Sun 15 - Sat 21 January 2023
Boston, Massachusetts, United States

DISTINGUISHED PAPER

Statically Resolvable Ambiguity

VIKTOR PALMKVIST, *KTH Royal Institute of Technology, Sweden*
ELIAS CASTEGREN, *Uppsala University, Sweden*
PHILIPP HALLER, *KTH Royal Institute of Technology, Sweden*
DAVID BROMAN, *KTH Royal Institute of Technology, Sweden*

Proceedings of the ACM on Programming Languages, Volume 7, Issue POPL, Article No.: 58pp 1686–1712, https://doi.org/10.1145/3571251

Meta Data & Stats

Conference:	POPL
Track:	Program Analysis & Parsing
Year:	2023
Number of Authors:	4
Citations:	0
Pages (PDF):	27
Figures:	10
References:	30
Formals:	3 definitions, 6 lemmas, 1 theorem

What is the Study About?

An approach to work with less strict, ambiguous programming language grammars, which defers ambiguity resolution until parsing time and allows user to specify how resolution is done.

Table of Content

1.§. Contributions of the Study

- Novel approach of implicit and explicit grouping to resolve ambiguities
- Mechanized proof of correctness in Coq
- Reusable library

2. Motivating Resolvable Ambiguity and Overview

- Ambiguity example: `1&3==1` leads to `1&(3==1)` and `(1&3)==1`
- Define *ambiguous* (def 2.1) and *resolvable* (def 2.2) programs .
- Ambiguity from three angles: *pre-existing languages, custom operators in libs* and *new operations in DSL*.
- *Static guarantee* that all syntactically valid programs are resolvable.

2.3. Approach Overview

- Embedding the approach in a *conventional parser*.
- Introduction of running example.
- *Splittable productions, operator sequence, grouping*
- *Static resolvability* (restrictions): split productions as linked list; first/non-first operator partitioning.

3. Formal Semantics for Grouping

- Formal definitions (*bind, grouping, rules*)
- *Consistently split grouping specification* (def 3.1)

4. Static Resolvability and Its Mechanized Proof

- Statement: consistently split grouping specification => resolvable ambiguities
- Proof outline. Lemma 4.1. Lemma 4.2. Lemma 4.3. Lemma 4.4. Lemma 4.5. Lemma 4.6. Static Resolvability Theorem (Theorem 4.7).
- Proof mechanisation using Coq/TLC library (~7000 lines). Re-writing formalisation in Coq.

5. Adapting OCaml Expressions to Use a Grouper

- Modification of *Menhir* parser generator used in OCaml compiler
- Challenges: two meanings of `;` in OCaml; *if-then-else* as split production; *match* as a split production; splitting records

6. Implementation and Evaluation

- Grouping as a **library**. MCore/OCaml-like language, **~800** lines. Compute *Shared Packed Parse Forest* representing valid trees. Compute partial ambiguity resolutions with O(n) complexity.
- Use the library to re-implement OCaml expression language. **~0.04** % failure rate. Worst case overhead: **6X (0.001/0.006)**.
- Parser generator for DSLs.

7. Related Work

- *Prior work* (Palmkvist, 2021) of the same authors on resolvable ambiguity: more expressive formalism, no static guarantees.
- *Formalisms of Precedence.* (Floyd, 1963) Similarity to *operator-precedence grammars* (OPGs). (Aasa, 1995) is strongly unambiguous. (Afroozeh, 2013) Precedence through *grammar rewrites* - more verbose. (Afroozeh, 2015) *Data-dependant grammars* - more restrictive. (Danielsson, 2011) *Mixfix operators* - more unrestrictive.
- *Other approaches to ambiguity.* (Ford, 2004) Unambiguous formalism. Parser generators solutions: (Pottier, 2005), (Parr, 2011), (Parr, 2014), (Lang, 1974), (Scott, 2010), (Early, 1970).

8. Conclusion

Issues

- Why do we need this and how to apply the theory of resolvable ambiguity?
- Formal specification are hard to comprehend (gaps?)

Keywords & Terms

restricted grammar

statically resolvable ambiguity

OCaml

Coq

resolvable ambiguity

grouper

splittable productions

operator sequences