On the Syntactic Expressiveness of Variability Languages

FOSD 2025 | **Benjamin Moosherr**, Paul Bittner, Thomas Thüm | 2025-03-26

Technische Universität Braunschweig

universität ulm

Variability Languages

i.e. variability in the solution space

preprocessor

```
#ifdef Feature1
    Line1
#else
    Line2
#endif
```

binary choice calculus (2CC)

$\text{file} \prec \texttt{Feature1} \left\langle \text{Line1} \prec\succ, \text{Line2} \prec\succ \right\rangle \succ$

Motivation

variability languages

- binary choice calculus (2CC)
- n-ary choice calculus (NCC)
- core choice calculus (CCC)
- algebraic decision trees (ADT)
- feature structure trees (FST)
- option calculus (OC)
- clown-and-own (CaO)

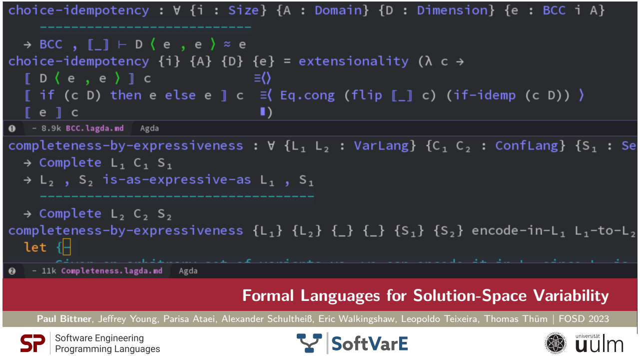
Motivation

variability languages

- binary choice calculus (2CC)
- n-ary choice calculus (NCC)
- core choice calculus (CCC)
- algebraic decision trees (ADT)
- feature structure trees (FST)
- option calculus (OC)
- clown-and-own (CaO)

question: which one should you to choose for your project?

Previous Work



FOSD 2023 (Paul Bittner)

On the Expressive Power of Languages for Static Variability

PAUL MAXIMILIAN BITTNER, Paderborn University and Ulm University, Germany
ALEXANDER SCHULTHEISS, Paderborn University, Germany and University of Bern, Switzerland
BENJAMIN MOOSHERR, Ulm University, Germany
JEFFREY M. YOUNG, Input Output Global, USA
LEOPOLDO TEIXEIRA, Federal University of Pernambuco, Brazil
ERIC WALKINGSHAW, Unaffiliated, USA
PARISA ATAEI, Input Output Global, USA
THOMAS THÜM, Paderborn University and TU Braunschweig, Germany

Variability permeates software development to satisfy ever-changing requirements and mass-customization needs. A prime example is the Linux kernel, which employs the C preprocessor to specify a set of related but distinct kernel variants. To study, analyze, and verify variational software, several formal languages have been proposed. For example, the choice calculus has been successfully applied for type checking and symbolic execution of configurable software, while other formalisms have been used for variational model checking, change impact analysis, among other use cases. Yet, these languages have not been formally compared, hence, little is known about their relationships. Crucially, it is unclear to what extent one language subsumes another, how research results from one language can be applied to other languages, and which language is suitable for which purpose or domain. In this paper, we propose a formal framework to compare the expressive power of languages for static (i.e. compile-time) variability. By establishing a common semantic domain to capture a widely used intuition of explicit variability, we can formulate the basic, yet to date neglected, properties of soundness, completeness, and expressiveness for variability languages. We then prove the (un)soundness and (in)completeness of a range of existing languages, and relate their ability to express the same variational systems. We implement our framework as an extensible open source Agda library in which proofs act as correct compilers between languages or differencing algorithms. We find different levels of expressiveness as well as complete and incomplete languages w.r.t. our unified semantic domain, with the choice calculus being among the most expressive languages.

CCS Concepts: • **Software and its engineering** → *Formal language definitions*; **Software configuration management and version control systems**; **Software product lines**.

