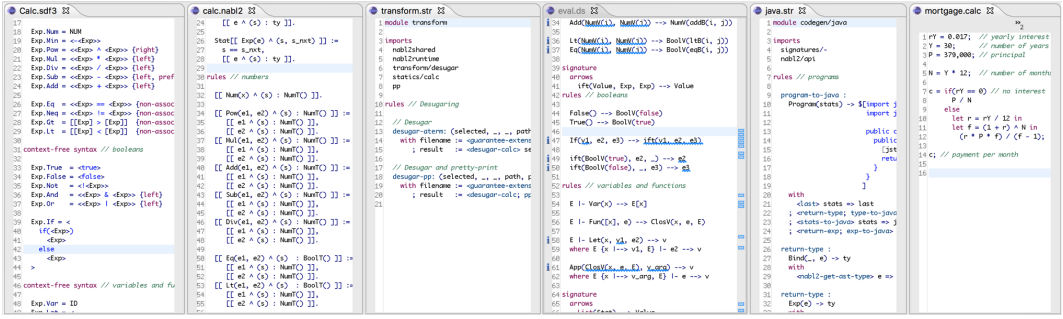
Safety and Completeness of Disambiguation corresponds to Termination and Confluence of Reordering

Eelco Visser

Joint work with Eduardo Amorim

June 30, 2020
International Workshop on Confluence (IWC'20)
Co-located with FSCD and IJCAR
'Paris, France'

Spoofax Language Workbench



SDF3: Syntax Definition + **NaBL2: Static Semantics** + **DynSem: Dynamic Semantics** + **Stratego: Program Transformation** → **Programming Environment**

Syntax Definition in Spoofax

Many syntactic (editor) services from single declarative syntax definition

```
lexical syntax
  ID = [a-zA-Z][a-zA-Z0-9]*
context-free syntax
  Exp.Var = <<ID>>
  Exp.Add = <<Exp> + <Exp>> {left}
  Exp.Mul = <<Exp> * <Exp>> {left}
context-free priorities
  Exp.Mul > Exp.Add
```

- syntax checking
- error recovery
- syntax highlighting
- abstract syntax
- formatting
- syntactic completion
- parenthesis insertion
- declarative disambiguation

History

1997 My PhD thesis with semantics of disambiguation rules for SDF2

2011 Peter Mosses observes unsafety of SDF2 rules

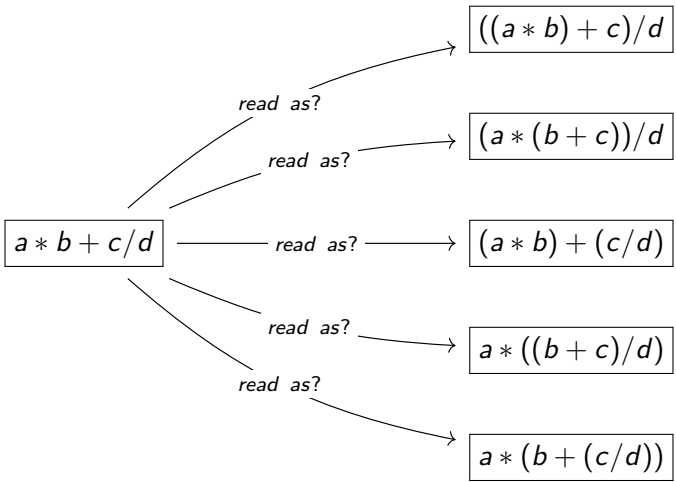
2013 Afroozeh et al. (SLE'13) define safe disambiguation with grammar transformation; semantics in terms of derivations; no proof of correctness

2018 First submission to TOPLAS and implementation of new parser generator for SDF3 integrated in Spoofax

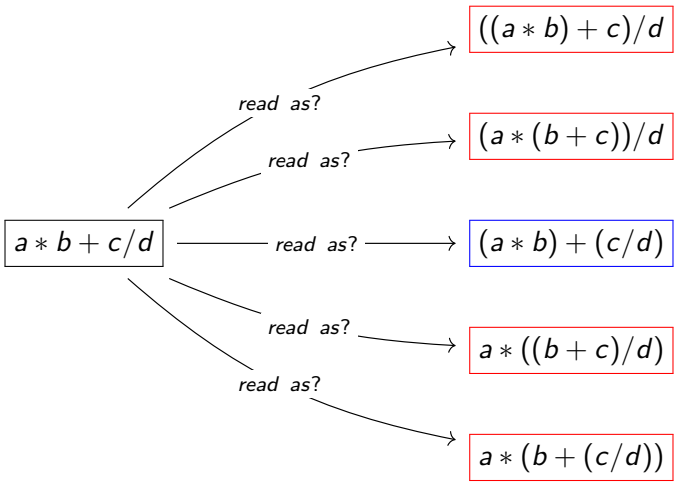
2019 Chapter in PhD thesis Eduardo Amorim and major revision for TOPLAS: safe and complete semantics based on subtree exclusion with proof (sketch)

