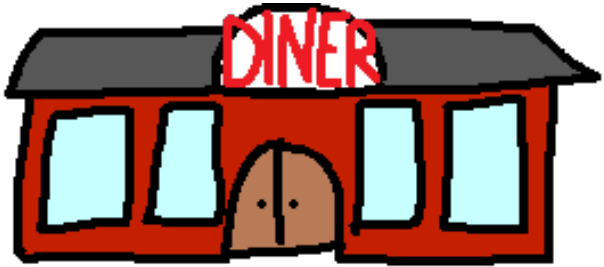
On the Expressive Power of Languages for Static Variability

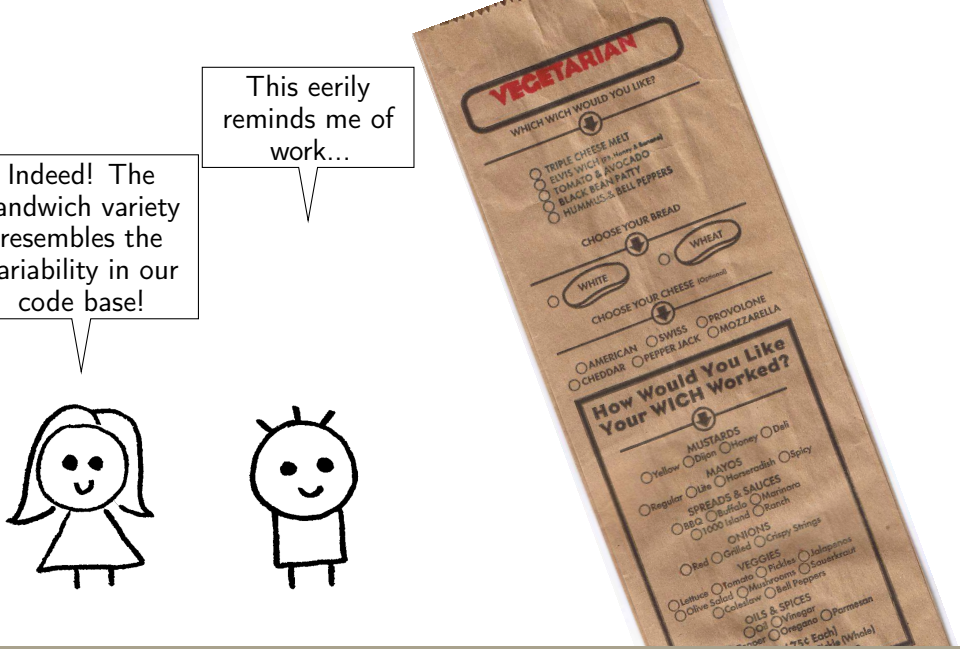
Paul Bittner, A. Schultheiß, B. Moosherr, J. Young, L. Teixeira, E. Walkingshaw, P. Ataei, T. Thüm | Oct 25 | OOPSLA

PADERBORN UNIVERSITY · universität ulm · INPUT | OUTPUT · Universidade Federal de Pernambuco · Technische Universität Braunschweig


DINER


VEGETARIAN
WHICH WICH WOULD YOU LIKE?
TRIPLE CHEESE MELT
ELVIS WICH (Peanut Butter & Banana)
TOMATO & AVOCADO
BLACK BEAN PATTY
HUMMUS & BELL PEPPERS
CHOOSE YOUR BREAD
WHEAT
WHITE
CHOOSE YOUR CHEESE (Optional)
AMERICAN
SWISS
PROVOLONE
CHEDDAR
PEPPER JACK
MOZZARELLA
How Would You Like Your WICH Worked?
MUSTARDS
Yellow
Dijon
Honey
Deli
MAYOS
Regular
Lite
Horseradish
Spicy
SPREADS & SAUCES
BBQ
Buffalo
Marinara
1000 Island
Ranch
ONIONS
Red
Grilled
Crispy Strings
VEGGIES
Lettuce
Tomato
Pickles
Jalapenos
Olive Salad
Mushrooms
Beans
Sauerkraut
Coleslaw
Bell Peppers
OILS & SPICES
Oil
Vinegar
Oregano
Parmesan

This eerily reminds me of work...


This eerily reminds me of work...
Indeed! The sandwich variety resembles the variability in our code base!
VEGETARIAN
WHICH WICH WOULD YOU LIKE?
TRIPLE CHEESE MELT
ELVIS WICH (Ft. Nutter & Bananas)
TOMATO & AVOCADO
BLACK BEAN PATTY
HUMMUS & BELL PEPPERS
CHOOSE YOUR BREAD
WHEAT
WHITE
CHOOSE YOUR CHEESE (Optional)
AMERICAN
SWISS
PROVOLONE
CHEDDAR
PEPPER JACK
MOZZARELLA
How Would You Like Your WICH Worked?
MUSTARDS
Yellow
Dijon
Honey
Deli
MAYOS
Regular
Lite
Horseradish
Spicy
SPREADS & SAUCES
BBQ
Buffalo
Marinara
1000 Island
Ranch
ONIONS
Red
Grilled
Crispy Strings
VEGGIES
Lettuce
Tomato
Pickles
Jalapenos
Olive Salad
Mushrooms
Bell Peppers
Coleslaw
Sauerkraut
OILS & SPICES
Oil
Vinegar
Oregano
Parmesan