Additional Key Words and Phrases: variation, software product lines, configuration, language semantics

ACM Reference Format:
Paul Maximilian Bittner, Alexander Schultheiß, Benjamin Moosherr, Jeffrey M. Young, Leopoldo Teixeira, Eric Walkingshaw, Parisa Ataei, and Thomas Thüm. 2024. On the Expressive Power of Languages for Static Variability. *Proc. ACM Program. Lang.* 8, OOPSLA2, Article 307 (October 2024), 33 pages. https://doi.org/10.1145/3689747

Authors' Contact Information: Paul Maximilian Bittner, paul.bittner@uni-ulm.de, Paderborn University and Ulm University, Germany; Alexander Schultheiß, alexander.schultheiss@hu-berlin.de, Paderborn University, Paderborn, Germany and University of Bern, Bern, Switzerland; Benjamin Moosherr, benjamin.moosherr@uni-ulm.de, Ulm University, Ulm, Germany; Jeffrey M. Young, jeffrey.young@iohk.io, Input Output Global, Longmont, Colorado, USA; Leopoldo Teixeira, lmt@cin.ufpe.br, Federal University of Pernambuco, Pernambuco, Brazil; Eric Walkingshaw, eric@walkingshaw.net, Unaffiliated, Corvallis, Oregon, USA; Parisa Ataei, paris.ataei@gmail.com, Input Output Global, Buffalo, New York, USA; Thomas Thüm, thomas.thuem@uni-paderborn.de, Paderborn University and TU Braunschweig, Paderborn, Germany.

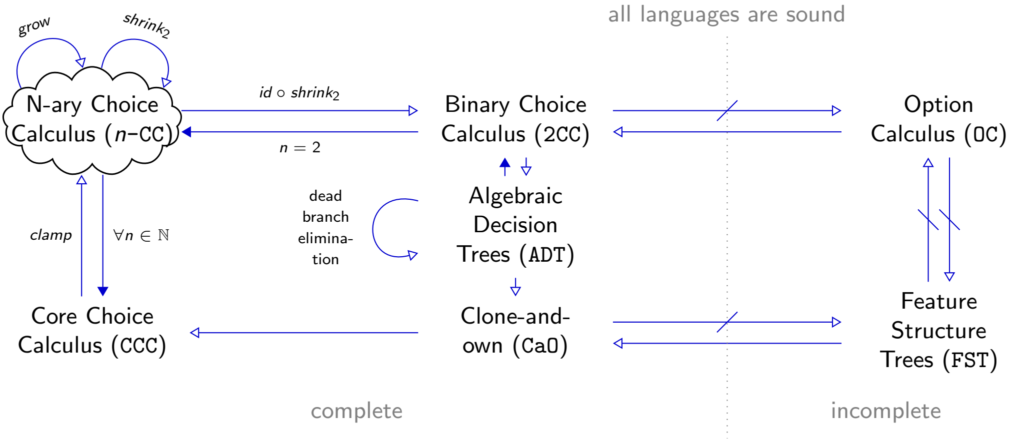
This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.
© 2024 Copyright held by the owner/author(s).
ACM 2475-1421/2024/10-ART307
https://doi.org/10.1145/3689747

Proc. ACM Program. Lang., Vol. 8, No. OOPSLA2, Article 307. Publication date: October 2024.

OOPSLA 2024[1]

[1] Bittner et al. "On the Expressive Power of Languages for Static Variability". In: *Proc. ACM Program. Lang.* 8.OOPSLA2 (Oct. 2024)

Previous Work[1]



[1] Bittner et al. "On the Expressive Power of Languages for Static Variability". In: *Proc. ACM Program. Lang.* 8.OOPSLA2 (Oct. 2024)

Motivation

variability languages

- binary choice calculus (2CC)
- n-ary choice calculus (NCC)
- core choice calculus (CCC)
- algebraic decision trees (ADT)
- feature structure trees (FST)
- option calculus (OC)
- clown-and-own (CaO)

question: which one should you to choose for your project?

goal for this thesis:
compare these variability languages regarding their expressiveness/compression

Example of Size Differences

```
#ifdef Feature1
    Line1
#else
    Line2
#endif
```