2020 Work in progres: TOPLAS 'minor' revision with new approach to proof of safety and completeness

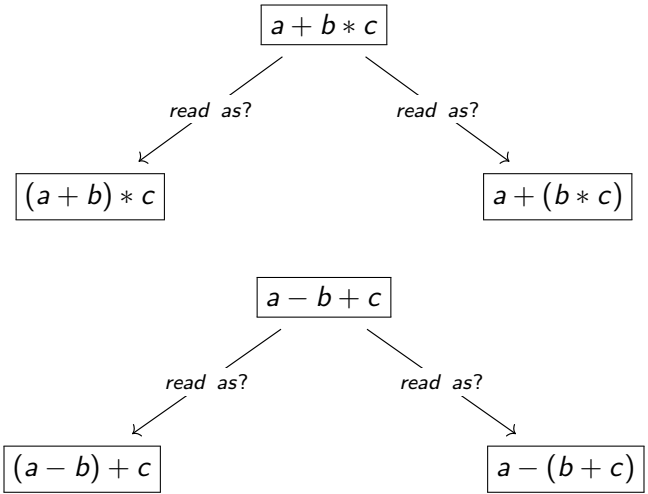
Order of Operations



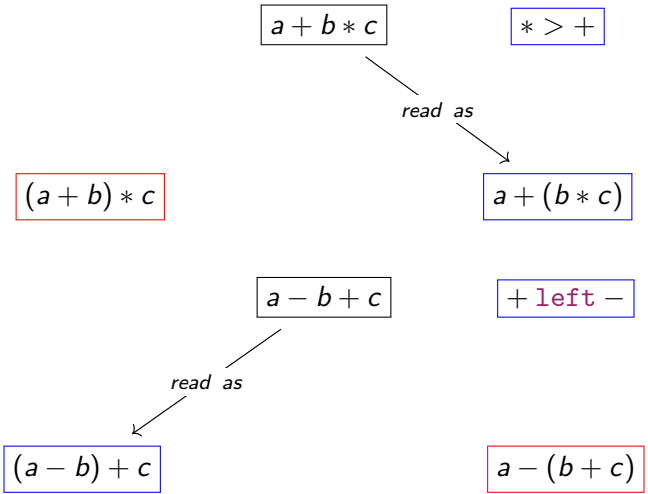
Order of Operations



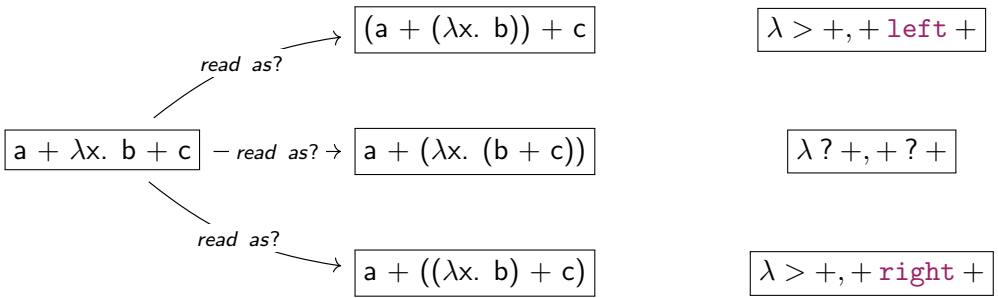
Order of Operations



Associativity and Priority



Order of Operations



Semantics of Associativity and Priority

In this talk:

- What is the semantics of associativity and priority rules?
- Is a set of disambiguation rules safe?
- Is a set of disambiguation rules complete?
- How to prove that?

Semantics of Associativity and Priority

In this talk:

- What is the semantics of associativity and priority rules?
- Is a set of disambiguation rules safe?
- Is a set of disambiguation rules complete?
- How to prove that?

Not in this talk:

- What classes of ambiguities do associativity and priority rules solve?
- What is an effective implementation strategy for disambiguation rules?

Semantics of Associativity and Priority

In this talk:

- What is the semantics of associativity and priority rules?
- Is a set of disambiguation rules safe?
- Is a set of disambiguation rules complete?
- How to prove that?

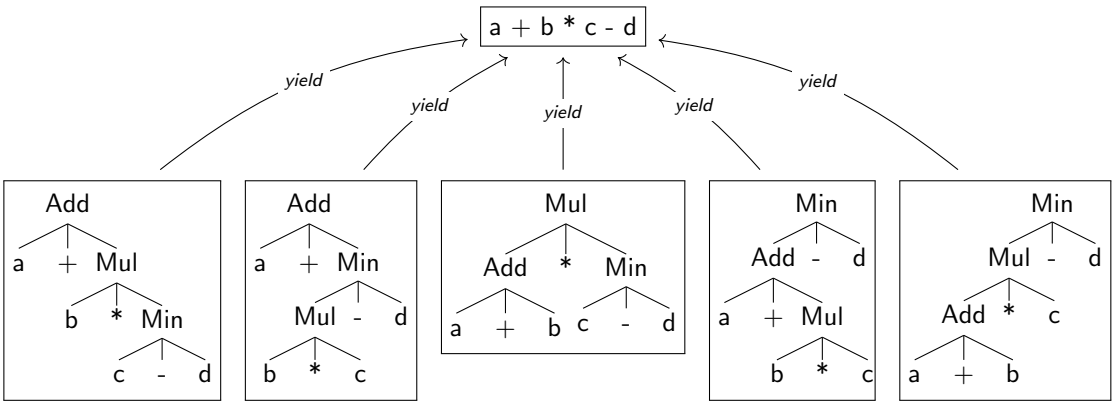
Not in this talk:

- What classes of ambiguities do associativity and priority rules solve?
- What is an effective implementation strategy for disambiguation rules?

Why is this not a solved problem?

