Parsing Techniques for Contex-Free Path Querying

Semyon Grigorev
s.v.grigoriev@spbu.ru
Semen.Grigorev@jetbrains.com

JetBrains Research, Programming Languages and Tools Lab
Saint Petersburg University

April 05, 2019

Formal language constrained path querying

- Finite directed edge-laballed graph $\mathcal{G} = (V, E, L)$
- The path is a world over L:
  $\omega(p) = \omega(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \ldots \xrightarrow{l_{n-1}} v_n) = l_0 \cdot l_1 \cdot \ldots \cdot l_{n-1}$
- The language $\mathcal{L}$ (over L)

Formal language constrained path querying

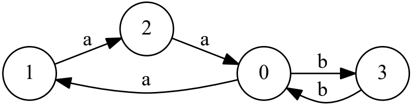
- Finite directed edge-laballed graph $\mathcal{G} = (V, E, L)$
- The path is a world over L:
  $\omega(p) = \omega(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \dots \xrightarrow{l_{n-1}} v_n) = l_0 \cdot l_1 \cdot \ldots \cdot l_{n-1}$
- The language $\mathcal{L}$ (over L)

- Reachability problem: $Q = \{(v_i, v_j) \mid \exists p = v_i \dots v_j, \omega(p) \in \mathcal{L}\}$
- Path querying problem: $Q = \{p \mid \omega(p) \in \mathcal{L}\}$
  - Single path, all paths, shortest path ...

Context-Free path querying

- $\mathcal{L}$ is a context-free language
- $G_{\mathcal{L}} = (N, \Sigma, R, S)$
- Reachability problem: $Q = \{(v_i, v_j) \mid \exists p = v_i \dots v_j, S \xrightarrow[G_L]{*} \omega(p)\}$
- Path querying problem: $Q = \{p \mid \omega(p) \in \mathcal{L}\}$

Example of CFPQ



Input graph

$0: \quad S \to a\ S\ b$

$1: \quad S \to Middle$

$2: \quad Middle \to a\ b$

Query: language $\{a^n b^n \mid n > 0\}$

Paths:

$2 \xrightarrow{a} 0 \xrightarrow{b} 3$

$1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0$

$p_1 = 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3$

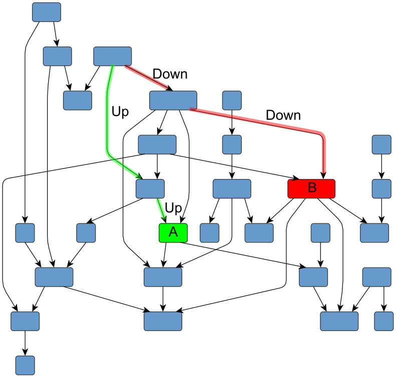
$p_2 = 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3 \xrightarrow{b} 0$

$\dots$

Applications

- Graph data bases querying
  Yann ...
- Static code analysis
  Reps CFL reachability
- CFL editing distance/Error recovery
  Aho

Graph data bases querying

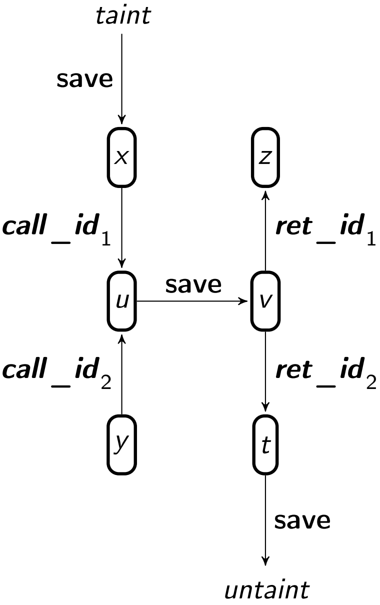


Navigation through a graph

- Are nodes A and B on the same level of hierarchy?
- Is there a path of form **Up**^{n} **Down**^{n}?
- Find all paths of form **Up**^{n} **Down**^{n} which start from the node A

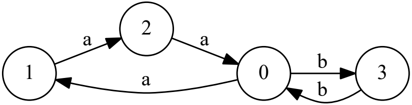
Static code analysis

```
int id(int u)
{
  v = u;
  return v;
}
int main()
{
  //taint
  int x;
  int z, y;
  //untaint
  int t;
  z = id(x);
  t = id(y);
}
```



Error recovery

- !!!!