Example of Size Differences

```
#ifdef Feature1
    Line1
#else
    Line2
#endif


#ifdef Feature2
    Line3
#else
    Line4
#endif
```

Example of Size Differences

```
#ifdef Feature1
    Line1
#else
    Line2
#endif

#ifdef Feature2
    Line3
#else
    Line4
#endif

#ifdef Feature3
    Line5
#else
    Line6
#endif
```

Example of Size Differences

```
#ifdef Feature1
    Line1
#else
    Line2
#endif
```

```
#ifdef Feature2
    Line3
#else
    Line4
#endif
```

```
#ifdef Feature3
    Line5
#else
    Line6
#endif
```

```
#ifdef Feature4
    Line7
#else
    Line8
#endif
```

Example of Size Differences

2CC

file ≺
  Feature1 ⟨ Line1 ≺≻, Line2 ≺≻ ⟩
≻

ADT

Feature1 ⟨ file ≺ Line1 ≺≻ ≻, file
≺ Line2 ≺≻ ≻ ⟩

Example of Size Differences

2CC

```
file ≺
   Feature1 ⟨ Line1 ≺≻, Line2 ≺≻ ⟩,
   Feature2 ⟨ Line3 ≺≻, Line4 ≺≻ ⟩
≻
```

ADT

```
Feature1 ⟨ Feature2 ⟨ file ≺ Line1
≺≻, Line3 ≺≻ ≻, file ≺ Line1 ≺≻,
Line4 ≺≻ ≻ ⟩, Feature2 ⟨ file ≺
Line2 ≺≻, Line3 ≺≻ ≻, file ≺
Line2 ≺≻, Line4 ≺≻ ≻ ⟩ ⟩
```

Example of Size Differences

2CC

```
file ≺
   Feature1 ⟨ Line1 ≺≻, Line2 ≺≻ ⟩,
   Feature2 ⟨ Line3 ≺≻, Line4 ≺≻ ⟩,
   Feature3 ⟨ Line5 ≺≻, Line6 ≺≻ ⟩
≻
```

ADT

```
Feature1 ⟨ Feature2 ⟨ Feature3 ⟨ file ≺
Line1 ≺≻, Line3 ≺≻, Line5 ≺≻ ≻, file ≺
Line1 ≺≻, Line3 ≺≻, Line6 ≺≻ ≻ ⟩,
Feature3 ⟨ file ≺ Line1 ≺≻, Line4 ≺≻,
Line5 ≺≻ ≻, file ≺ Line1 ≺≻, Line4 ≺≻,
Line6 ≺≻ ≻ ⟩ ⟩, Feature2 ⟨ Feature3 ⟨ file
≺ Line2 ≺≻, Line3 ≺≻, Line5 ≺≻ ≻, file ≺
Line2 ≺≻, Line3 ≺≻, Line6 ≺≻ ≻ ⟩,
Feature3 ⟨ file ≺ Line2 ≺≻, Line4 ≺≻,
Line5 ≺≻ ≻, file ≺ Line2 ≺≻, Line4 ≺≻,
Line6 ≺≻ ≻ ⟩ ⟩ ⟩
```

Example of Size Differences

2CC

```
file ≺
   Feature1 ⟨ Line1 ≺≻, Line2 ≺≻ ⟩,
   Feature2 ⟨ Line3 ≺≻, Line4 ≺≻ ⟩,
   Feature3 ⟨ Line5 ≺≻, Line6 ≺≻ ⟩,
   Feature4 ⟨ Line7 ≺≻, Line8 ≺≻ ⟩
≻
```