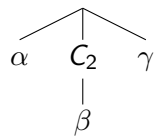
- Ambiguity of context-free grammars is undecidable; why bother?
- Existing definitions depend on specific implementations

Ambiguous Sentence has Multiple Parse Trees

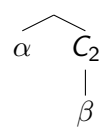


Associativity and Priority as Subtree Exclusion Rules [SDF2 (1997)]

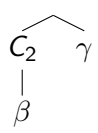
$A.C_1 > A.C_2$



$A.C_1$ `left` $A.C_2$

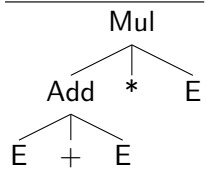


$A.C_1$ `right` $A.C_2$

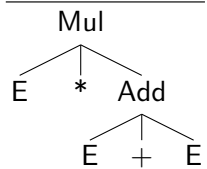


Disambiguation rules generate subtree exclusion patterns (aka conflict patterns)

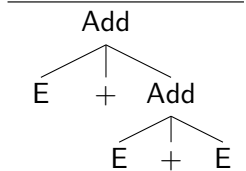
E.Mul > *E.Add*



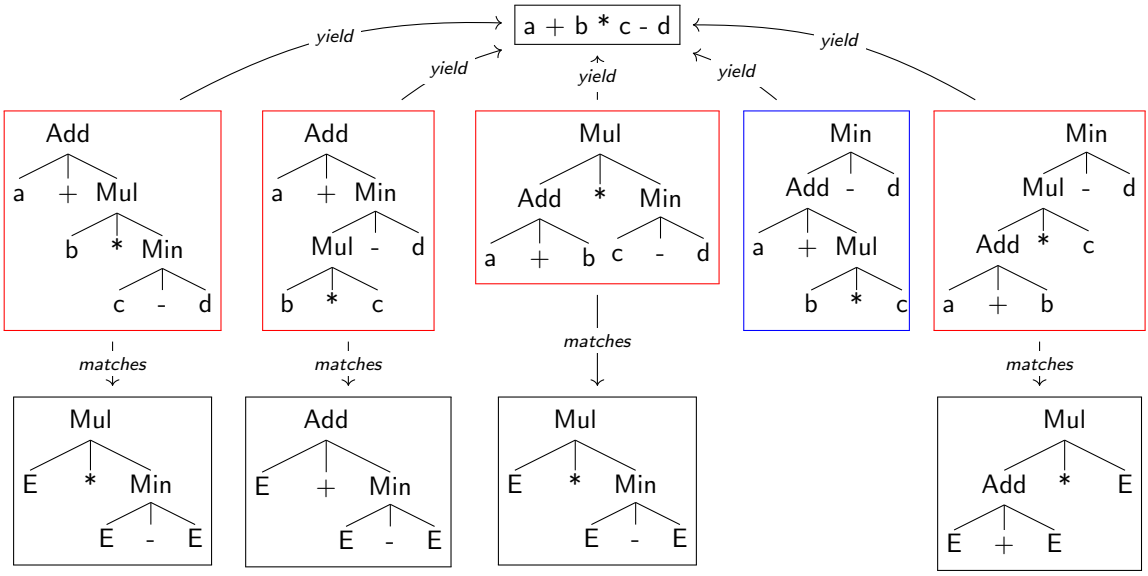
E.Mul > *E.Add*



E.Add `left` *E.Add*

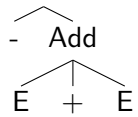


Disambiguation by Subtree Exclusion

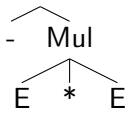


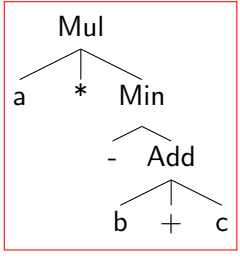
Safe for High Priority Prefix Operators

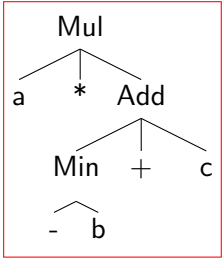
$$\frac{E.Min > E.Add}{\text{Min}}$$

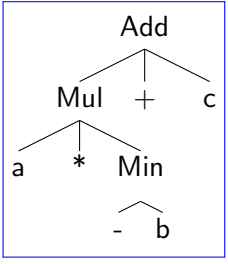


$$\frac{E.Min > E.Mul}{\text{Min}}$$

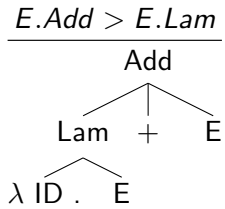


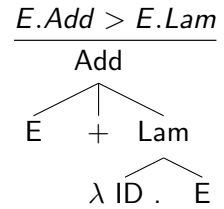


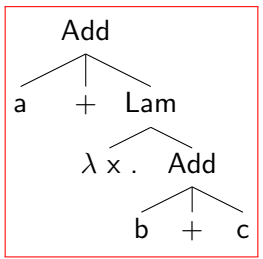


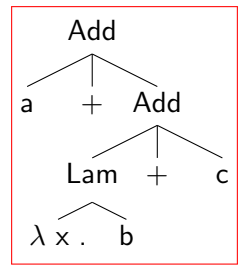


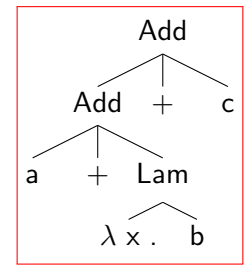
Unsafe for Low Priority Prefix Operators [SDF2]





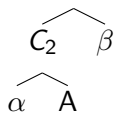




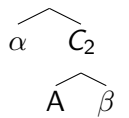


Safe Subtree Exclusion Rules [SDF3 (2019)]