Input graph

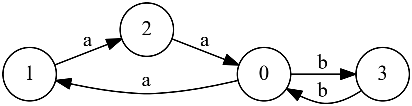
$0: \quad S \to a\ S\ b$

$1: \quad S \to Middle$

$2: \quad Middle \to a\ b$

Grammar

Structural representation of result



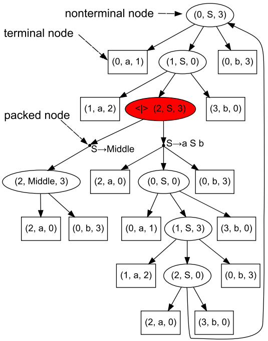
Input graph

$0: \quad S \to a \ S \ b$

$1: \quad S \to Middle$

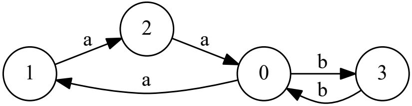
$2: \quad Middle \to a \ b$

Grammar



Query result (SPPF)

Structural representation of result



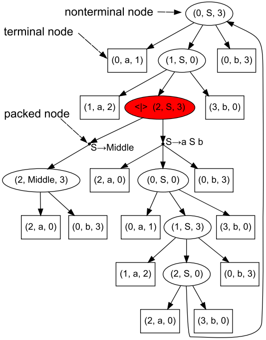
Input graph

$0: \quad S \to a\ S\ b$

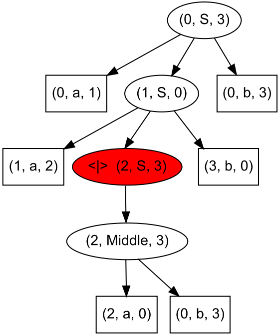
$1: \quad S \to Middle$

$2: \quad Middle \to a\ b$

Grammar

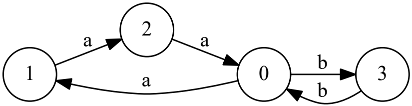


Query result (SPPF)



Tree for p_1

Structural representation of result



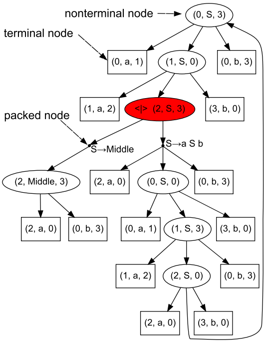
Input graph

$0: \quad S \to a\ S\ b$

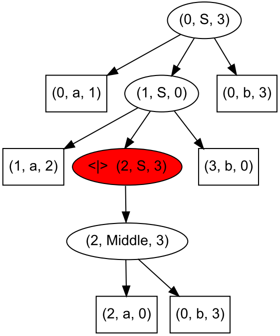
$1: \quad S \to Middle$

$2: \quad Middle \to a\ b$

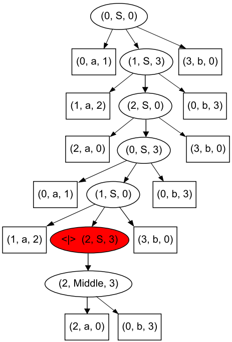
Grammar



Query result (SPPF)

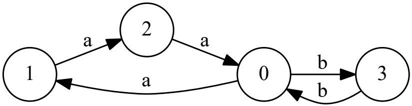


Tree for p_1

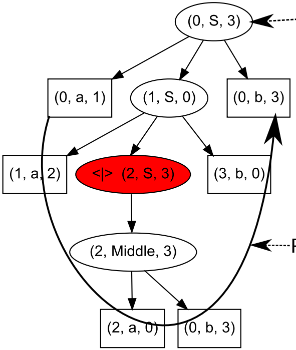


Tree for p_2

Paths extraction



$$
\begin{aligned}
0: &\quad S \to a\ S\ b\\
1: &\quad S \to Middle\\
2: &\quad Middle \to a\ b
\end{aligned}
$$



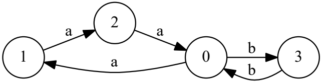
There exists a path p from vertex 0 to vetex 3 such that $\omega(p)$ is derivable from S. Subtree which is rooted by this node is a derivation tree of $\omega(p)$.

Path is a seqence of leafs

Path: $0 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{a} 0 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{b} 3$

Bar-Hillel theorem

Context-free languages are closed under intersection with regular languages



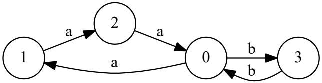
Regular language

$$\begin{array}{ll} 0: & S \to a\ S\ b \\ 1: & S \to Middle \\ 2: & Middle \to a\ b \end{array}$$

Context-free language

Bar-Hillel theorem

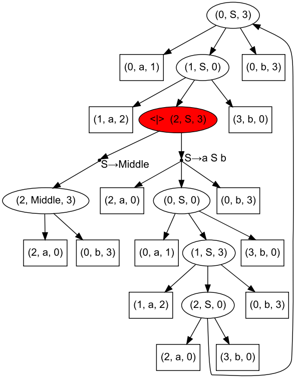
Context-free languages are closed under intersection with regular languages



