Formal Theory of Context-Free Grammars

Second Bachelor Seminar Talk

Jana Hofmann
Advisor: Prof. Dr. Gert Smolka

SAARLAND UNIVERSITY
COMPUTER SCIENCE

29th April 2016

What is a Context-Free Grammar?

Example

$$A \setminus aAc$$
$$A \setminus aBc$$
$$B \setminus BB$$
$$B \setminus b$$

- Describe context-free languages
  e.g. $\{a^n b^m c^n \mid n > 0\}$
- Are used to describe (programming) languages

Important Terms

Example

$A \setminus aAc$

$A \setminus aBc$

$B \setminus BB$

$B \setminus b$

- **grammar** G consists of:
  - **symbols** s
    terminals $a,\ b,\ c, \ldots$
    variables $A,\ B,\ C, \ldots$
  - **phrases** $u,\ v,\ w, \ldots$
  - **rules** $A \setminus u$
- **Words** are phrases containing only terminals
- From A one can **derive** a word by rewriting the rules of the grammar
- A **language** of a grammar is the set of all words we can derive starting with a variable ($\mathcal{L}_G^A$ or $\mathcal{L}_G^B$)

Important Terms

Example

$$A \setminus aAc$$
$$A \setminus aBc$$
$$B \setminus BB$$
$$B \setminus b$$

Example

Derivation of *aaabccc*

$$\begin{aligned} A &\rightsquigarrow aAc && A \setminus aAc \in G \\ &\rightsquigarrow aaAcc && A \setminus aAc \in G \\ &\rightsquigarrow aaaBccc && A \setminus aBc \in G \\ &\rightsquigarrow aaabccc && B \setminus b \in G \end{aligned}$$

Interesting Questions

1. Empty word problem: $\varepsilon \in \mathcal{L}_G^A$?
2. Empty language problem: $\mathcal{L}_G^A = \emptyset$?
3. Word problem: $w \in \mathcal{L}_G^A$?
4. Finiteness: Is $\mathcal{L}_G^A$ finite?

} decidable

5. Equality problem: $\mathcal{L}_G^A = \mathcal{L}_{G'}^{A'}$?
6. Regularity: Is $\mathcal{L}_G^A$ regular?

} undecidable

Closure Properties

Let C_1, C_2 be a context-free and R a regular language

- $C_1 \cup C_2$ is context-free
- $C_1 \cap C_2$ is in general not context-free
- $C_1 \cup R$ is context-free
- $\overline{C_1}$ is not context-free

Normal Forms

Chomsky Normal Form

All rules must be of one of the following forms:

- $A \setminus a$
- $B \setminus CD$
- $S \setminus \varepsilon \mid E$
  where S is the start symbol and $S \neq A,\ B, \ldots$

All grammars can be transformed into CNF

Serves as basis for CYK-algorithm to decide the word problem

Normal Forms

Greibach Normal Form

All rules must be of the following form:

- $A \setminus aB_1B_2 \dots B_n$

$\Rightarrow \varepsilon \notin \mathcal{L}_G$

All ε-free grammars can be transformed into Greibach normal form

Advantage: one new terminal introduced in each derivation step
$\Rightarrow$ Decidability of word problem becomes relatively intuitive

The Word Problem

Cocke-Younger-Kasami Algorithm

- Needs the grammar to be in Chomsky normal form
- Bottom-up parsing algorithm using dynamic programming
- In $\mathcal{O}(n^3)$

Earley Algorithm

- Requires the grammar to be ε-free
- Top-down parsing algorithm using dynamic programming
- In $\mathcal{O}(n^3)$

Previous Work

Denis Firsov and Tarmo Uustalu
Certified Normalization of Context-Free Grammars
Institute of Cybernetics at TUT, 2015

Denis Firsov and Tarmo Uustalu
Certified CYK parsing of context-free languages
Journal of Logical and Algebraic Methods in Programming 83.5 (2014): 459-468

Maksym Bortin
A formalisation of the Cocke-Younger-Kasami algorithm
Archive of Formal Proofs (2016, April)

Where Do We Start?