ADT

```
Feature1 ⟨ Feature2 ⟨ Feature3 ⟨ Feature4 ⟨ file ≺ Line1 ≺≻,
Line3 ≺≻, Line5 ≺≻, Line7 ≺≻ ≻, file ≺ Line1 ≺≻, Line3
≺≻, Line5 ≺≻, Line8 ≺≻ ≻ ⟩, Feature4 ⟨ file ≺ Line1 ≺≻,
Line3 ≺≻, Line6 ≺≻, Line7 ≺≻ ≻, file ≺ Line1 ≺≻, Line3
≺≻, Line6 ≺≻, Line8 ≺≻ ≻ ⟩ ⟩, Feature3 ⟨ Feature4 ⟨ file ≺
Line1 ≺≻, Line4 ≺≻, Line5 ≺≻, Line7 ≺≻ ≻, file ≺ Line1
≺≻, Line4 ≺≻, Line5 ≺≻, Line8 ≺≻ ≻ ⟩, Feature4 ⟨ file ≺
Line1 ≺≻, Line4 ≺≻, Line6 ≺≻, Line7 ≺≻ ≻, file ≺ Line1
≺≻, Line4 ≺≻, Line6 ≺≻, Line8 ≺≻ ≻ ⟩ ⟩ ⟩, Feature2 ⟨
Feature3 ⟨ Feature4 ⟨ file ≺ Line2 ≺≻, Line3 ≺≻, Line5 ≺≻,
Line7 ≺≻ ≻, file ≺ Line2 ≺≻, Line3 ≺≻, Line5 ≺≻, Line8 ≺≻
≻ ⟩, Feature4 ⟨ file ≺ Line2 ≺≻, Line3 ≺≻, Line6 ≺≻, Line7
≺≻ ≻, file ≺ Line2 ≺≻, Line3 ≺≻, Line6 ≺≻, Line8 ≺≻ ≻ ⟩
⟩, Feature3 ⟨ Feature4 ⟨ file ≺ Line2 ≺≻, Line4 ≺≻, Line5
≺≻, Line7 ≺≻ ≻, file ≺ Line2 ≺≻, Line4 ≺≻, Line5 ≺≻,
Line8 ≺≻ ≻ ⟩, Feature4 ⟨ file ≺ Line2 ≺≻, Line4 ≺≻, Line6
≺≻, Line7 ≺≻ ≻, file ≺ Line2 ≺≻, Line4 ≺≻, Line6 ≺≻,
Line8 ≺≻ ≻ ⟩ ⟩ ⟩ ⟩
```

Definition of Size

- derived from the syntax
- count productions/constructors

Definition of Size

- derived from the syntax
- count productions/constructors
- example: 2CC
  - grammar:

$$\begin{aligned} \mathit{2CC} \; &\texttt{:=} \; \texttt{a} \prec \mathit{2CC}, \mathit{2CC}, \ldots, \mathit{2CC} \succ && \text{(artifact)} \\ &\;\texttt{|}\;\; \texttt{D} \, \langle \, \mathit{2CC}, \mathit{2CC} \, \rangle && \text{(choice)} \end{aligned}$$

Definition of Size

- derived from the syntax
- count productions/constructors
- example: 2CC
  - grammar:

    $$\begin{aligned} \mathit{2CC} \;:=\;& \mathtt{a} \prec \mathit{2CC}, \mathit{2CC}, \ldots, \mathit{2CC} \succ && \text{(artifact)} \\ \mid\;& \mathtt{D} \langle \mathit{2CC}, \mathit{2CC} \rangle && \text{(choice)} \end{aligned}$$

  - expression:

    $\mathit{2CC}$

    size: 0

Definition of Size

- derived from the syntax
- count productions/constructors
- example: 2CC
  - grammar:
    $$\begin{aligned} \mathit{2CC} \;\texttt{:=}\; & \texttt{a} \prec \mathit{2CC}, \mathit{2CC}, \ldots, \mathit{2CC} \succ && \text{(artifact)} \\ \mid\; & \texttt{D} \langle \mathit{2CC}, \mathit{2CC} \rangle && \text{(choice)} \end{aligned}$$
  - expression:
    $\texttt{file} \prec \mathit{2CC} \succ$
    size: 1