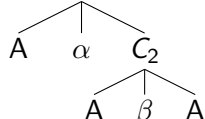
$$\frac{A.C_1 > A.C_2}{C_1}$$



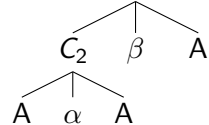
$$\frac{A.C_1 > A.C_2}{C_1}$$



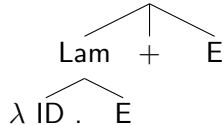
$$\frac{A.C_1 \texttt{ left } A.C_2}{C_1}$$



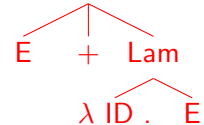
$$\frac{A.C_1 \texttt{ right } A.C_2}{C_1}$$



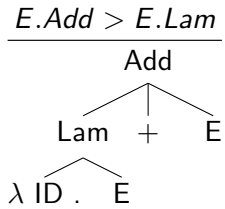
$$\frac{E.Add > E.Lam}{\text{Add}}$$

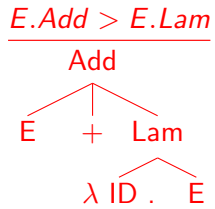


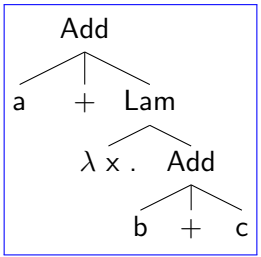
$$\frac{E.Add > E.Lam}{\text{Add}}$$ (not a pattern)

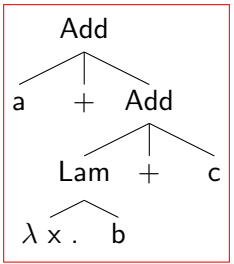


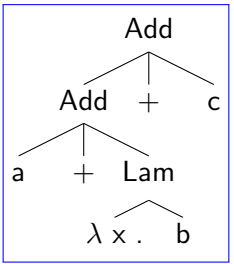
Shallow Interpretation: Safe for Low Priority Prefix Operators