Regular language

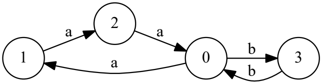
$$
\begin{aligned}
0: &\quad S \to a\ S\ b \\
1: &\quad S \to Middle \\
2: &\quad Middle \to a\ b
\end{aligned}
$$

Context-free language



Bar-Hillel theorem

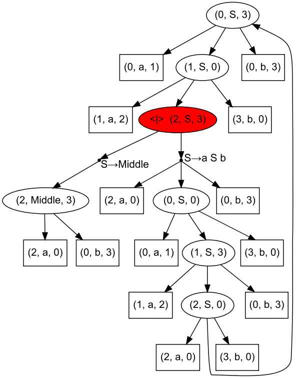
Context-free languages are closed under intersection with regular languages



Regular language

$$
\begin{array}{ll}
0: & S \to a\ S\ b \\
1: & S \to Middle \\
2: & Middle \to a\ b
\end{array}
$$

Context-free language



$$
\begin{array}{rl}
(0, S, 3) & \to (0, a, 1)\ (1, S, 0)\ (0, b, 3) \\
(1, S, 0) & \to (1, a, 2)\ (2, S, 3)\ (3, b, 0) \\
(2, S, 3) & \to (2, a, 0)\ (0, S, 0)\ (0, b, 3) \\
(2, S, 3) & \to (2, Middle, 3) \\
(0, S, 0) & \to (0, a, 1)\ (1, S, 3)\ (3, b, 0) \\
(1, S, 3) & \to (1, a, 2)\ (2, S, 0)\ (0, b, 3) \\
(2, S, 0) & \to (2, a, 0)\ (0, S, 3)\ (3, b, 0) \\
(0, Middle, 3) & \to (2, a, 0)\ (0, b, 3)
\end{array}
$$

Our experiments

- Generalized LR for CFPQ
  - Based on Right Nulled Generalized LR: *Scott E., Johnstone A.* "Right Nulled GLR Parsers"
  - *Ekaterina Verbitskaia, Semyon Grigorev, and Dmitry Avdyukhin.* "Relaxed Parsing of Regular Approximations of String-Embedded Languages" 2015

Our experiments

- Generalized LR for CFPQ
  - Based on Right Nulled Generalized LR: *Scott E., Johnstone A.* "Right Nulled GLR Parsers"
  - *Ekaterina Verbitskaia, Semyon Grigorev, and Dmitry Avdyukhin.* "Relaxed Parsing of Regular Approximations of String-Embedded Languages" 2015
- Generalized LL for CFPQ (**GLL**)
  - Based on Generalized LL: *Scott E., Johnstone A.* "GLL parsing"
  - *Semyon Grigorev and Anastasiya Ragozina.* "Context-free path querying with structural representation of result." 2017

Query language integration

How to integrate query language into general-purpose programming language?

- Transparency
- Compositionality
- Static error checking

Query language integration

How to integrate query language into general-purpose programming language?

- Transparency
- Compositionality
- Static error checking

- String-embedded languages
- ORMs
- Combinators

Combinators for CFPQ

- Implemented in Scala
- Based on Meerkat parser combinator library: *Anastasia Izmaylova, Ali Afroozeh, and Tijs van der Storm.* "Practical, general parser combinators" 2016
- *Ekaterina Verbitskaia, Ilya Kirillov, Ilya Nozkin, Semyon Grigorev.* "Parser Combinators for Context-Free Path Querying" 2019

Supported combinators

Combinator	Description
a ~ b	sequential parsing: a then b
a | b	choice: a or b

Supported combinators

Combinator	Description
a ~ b	sequential parsing: a then b
a | b	choice: a or b
a ?	optional parsing: a or nothing
a *	repetition of zero or more a
a +	repetition of at least one a

Supported combinators

Combinator	Description
a ~ b	sequential parsing: a then b
a | b	choice: a or b
a ?	optional parsing: a or nothing
a *	repetition of zero or more a
a +	repetition of at least one a
a ^ f	apply f function to a if a is a token
a ^^	capture output of a if a is a token
a & f	apply f function to a if a is a parser
a &&	capture output of a if a is a parser

A set of functions for edges and vertices values handling.

```
def LV(labels: String*) =
  V(e => labels.forall(e.hasLabel))
def outLE(label:String) = outE(_.label() == label)
def inLE (label:String) = inE (_.label() == label)
```