Definition of Size

- derived from the syntax
- count productions/constructors
- example: 2CC
  - grammar:
    $$\mathit{2CC} \ \texttt{:=}\ \texttt{a} \prec \mathit{2CC}, \mathit{2CC}, \ldots, \mathit{2CC} \succ \qquad \text{(artifact)}$$
    $$\mid\ \texttt{D} \left\langle\ \mathit{2CC}, \mathit{2CC}\ \right\rangle \qquad \text{(choice)}$$
  - expression:
    $$\texttt{file} \prec \texttt{Feature1} \left\langle\ \mathit{2CC}\ ,\ \mathit{2CC}\ \right\rangle \succ$$
    size: 2

Definition of Size

- derived from the syntax
- count productions/constructors
- example: 2CC
  - grammar:
    $$\begin{aligned} \mathit{2CC} \;&{:=}\; \texttt{a} \prec \mathit{2CC}, \mathit{2CC}, \ldots, \mathit{2CC} \succ && \text{(artifact)} \\ &\mid\; \texttt{D}\,\langle\, \mathit{2CC}, \mathit{2CC} \,\rangle && \text{(choice)} \end{aligned}$$
  - expression:
    $$\texttt{file} \prec \texttt{Feature1}\,\langle\, \texttt{Line1} \prec\succ, \mathit{2CC} \,\rangle \succ$$
    size: 3

Definition of Size

- derived from the syntax
- count productions/constructors
- example: 2CC
  - grammar:
    $$\begin{aligned} \mathit{2CC} \; &:= \; \texttt{a} \prec \mathit{2CC}, \mathit{2CC}, \ldots, \mathit{2CC} \succ && \text{(artifact)} \\ &\;\;\mid \; \texttt{D} \langle \mathit{2CC}, \mathit{2CC} \rangle && \text{(choice)} \end{aligned}$$
  - expression:
    $$\texttt{file} \prec \texttt{Feature1} \langle \texttt{Line1} \prec\succ, \texttt{Line2} \prec\succ \rangle \succ$$
    size: 4

Size comparison

- should be independent of small differences
- inspiration: Big-O notation
- used for comparing the growth of functions
- $f = O(g) \quad := \quad \exists c, n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
- this requires enumeration

Formalization

$$f = O(g) \quad := \quad \exists c, n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq c \cdot g(n)$$

Formalization

$$f = O(g) \quad := \quad \exists c, n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq c \cdot g(n)$$
$$\Leftrightarrow \quad \exists c \in \mathbb{N} : \forall n \in \mathbb{N} : f(n) \leq c \cdot g(n) \qquad (\text{if } g > 0)$$

Formalization

$$f = O(g) \quad := \quad \exists c, n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq c \cdot g(n)$$

$$\Leftrightarrow \quad \exists c \in \mathbb{N} : \forall n \in \mathbb{N} : f(n) \leq c \cdot g(n) \qquad (\text{if } g > 0)$$

$$L_1 \leq_s L_2 \quad := \quad \exists c \in \mathbb{N} : \forall e_2 \in L_2 : \exists e_1 \in L_1 : e_1 \cong e_2 \wedge size(e_1) \leq c \cdot size(e_2)$$

Formalization

$$
\begin{aligned}
f = O(g) \quad &:= \quad \exists c, n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq c \cdot g(n) \\
&\Leftrightarrow \quad \exists c \in \mathbb{N} : \forall n \in \mathbb{N} : f(n) \leq c \cdot g(n) \qquad \text{(if } g > 0\text{)}
\end{aligned}
$$

$$
L_1 \leq_s L_2 \quad := \quad \exists c \in \mathbb{N} : \forall e_2 \in L_2 : \exists e_1 \in L_1 : e_1 \cong e_2 \wedge \mathit{size}(e_1) \leq c \cdot \mathit{size}(e_2)
$$

$$
\begin{aligned}
L_1 \not\leq_s L_2 \quad &:= \quad \neg(L_1 \leq_s L_2) \\
&\Leftrightarrow \quad \forall c \in \mathbb{N} : \exists e_2 \in L_2 : \forall e_1 \in L_1 : e_1 \cong e_2 \rightarrow \mathit{size}(e_1) > c \cdot \mathit{size}(e_2)
\end{aligned}
$$