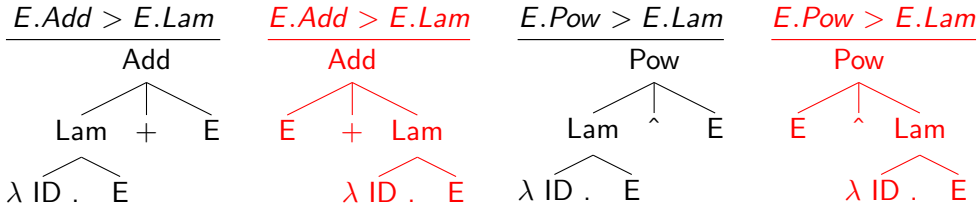




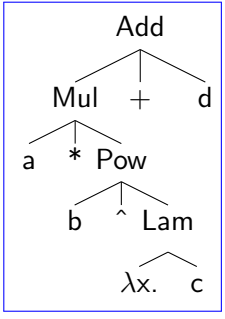


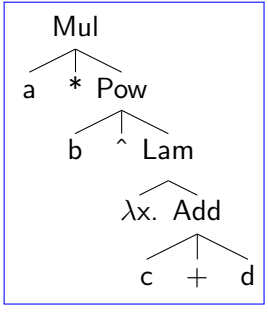


Shallow Interpretation: Incomplete for Low Priority Prefix Operators

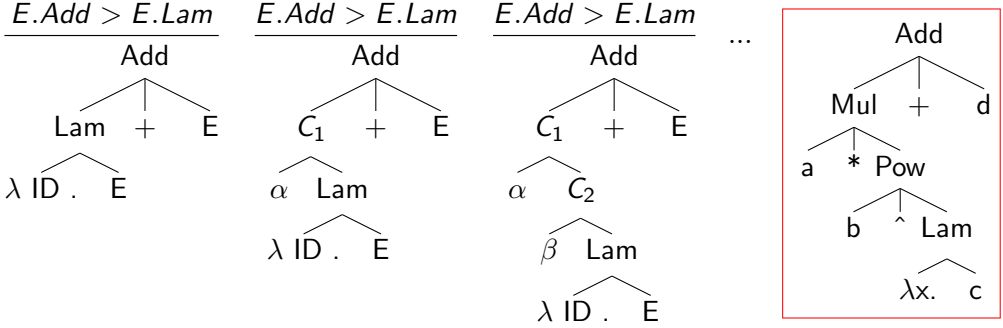


a * b ^ λ x. c + d





Deep Priority Conflicts: Match Subpattern in Right-Most Subtree



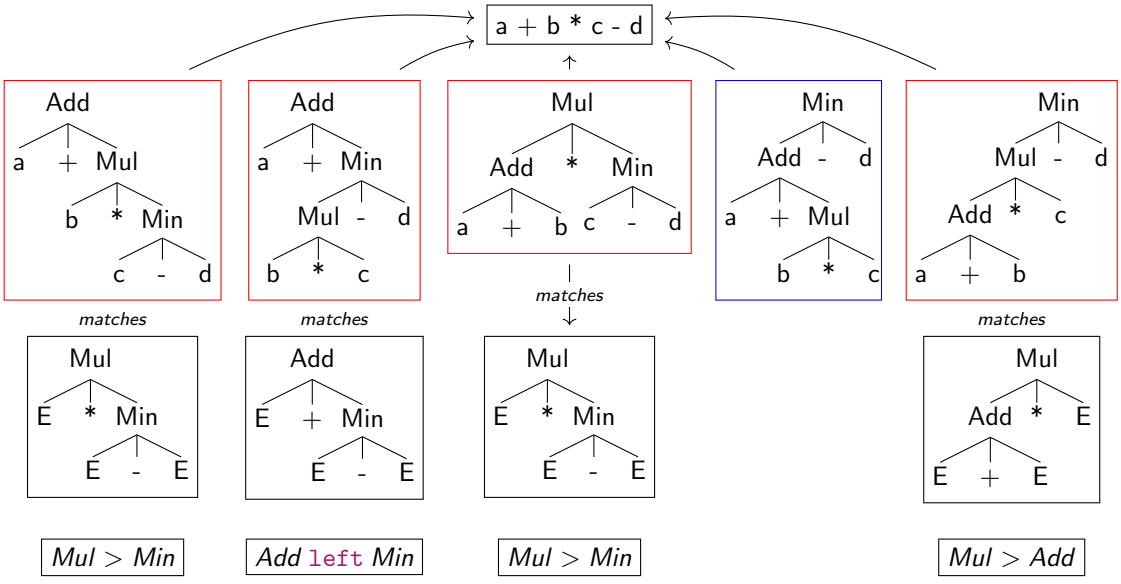
Infinite set of conflict patterns

Semantics of Associativity and Priority

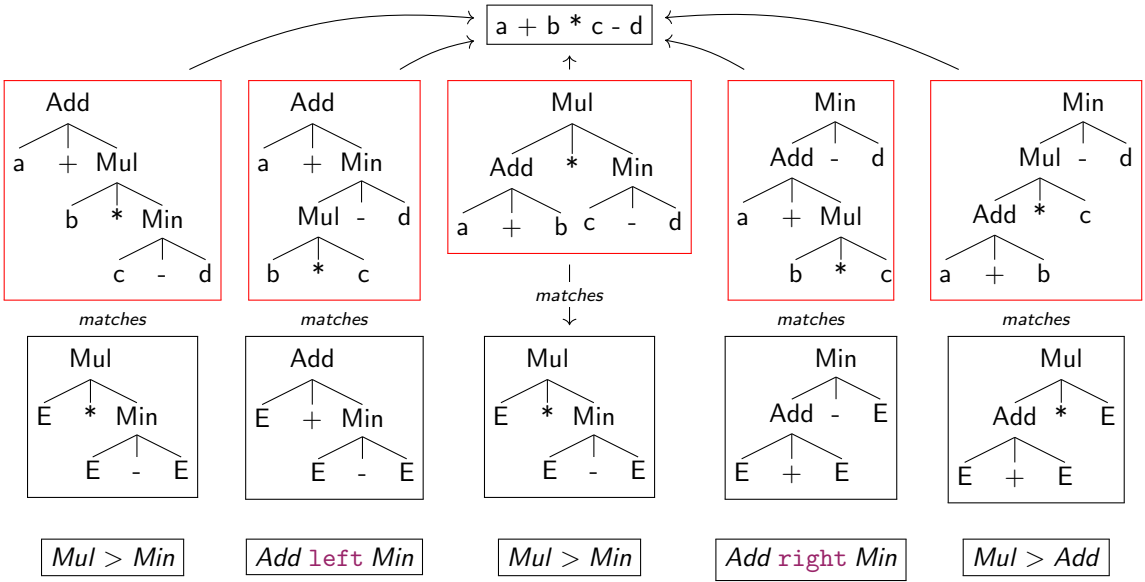
- ✓ What is the semantics of associativity and priority rules?
  - Integrated in implementation of SDF3 parser generator
  - Has been available since 2018 in Spoofax
- Is a set of disambiguation rules safe for a particular grammar?
- Is a set of disambiguation rules complete for a particular grammar?
- How to prove that?

Restricting to the case of infix expression grammars for this talk.

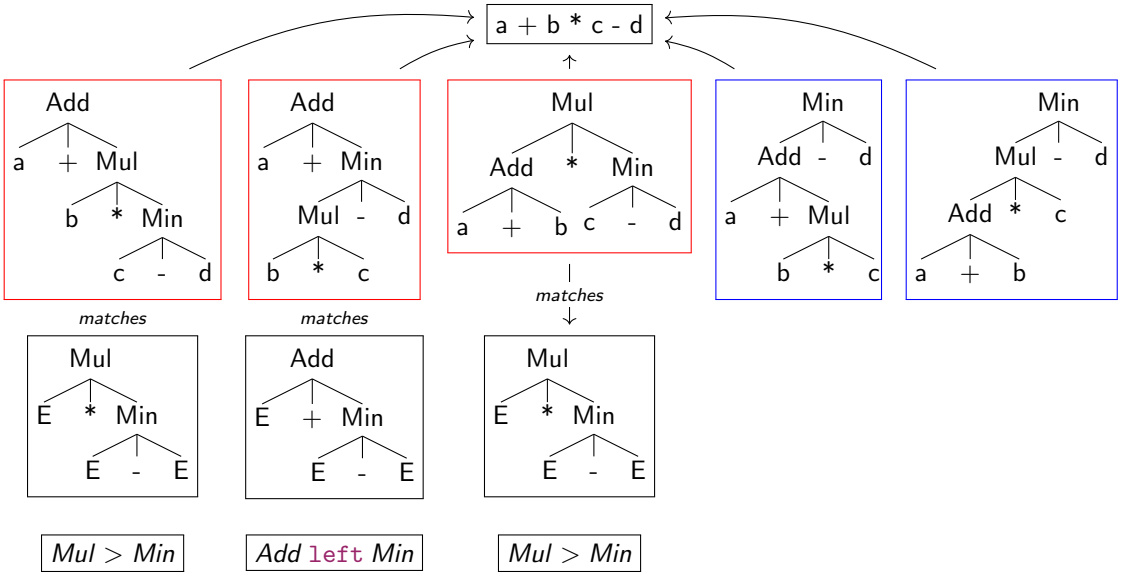
Safe and Complete Disambiguation Rules



Unsafe: Too Many Disambiguation Rules



Incomplete: Too Few Disambiguation Rules



Semantics of Associativity and Priority

- ✓ What is the semantics of associativity and priority rules?
- ✓ Is a set of disambiguation rules safe for a particular grammar?
  - At most one rule for each pair of productions
  - + some well-formedness criteria
- ✓ Is a set of disambiguation rules complete for a particular grammar?
  - At least one rule for each pair of productions
  - + some well-formedness criteria
- How to prove that?