Basic example

Is there a path from vertex 0 to vertex 3 which has form $a^n b^n$?

```
val Query : Nonterminal
    = syn (LV("0") ~ S ~ LV("3"))

val S: Nonterminal
    = syn ( "a" ~ S ~ "b"
          | "a" ~ "b"
          )
```

Example of generalization

```
def sameGen(brs) =
  reduceChoice(
    brs.map {case (lbr, rbr) =>
      lbr ~ syn(sameGen(brs).?) ~ rbr})
```

Example of generalization

```
def sameGen(brs) =
  reduceChoice(
    brs.map {case (lbr, rbr) =>
      lbr ~ syn(sameGen(brs).?) ~ rbr})

val query1 = syn(sameGen(List(("a", "b"))))

val query2 = syn(
  sameGen(List((p1, p2),("(",")"))) ~ p3)
```

Example of values handling

Actors who played in some film

In Cypher

```
MATCH (m:Movie {title: 'Forrest Gump'})
        <-[:ACTS_IN]-(a:Actor)
RETURN a.name, a.birthplace;
```

In Meerkat

```
val query =
  syn((
    (LV("Movie")::V(_.title == "Forrest Gump")) ~
    inLE("ACTS_IN") ~
    syn(LV("Actor") ^
          (e => (e.name, e.birthplace)))) &&)
executeQuery(query, input)
```

Limitations

- Overhead for the regular constraints
- Not exactly clear how to compute arbitrary semantics for the paths
  - Paths can be lazily extracted, but in what order?
  - Is it possible to compute some semantics in case of cycles?

The algorithm

Algorithm Context-free recognizer for graphs

```
function ContextFreePathQuerying(D, G)
    n ← the number of nodes in D
    E ← the directed edge-relation from D
    P ← the set of production rules in G
    T ← the matrix n × n in which each element is ∅
    for all (i, x, j) ∈ E do                          ▷ Matrix initialization
        T_{i,j} ← T_{i,j} ∪ {A | (A → x) ∈ P}
    while matrix T is changing do
        T ← T ∪ (T × T)                     ▷ Transitive closure T^{cf} calculation
    return T
```

Transitive Closure

- Subset multiplication, $N_1, N_2 \subseteq N$
  - $N_1 \cdot N_2 = \{A \mid \exists B \in N_1, \exists C \in N_2$ such that $(A \to BC) \in P\}$
- Subset addition: set-theoretic union.
- Matrix multiplication
  - Matrix of size $|V| \times |V|$
  - Subsets of N are elements
  - $c_{i,j} = \bigcup_{k=1}^{n} a_{i,k} \cdot b_{k,j}$
- Transitive closure
  - $a^{cf} = a^{(1)} \cup a^{(2)} \cup \cdots$
  - $a^{(1)} = a$
  - $a^{(i)} = a^{(i-1)} \cup (a^{(i-1)} \times a^{(i-1)}),\ i \geq 2$

Performance comparison setup

We use graphs from the classical set of ontologies: *skos, foaf, univ-bench, wine, pizza, etc.*

Queries are classical variants of the same-generation query

$$
\begin{array}{ll}
0: & \mathbf{S} \to subClassOf^{-1}\ \mathbf{S}\ subClassOf \\
1: & \mathbf{S} \to type^{-1}\ \mathbf{S}\ type \\
2: & \mathbf{S} \to subClassOf^{-1}\ subClassOf \\
3: & \mathbf{S} \to type^{-1}\ type
\end{array}
$$

Query 1

$$
\begin{array}{ll}
0: & \mathbf{S} \to \mathbf{B}\ subClassOf \\
1: & \mathbf{S} \to subClassOf \\
2: & \mathbf{B} \to subClassOf^{-1}\ \mathbf{B}\ subClassOf \\
3: & \mathbf{B} \to subClassOf^{-1}\ subClassOf
\end{array}
$$

Query 2

Performance comparison results

№	#V	#E	Query 1 (ms)			Query 2 (ms)	
			CYK[1]	GLL	GPGPU	GLL	GPGPU
1	144	323	1044	10	12	1	1
2	129	351	6091	19	13	1	0
3	131	397	13971	24	30	1	10
4	179	413	20981	25	15	11	9
5	337	834	82081	89	32	3	6
6	291	685	515285	255	22	66	2
7	341	711	420604	261	20	45	24
8	671	2604	3233587	697	24	29	23
9	733	2450	4075319	819	54	8	6
10	6224	11840	–	1926	82	167	38
11	5864	19600	–	6246	185	46	21
12	5368	20832	–	7014	127	393	40

[1] Zhang, et al. "Context-free path queries on RDF graphs."

Directions for research

- Parallel and distributed parsing
- O(BMM) complexity
- Incremental parsing