Example of Size Differences

$$ADT \not\leq_s 2CC \quad \Leftrightarrow \quad \forall c \in \mathbb{N} : \exists e_2 \in 2CC : \forall e_1 \in ADT : e_1 \cong e_2 \rightarrow size(e_1) > c \cdot size(e_2)$$

2CC

size $= 4$

```
file ≺
   Feature1 ⟨ Line1 ≺≻, Line2 ≺≻ ⟩
≻
```

ADT

size $= 5$

```
Feature1 ⟨ file ≺ Line1 ≺≻ ≻, file
≺ Line2 ≺≻ ≻ ⟩
```

Example of Size Differences

$$ADT \not\leq_s 2CC \quad \Leftrightarrow \quad \forall c \in \mathbb{N} : \exists e_2 \in 2CC : \forall e_1 \in ADT : e_1 \cong e_2 \rightarrow size(e_1) > c \cdot size(e_2)$$

2CC
size = 7

```
file ≺
   Feature1 ⟨ Line1 ≺≻, Line2 ≺≻ ⟩,
   Feature2 ⟨ Line3 ≺≻, Line4 ≺≻ ⟩
≻
```

ADT
size = 27

```
Feature1 ⟨ Feature2 ⟨ file ≺ Line1
≺≻, Line3 ≺≻ ≻, file ≺ Line1 ≺≻,
Line4 ≺≻ ≻ ⟩, Feature2 ⟨ file ≺
Line2 ≺≻, Line3 ≺≻ ≻, file ≺
Line2 ≺≻, Line4 ≺≻ ≻ ⟩ ⟩
```

Example of Size Differences

$$ADT \not\leq_s 2CC \quad \Leftrightarrow \quad \forall c \in \mathbb{N} : \exists e_2 \in 2CC : \forall e_1 \in ADT : e_1 \cong e_2 \rightarrow size(e_1) > c \cdot size(e_2)$$

2CC
size = 10

```
file ≺
   Feature1 ⟨ Line1 ≺≻, Line2 ≺≻ ⟩,
   Feature2 ⟨ Line3 ≺≻, Line4 ≺≻ ⟩,
   Feature3 ⟨ Line5 ≺≻, Line6 ≺≻ ⟩
≻
```

ADT
size = 39

```
Feature1 ⟨ Feature2 ⟨ Feature3 ⟨ file ≺
Line1 ≺≻, Line3 ≺≻, Line5 ≺≻ ≻, file ≺
Line1 ≺≻, Line3 ≺≻, Line6 ≺≻ ≻ ⟩,
Feature3 ⟨ file ≺ Line1 ≺≻, Line4 ≺≻,
Line5 ≺≻ ≻, file ≺ Line1 ≺≻, Line4 ≺≻,
Line6 ≺≻ ≻ ⟩ ⟩, Feature2 ⟨ Feature3 ⟨ file
≺ Line2 ≺≻, Line3 ≺≻, Line5 ≺≻ ≻, file ≺
Line2 ≺≻, Line3 ≺≻, Line6 ≺≻ ≻ ⟩,
Feature3 ⟨ file ≺ Line2 ≺≻, Line4 ≺≻,
Line5 ≺≻ ≻, file ≺ Line2 ≺≻, Line4 ≺≻,
Line6 ≺≻ ≻ ⟩ ⟩ ⟩
```

Example of Size Differences

$$ADT \not\leq_s 2CC \quad \Leftrightarrow \quad \forall c \in \mathbb{N} : \exists e_2 \in 2CC : \forall e_1 \in ADT : e_1 \cong e_2 \rightarrow size(e_1) > c \cdot size(e_2)$$

2CC
size = 13

```
file ≺
   Feature1 ⟨ Line1 ≺≻, Line2 ≺≻ ⟩,
   Feature2 ⟨ Line3 ≺≻, Line4 ≺≻ ⟩,
   Feature3 ⟨ Line5 ≺≻, Line6 ≺≻ ⟩,
   Feature4 ⟨ Line7 ≺≻, Line8 ≺≻ ⟩
≻
```