Trees under Subtree Exclusion

Definition

A tree $t \in T^Q(G)$ iff $t \in T(G)$ and no subtree of t matches a conflict pattern in Q.

Lemma (Safety)

A disambiguation relation is safe, if for each $w \in L(G)$ there is at least one tree $t \in T^Q(G)$ such that $\text{yield}(t) = w$.

Lemma (Completeness)

A disambiguation relation is complete, if for each $w \in L(G)$ there is at most one tree $t \in T^Q(G)$ such that $\text{yield}(t) = w$.

Essence of the Proof Attempt

To prove safety:

If a tree $t \in T(G)$ has a conflict, then there is another tree for the same sentence that does not have a conflict.

Essence of the Proof Attempt

To prove safety:

If a tree $t \in T(G)$ has a conflict, then there is another tree for the same sentence that does not have a conflict.

To prove completeness:

If a tree $t \in T(G)$ does not have a conflict, then all other trees for the same sentence have conflicts.

Essence of the Proof Attempt

To prove safety:

If a tree $t \in T(G)$ has a conflict, then there is another tree for the same sentence that does not have a conflict.

To prove completeness:

If a tree $t \in T(G)$ does not have a conflict, then all other trees for the same sentence have conflicts.

Subtree exclusion is a statement about a single tree

How do we relate all trees for the same sentence?

Essence of the Proof Attempt

To prove safety:

If a tree $t \in T(G)$ has a conflict, then there is another tree for the same sentence that does not have a conflict.

To prove completeness:

If a tree $t \in T(G)$ does not have a conflict, then all other trees for the same sentence have conflicts.

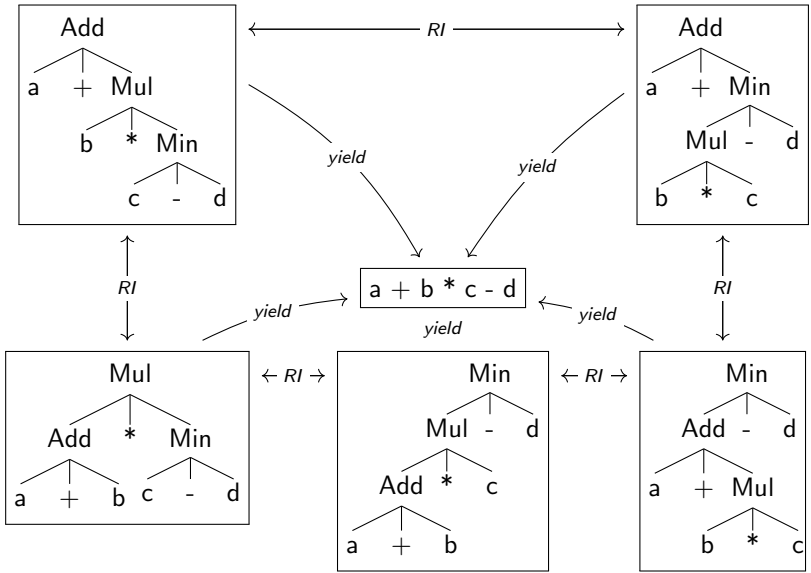
Subtree exclusion is a statement about a single tree

How do we relate all trees for the same sentence?

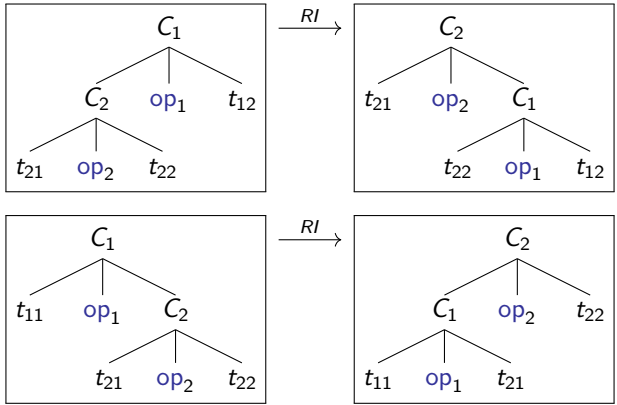
The solution is simple and elegant; once you have seen it you can't unsee it; but for the longest time I didn't see it (nor did co-authors, reviewers, other readers)*.