- Greibach normal form requires $\mathcal{L}_G$ to be ε-free
- Chomsky normal form includes transformation to ε-freeness
- Earley algorithm requires $\mathcal{L}_G$ to be ε-free

$\Rightarrow$ Compute G^- s.t. $\mathcal{L}_G^A \backslash \{\varepsilon\} \equiv \mathcal{L}_{G^-}^A$

Therefore decide $\varepsilon \in \mathcal{L}_G^A$

Definitions

We use lists for grammars and derivations

$$
\begin{array}{lcll}
\mathit{var} & := & n & n \in \mathbb{N} \\
\mathit{ter} & := & n & n \in \mathbb{N} \\
\mathit{symbol} & := & \mathit{var} \mid \mathit{ter} &
\end{array}
$$

$$
\begin{array}{lcl}
\mathit{phrase} & := & \mathscr{L}(\mathit{symbol}) \\
\mathit{rule} & := & \mathit{var} \times \mathit{phrase} \\
\mathit{grammar} & := & \mathscr{L}(\mathit{rule})
\end{array}
$$

Definitions

We use lists for grammars and derivations

The notion of derivability can be defined inductively:

$$\frac{}{A \overset{G}{\Rightarrow} A} \qquad \frac{A \setminus u \in G}{A \overset{G}{\Rightarrow} u} \qquad \frac{A \overset{G}{\Rightarrow} uBw \quad B \overset{G}{\Rightarrow} v}{A \overset{G}{\Rightarrow} uvw}$$

$$\mathcal{L}_G^A \coloneqq \lambda\ w.\ A \overset{G}{\Rightarrow} w\ \wedge\ w \textit{ is a word}$$

$\varepsilon \in \mathcal{L}_G^A$?

Task: Find all variables that can derive ε (we call this **nullable**)

We define $nullable_G\ A$ inductively:

$$\frac{\exists\ u.\ A \setminus u \in G \wedge \forall s \in u.\ nullable_G\ s}{nullable_G\ A}$$

Lemma

$nullable_G\ A \leftrightarrow A \overset{G}{\Rightarrow} \varepsilon$

Nullable Variables

$$\frac{\exists\ u.\ A \setminus u \in G \wedge \forall s \in u.\ \mathit{nullable}_G\ s}{\mathit{nullable}_G\ A}$$

Computing the set of nullable variables works parallel to inductive definition:

Example

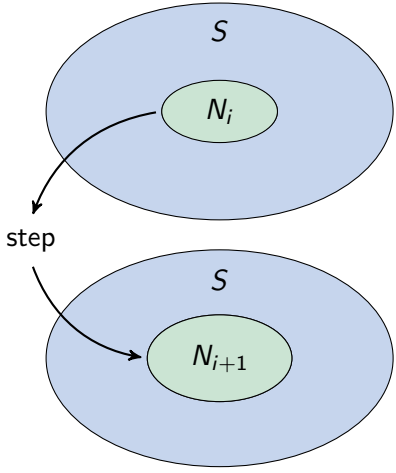
$A \setminus a \mid \varepsilon$
$B \setminus AA$
$C \setminus ABD$
$D \setminus \varepsilon$

- A, D are nullable
- A, D, B are nullable
- A, D, B, C are nullable

$\Rightarrow$ Fixpoint Algorithm!

Finite Closure Iteration

ICL 2014 : general and verified algorithm to compute fixed points on lists
Task: Given step-predicate *step*, find N, s.t. $step\ N \equiv step\ (step\ N)$



To show:

$\forall s \in N_i.\ p\ s$

$\forall s \in N_{i+1}.\ p\ s$

Finite Closure Iteration

Get: list N s.t.

1. **N is step-closed**: If $step\ N\ s$, and $s \in S$, then $s \in N$
2. p **holds**: For all s in N, $p\ s$

Instantiate:

S := list of all variables in G
$step$:= $\lambda\ N_i\ A.\ \exists u.\ A \setminus u \in G\ \wedge\ \forall s \in u.\ s \in N_i$
p := $nullable_G$

Lemma
$nullable_G\ s \leftrightarrow s \in N$
$\leftarrow$: exact 2)
$\rightarrow$: by induction on $nullable_G$ using 1)

ε-Elimination

Example

$A \setminus aBc$

$B \setminus b$

$B \setminus \varepsilon$

$A \setminus ac$

We can not just drop all rules $A \setminus \varepsilon$

Solution: Add rules compensating this problem

$\Rightarrow$ Add rules where *some nullable variables* are dropped

Our approach:

1. Compute extended grammar
2. Drop all rules $A \setminus \varepsilon$

ε-Closure

Dropping variables can be formalized with an inductive predicate **sublist**:

$$\frac{}{\varepsilon \precsim_p \varepsilon} \qquad \frac{v \precsim_p u \quad p\ s}{v \precsim_p su} \qquad \frac{v \precsim_p u}{sv \precsim_p su}$$

- can be characterized by a generalized power function

$$\begin{aligned} &ppower\ p\ [] &&:= [[]] \\ &ppower\ p\ (s::u) &&:= \texttt{if}\ p\ s\ \texttt{then}\ ppower\ p\ u +\!\!+ map\ (cons\ s)\ (ppower\ p\ u) \\ &&&\quad \texttt{else}\ map\ (cons\ s)\ (ppower\ p\ u) \end{aligned}$$