ADT
size = 95

```
Feature1 ⟨ Feature2 ⟨ Feature3 ⟨ Feature4 ⟨ file ≺ Line1 ≺≻,
Line3 ≺≻, Line5 ≺≻, Line7 ≺≻ ≻, file ≺ Line1 ≺≻, Line3
≺≻, Line5 ≺≻, Line8 ≺≻ ≻ ⟩, Feature4 ⟨ file ≺ Line1 ≺≻,
Line3 ≺≻, Line6 ≺≻, Line7 ≺≻ ≻, file ≺ Line1 ≺≻, Line3
≺≻, Line6 ≺≻, Line8 ≺≻ ≻ ⟩ ⟩, Feature3 ⟨ Feature4 ⟨ file ≺
Line1 ≺≻, Line4 ≺≻, Line5 ≺≻, Line7 ≺≻ ≻, file ≺ Line1
≺≻, Line4 ≺≻, Line5 ≺≻, Line8 ≺≻ ≻ ⟩, Feature4 ⟨ file ≺
Line1 ≺≻, Line4 ≺≻, Line6 ≺≻, Line7 ≺≻ ≻, file ≺ Line1
≺≻, Line4 ≺≻, Line6 ≺≻, Line8 ≺≻ ≻ ⟩ ⟩ ⟩, Feature2 ⟨
Feature3 ⟨ Feature4 ⟨ file ≺ Line2 ≺≻, Line3 ≺≻, Line5 ≺≻,
Line7 ≺≻ ≻, file ≺ Line2 ≺≻, Line3 ≺≻, Line5 ≺≻, Line8 ≺≻
≻ ⟩, Feature4 ⟨ file ≺ Line2 ≺≻, Line3 ≺≻, Line6 ≺≻, Line7
≺≻ ≻, file ≺ Line2 ≺≻, Line3 ≺≻, Line6 ≺≻, Line8 ≺≻ ≻ ⟩
⟩, Feature3 ⟨ Feature4 ⟨ file ≺ Line2 ≺≻, Line4 ≺≻, Line5
≺≻, Line7 ≺≻ ≻, file ≺ Line2 ≺≻, Line4 ≺≻, Line5 ≺≻,
Line8 ≺≻ ≻ ⟩, Feature4 ⟨ file ≺ Line2 ≺≻, Line4 ≺≻, Line6
≺≻, Line7 ≺≻ ≻, file ≺ Line2 ≺≻, Line4 ≺≻, Line6 ≺≻,
Line8 ≺≻ ≻ ⟩ ⟩ ⟩ ⟩
```

Formalization

$$
\begin{aligned}
f = O(g) \quad &:= \quad \exists c, n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq c \cdot g(n) \\
&\Leftrightarrow \quad \exists c \in \mathbb{N} : \forall n \in \mathbb{N} : f(n) \leq c \cdot g(n) \qquad \text{(if } g > 0\text{)}
\end{aligned}
$$

$$
L_1 \leq_s L_2 \quad := \quad \exists c \in \mathbb{N} : \forall e_2 \in L_2 : \exists e_1 \in L_1 : e_1 \cong e_2 \wedge \mathit{size}(e_1) \leq c \cdot \mathit{size}(e_2)
$$

$$
\begin{aligned}
L_1 \not\leq_s L_2 \quad &:= \quad \neg(L_1 \leq_s L_2) \\
&\Leftrightarrow \quad \forall c \in \mathbb{N} : \exists e_2 \in L_2 : \forall e_1 \in L_1 : e_1 \cong e_2 \to \mathit{size}(e_1) > c \cdot \mathit{size}(e_2)
\end{aligned}
$$