* But Haskell infix operators are implemented using such reorderings

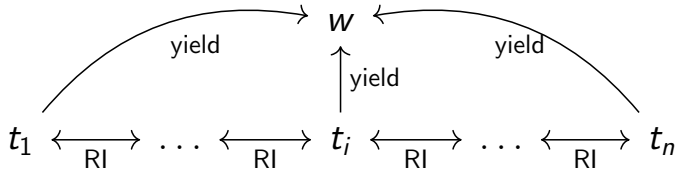
Insight: Trees for Ambiguous Sentence are Reorderings



Reordering Rewrite System

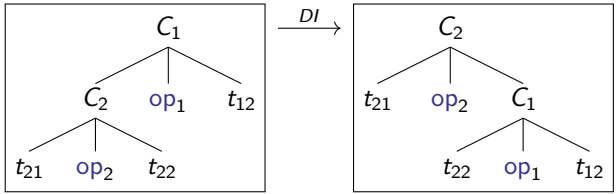


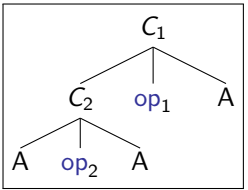
Theorem: Infix Ambiguities are Reorderings

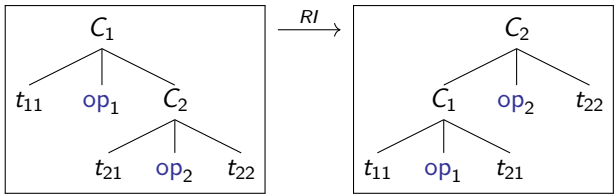


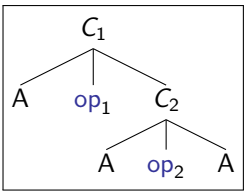
Fine print: for expression grammars beyond infix expression grammars, there are some extra requirements.