- equivalent to normal power function if $p := \lambda s.\ \top$
- $\mathcal{E}_G := \ulcorner ppower\ nullable_G\ G \urcorner$

ε-Closure

$\mathcal{E}_G := \ulcorner ppower\ nullable_G\ G \urcorner$

We can show:

- $\mathcal{E}_G$ is **ε-closed**
  ε-closed $:= A \setminus u \in G \to v \precsim_{nullable_G} u \to A \setminus v \in G$
- $\mathcal{L}_G^A \equiv \mathcal{L}_{\mathcal{E}_G}^A$

Removal of ε-Rules

$G^- :=$ G without rules of the form $A \setminus \varepsilon$

Assume G is ε-closed. We want to show: $\mathcal{L}_G^A \setminus \{\varepsilon\} \equiv \mathcal{L}_{G^-}^A$

We observe: An ε-closed grammar is also **ε-derivation-closed** :
ε-derivation-closed $:= A \overset{G}{\Rightarrow} u \to A \neq u \to v \precsim_{nullable_G} u \to A \overset{G}{\Rightarrow} v$

Removal of ε-Rules

$\mathcal{L}_G \setminus \{\varepsilon\} \supseteq \mathcal{L}_{G^-}$

Easy, since $G \supseteq G^-$

$\mathcal{L}_G \setminus \{\varepsilon\} \subseteq \mathcal{L}_{G^-}$

Use derivation-closedness

Example

We derive in G: $\rightarrow$ We derive in G^-:

$A \overset{G}{\Rightarrow} uBv$ $\quad$ $A \overset{G^-}{\Longrightarrow} uBv$ $\quad$ *Induction*

$A \overset{G}{\Rightarrow} uv$ $\quad$ $B \setminus \varepsilon \in G$ $\quad$ $A \overset{G^-}{\Longrightarrow} uv$ $\quad$ $uv \precsim_{nullable_G} uBv$

Conclusion

What you saw

- $\varepsilon \in \mathcal{L}_G^A$ is decidable with an easy fixed point iteration
- Construction of an ε-free grammar with $\mathcal{L}_G \setminus \{\varepsilon\} \equiv \mathcal{L}_{G^-}$
- All proofs done in about 600 lines

Future Work

- Many interesting facts about context-free grammars
- Complete transformation to Chomsky normal form
- Decide word problem
- Decide $\mathcal{L}_G^A = \emptyset$

Sources


Dexter C. Kozen
Automata and Computability
Springer, 1997


John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman
Introduction to Automata Theory, Languages, and Computation
AddisonWesley, 2nd edition, 2001


Gert Smolka and Chad E. Brown
Introduction to Computational Logic
Lecture Notes [PDF], 2014. Retrieved from
https://www.ps.uni-saarland.de/courses/cl-ss14/script/icl.pdf

Generalized Derivation Predicate

Counting steps:

$$\frac{}{A \overset{G}{\Rightarrow}^0 A} \qquad \frac{A \overset{G}{\Rightarrow}^n uBw \quad B \setminus v \in G}{A \overset{G}{\Rightarrow}^{n+1} uvw}$$

Derive phrases:

$$\frac{}{u \overset{G}{\Rightarrow}_{\mathcal{T}} u} \qquad \frac{A \setminus u \in G}{A \overset{G}{\Rightarrow}_{\mathcal{T}} u} \qquad \frac{x \overset{G}{\Rightarrow}_{\mathcal{T}} uvw \quad v \overset{G}{\Rightarrow}_{\mathcal{T}} v'}{x \overset{G}{\Rightarrow}_{\mathcal{T}} uv'w}$$

Properties of Sublists

1. $u \precsim_p u$ (Reflexivity)
2. If $v \precsim_p u$ and $u \precsim_p w$, then $v \precsim_p w$. (Transitivity)
3. If $v_1 \precsim_p u_1$ and $v_2 \precsim_p u_2$, then $v_1 v_2 \precsim_p u_1 u_2$. (Closure under Append)
4. If for all s in v, $p\ s$ holds, then $uw \precsim_p uvw$.
5. If p and p' are equivalent for all s, and $v \precsim_p u$, then $v \precsim_{p'} u$.
6. If $v \precsim_p u$ and for all s in v, $p\ s$ holds, then for all s in u, $p\ s$ holds.
7. If $v \precsim_p u_1 u_2$ then there exist v_1, v_2 such that $v_1 \precsim_p u_1$ and $v_2 \precsim_p u_2$. (Split)

Previous Work

Yasuhiko Minamide
Verified decision procedures on context-free grammars
Theorem Proving in Higher Order Logics. Springer Berlin Heidelberg, 2007. 173-188