Formalization

$$
\begin{aligned}
f = O(g) \quad &:= \quad \exists c, n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq c \cdot g(n) \\
&\Leftrightarrow \quad \exists c \in \mathbb{N} : \forall n \in \mathbb{N} : f(n) \leq c \cdot g(n) \qquad \text{(if } g > 0\text{)}
\end{aligned}
$$

$$
L_1 \leq_s L_2 \quad := \quad \exists c \in \mathbb{N} : \forall e_2 \in L_2 : \exists e_1 \in L_1 : e_1 \cong e_2 \wedge \mathit{size}(e_1) \leq c \cdot \mathit{size}(e_2)
$$

$$
\begin{aligned}
L_1 \not\leq_s L_2 \quad &:= \quad \neg(L_1 \leq_s L_2) \\
&\Leftrightarrow \quad \forall c \in \mathbb{N} : \exists e_2 \in L_2 : \forall e_1 \in L_1 : e_1 \cong e_2 \rightarrow \mathit{size}(e_1) > c \cdot \mathit{size}(e_2)
\end{aligned}
$$

$$
\begin{aligned}
L_1 =_s L_2 \quad &:= \quad L_1 \leq_s L_2 \wedge L_2 \leq_s L_1 \\
L_1 <_s L_2 \quad &:= \quad L_1 \leq_s L_2 \wedge L_2 \not\leq_s L_1
\end{aligned}
$$

Properties

- $\leq_s$ is a partial order
- $<_s$ is a strict partial order
- $=_s$ is an equivalence relation
- $\leq_s$ and $<_s$ are not total (i.e., there are uncomparable languages)

All of these are already verified using Agda (in our Vatras library)

Current Progress

legend:

2CC	binary choice calculus
NCC	n-ary choice calculus
CCC	core choice calculus
ADT	algebraic decision trees
FST	feature structure trees

- 2CC $=_s$ CCC $\leq_s$ NCC
- 2CC $<_s$ ADT
- 2CC $\not\leq_s$ FST
- FST $\not\leq_s$ 2CC (because FST is incomplete)

All of these are already verified using Agda (in our Vatras library)

Current Progress

legend:

2CC	binary choice calculus
NCC	n-ary choice calculus
CCC	core choice calculus
ADT	algebraic decision trees
FST	feature structure trees

- 2CC $=_s$ CCC $\leq_s$ NCC
- 2CC $<_s$ ADT
- 2CC $\not\leq_s$ FST
- FST $\not\leq_s$ 2CC (because FST is incomplete)
  $\forall c \in \mathbb{N} : \exists e_2 \in \text{2CC} : \forall e_1 \in \text{FST} : e_1 \cong e_2 \rightarrow \mathit{size}(e_1) > c \cdot \mathit{size}(e_2)$

All of these are already verified using Agda (in our Vatras library)

Future Work

open questions:

- How to handle incomplete languages?
  $$\forall c \in \mathbb{N} : \exists e_2 \in L_2 : (\forall e_1 \in L_1 : e_1 \cong e_2 \rightarrow \mathit{size}(e_1) > c \cdot \mathit{size}(e_2)) \land (\exists e_1 \in L_1 : e_1 \cong e_2)$$
  breaks transitivity
- Do I need to consider feature model? (i.e., do feature models make a difference?)

future work:

- explore relationships between variability languages using $\leq_s$
- explore the influence of sharing primitives (i.e., subexpression reuse)

Questions?

open questions:

- How to handle incomplete languages?
  $$\forall c \in \mathbb{N} : \exists e_2 \in L_2 : (\forall e_1 \in L_1 : e_1 \cong e_2 \rightarrow size(e_1) > c \cdot size(e_2)) \land (\exists e_1 \in L_1 : e_1 \cong e_2)$$
  breaks transitivity
- Do I need to consider feature model? (i.e., do feature models make a difference?)

recap of the main definition:

$$L_1 \leq_s L_2 \quad := \quad \exists c \in \mathbb{N} : \forall e_2 \in L_2 : \exists e_1 \in L_1 : e_1 \cong e_2 \land size(e_1) \leq c \cdot size(e_2)$$
$$L_1 \not\leq_s L_2 \quad := \quad \forall c \in \mathbb{N} : \exists e_2 \in L_2 : \forall e_1 \in L_1 : e_1 \cong e_2 \rightarrow size(e_1) > c \cdot size(e_2)$$