Ordering Reorderings with Conflict Patterns

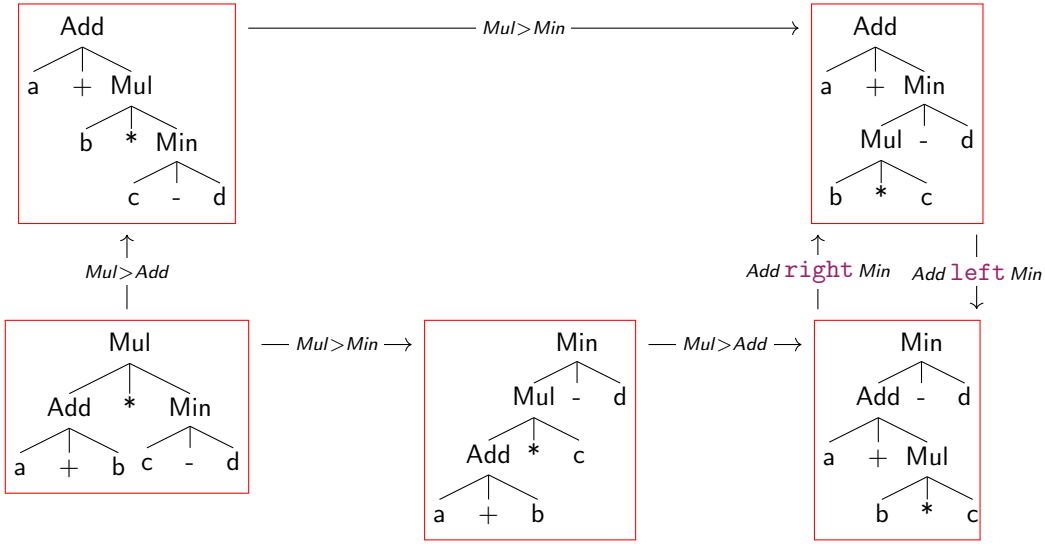


if  $\in Q$

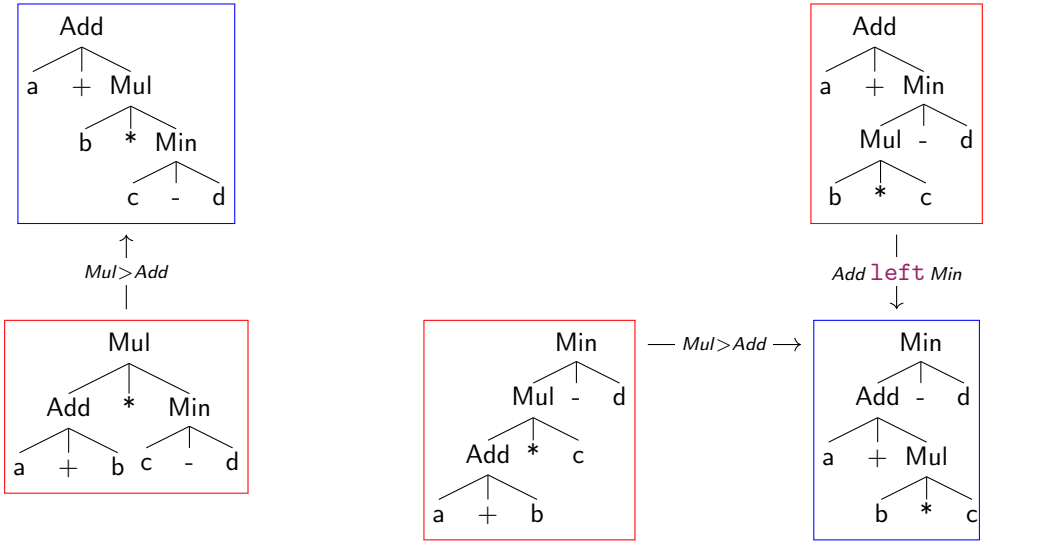


if  $\in Q$

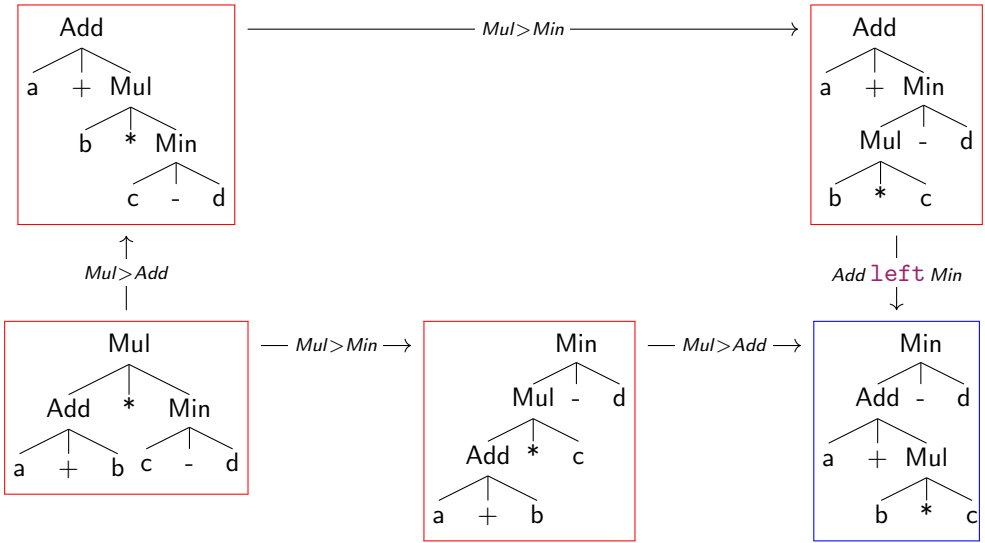
Correspondence: Unsafety is Non-Termination



Correspondence: Incompleteness is Non-Confluence (Non-Church Rosser)



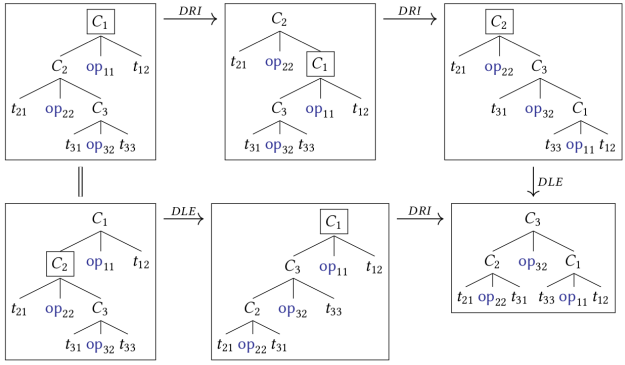
Correspondence: Safety + Completeness is Termination + Confluence



Proving Termination and Completeness

- Termination = Safety
  - A safe disambiguation relation induces a reduction order on DI
  - Roughly: number of conflicts reduces to zero
- Confluence = Completeness
  - If disambiguation relation is complete, then DI is locally confluent
  - DI (infix) has 5 critical pairs
  - The conditions for the rules in a critical pair + well-formedness criteria on disambiguation relation enable other rules and completion of the diagram
  - Extension with prefix and postfix operators: 8 rules, 28 critical pairs, 36 cases.
  - Automated by implementation in Stratego

Proving Local Confluence: Critical Pairs are Joinable



$$
\begin{aligned}
&(\mathrm{op}_{11} > \mathrm{op}_{22} \wedge \mathrm{op}_{22} > \mathrm{op}_{32} \Rightarrow \mathrm{op}_{11} > \mathrm{op}_{32}) \\
\vee\ &(\mathrm{op}_{11} > \mathrm{op}_{22} \wedge \mathrm{op}_{22}\ \texttt{left}\ \mathrm{op}_{32} \Rightarrow \mathrm{op}_{11} > \mathrm{op}_{32}) \\
\vee\ &(\mathrm{op}_{11}\ \texttt{right}\ \mathrm{op}_{22} \wedge \mathrm{op}_{22} > \mathrm{op}_{32} \Rightarrow \mathrm{op}_{11} > \mathrm{op}_{32}) \\
\vee\ &(\mathrm{op}_{11}\ \texttt{right}\ \mathrm{op}_{22} \wedge \mathrm{op}_{22}\ \texttt{left}\ \mathrm{op}_{32} \Rightarrow \texttt{false})
\end{aligned}
$$

Semantics of Associativity and Priority

- ✓ What is the semantics of associativity and priority rules?
- ✓ Is a set of disambiguation rules safe for a particular grammar?
  - At most one rule for each pair of productions
  - + some well-formedness criteria
- ✓ Is a set of disambiguation rules complete for a particular grammar?
  - At least one rule for each pair of productions
  - + some well-formedness criteria
- ✓ How to prove that?
  - Disambiguation corresponds to reordering conditional on conflict patterns
  - Trees under subtree exclusion: normal forms of DI
  - Safety of disambiguation
    - DI is terminating iff disambiguation relation is safe
  - Completeness of disambiguation
    - DI is confluent iff disambiguation relation is complete

What Else?

- What clasess of ambiguities do associativity rules solve?
  - Short answer: expression grammars for which ambiguities correspond to reorderings
  - We have investigated several classes of expression grammars: prefix/postfix operators, mixfix grammars, dangling suffix/prefix, indirect recursion, longest match of lists
- What happened to the undecidability of ambiguity?
  - Expression grammars without overlap: ambiguities are reorderings
  - Infix grammars: cannot have overlap
  - IPP grammars: harmful overlap is decidable (conjecture)
  - Mixfix grammars: harmful/less overlap undecidable in general
  - But: need only inspect productions involved in overlap
- What is an effective implementation strategy for disambiguation rules?
  - Contextual grammar transformations
  - Data dependent parsing
- A full paper is underway
  - A Direct Semantics for Declarative Disambiguation of Expression Grammars
  - Under revision for ACM TOPLAS