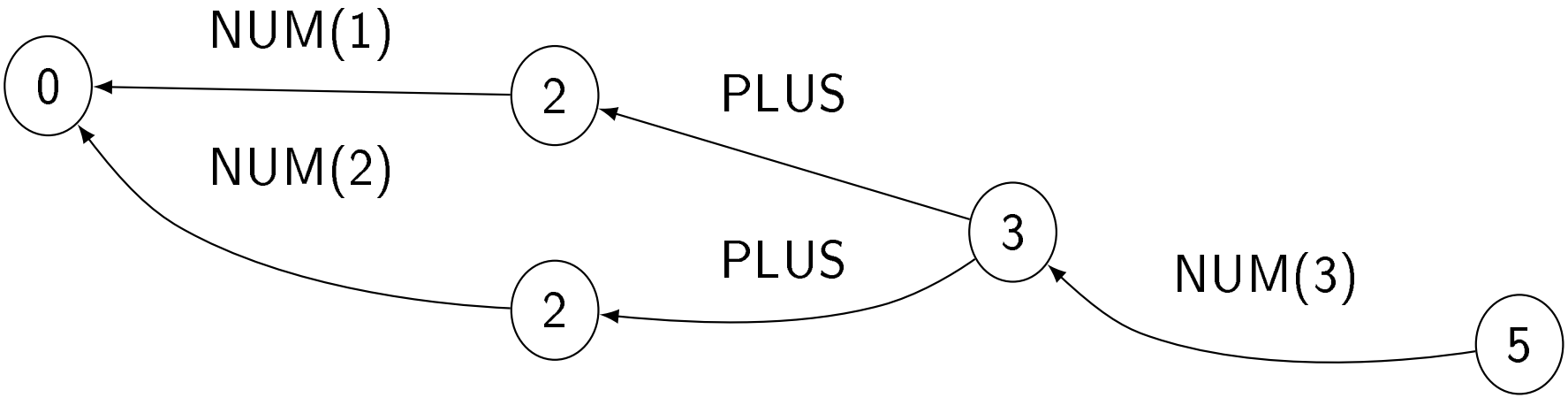
Пример: стек



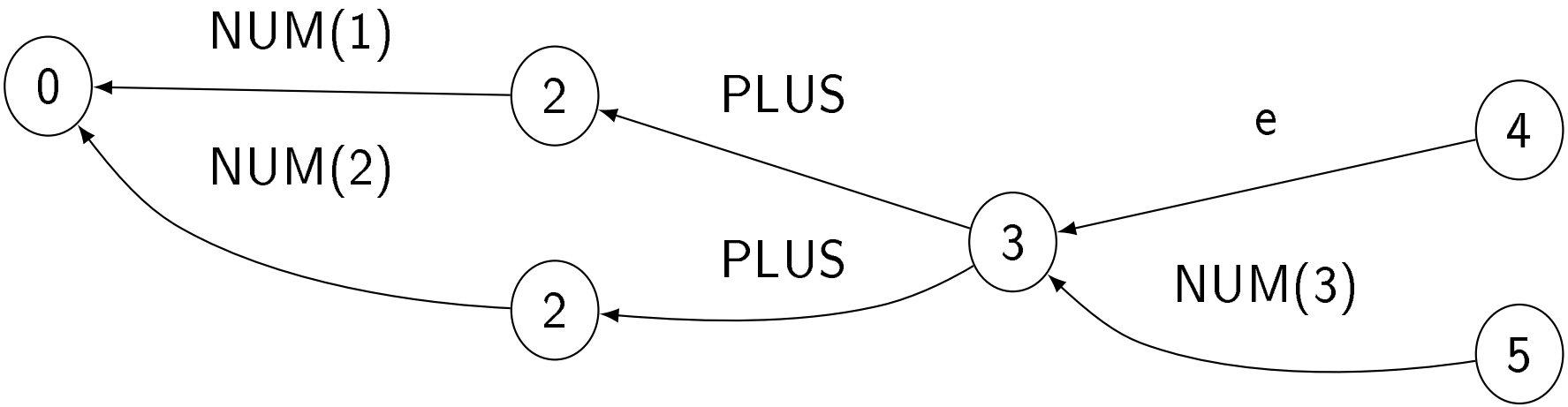
Пример: стек



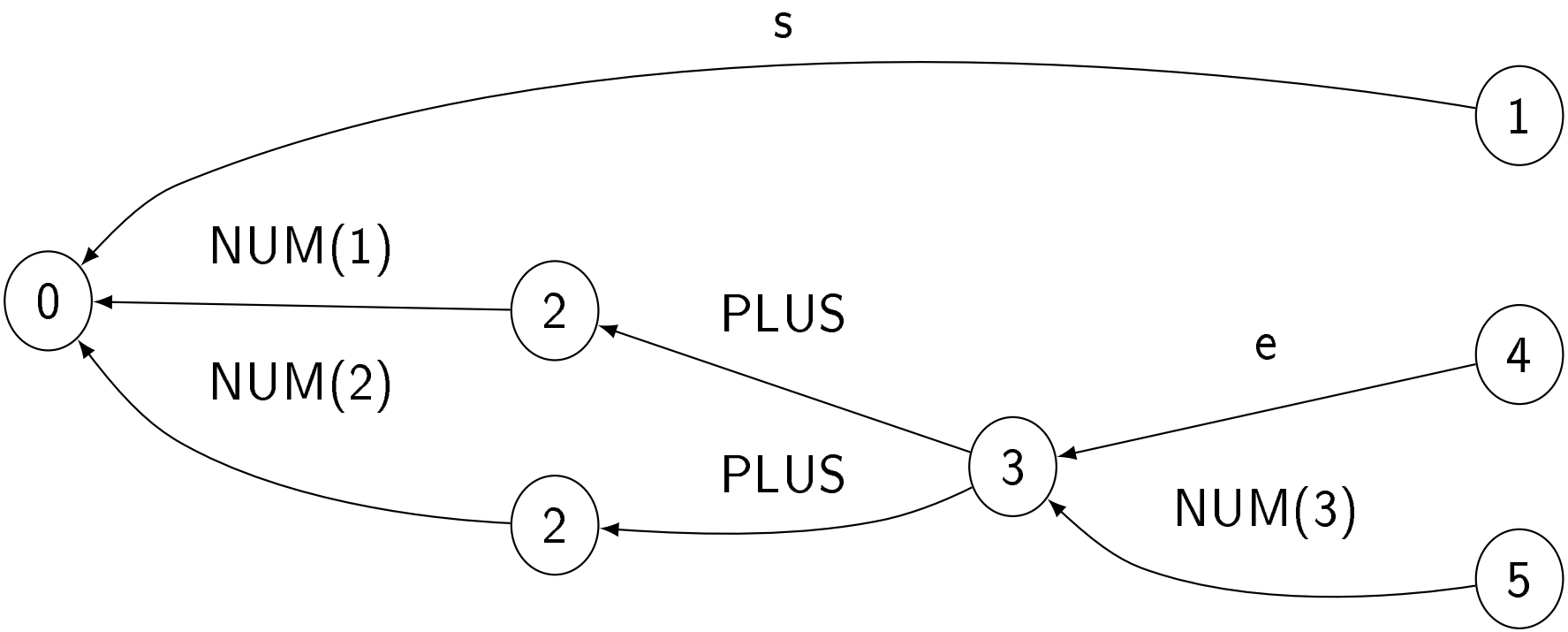
Пример: стек



Пример: стек



Пример: стек



Припер: результат разбора