Inde
sandw
reser
variab
cod

```
  static void
f_foreground(/* params */)
{
#ifdef FEAT_GUI
  if (gui.in_use)
  gui_mch_set_foreground();
#else
# ifdef MSWIN
  win32_set_foreground();
# endif
#endif
}
```

Vim Commit ab4cece

Inde
sandw
reser
variab
cod

```
   static void
f_foreground(/* params */)
{
#ifdef FEAT_GUI
   if (gui.in_use)
   gui_mch_set_foreground();
#else
# ifdef MSWIN
   win32_set_foreground();
# endif
#endif
}
```

#define FEAT_GUI 1

```
   static void
f_foreground(/* params */)
{
   if (gui.in_use)
   gui_mch_set_foreground();
}
```

Vim Commit ab4cece

Inde
sandw
reser
variab
cod

```
   static void
f_foreground(/* params */)
{
#ifdef FEAT_GUI
   if (gui.in_use)
   gui_mch_set_foreground();
#else
# ifdef MSWIN
   win32_set_foreground();
# endif
#endif
}
```

#define FEAT_GUI 1

```
   static void
f_foreground(/* params */)
{
   if (gui.in_use)
   gui_mch_set_foreground();
}
```

#define FEAT_GUI 0,
#define MSWIN 1

```
   static void
f_foreground(/* params */)
{
   win32_set_foreground();
}
```

Vim Commit ab4cece

Inde
sandw
reser
variab
cod

```
   static void
f_foreground(/* params */)
{
#ifdef FEAT_GUI
   if (gui.in_use)
   gui_mch_set_foreground();
#else
# ifdef MSWIN
   win32_set_foreground();
# endif
#endif
}
```

#define FEAT_GUI 1

```
   static void
f_foreground(/* params */)
{
   if (gui.in_use)
   gui_mch_set_foreground();
}
```

#define FEAT_GUI 0,
#define MSWIN 1

```
   static void
f_foreground(/* params */)
{
   win32_set_foreground();
}
```

#define FEAT_GUI 0,
#define MSWIN 0

```
   static void
f_foreground(/* params */)
{
}
```

Vim Commit ab4cece

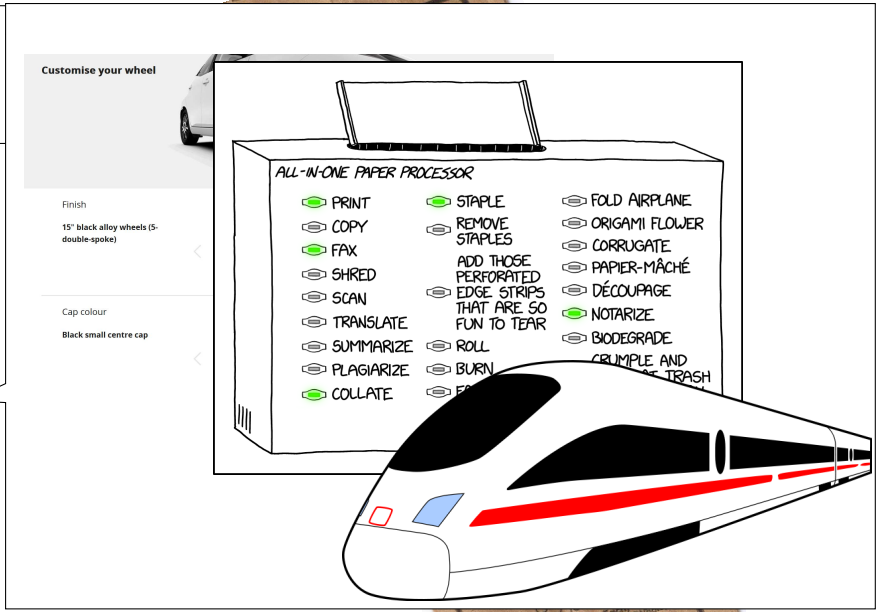

And its not just our code and sandwiches, I guess...
VEGETARIAN
WHICH WICH WOULD YOU LIKE?
TRIPLE CHEESE MELT
ELVIS WICH (Peanut Butter & Banana)
TOMATO & AVOCADO
BLACK BEAN PATTY
HUMMUS & BELL PEPPERS
CHOOSE YOUR BREAD
WHEAT
WHITE
CHOOSE YOUR CHEESE (Optional)
AMERICAN
SWISS
PROVOLONE
CHEDDAR
PEPPER JACK
MOZZARELLA
How Would You Like Your WICH Worked?
MUSTARDS
Yellow
Dijon
Honey
Deli
MAYOS
Regular
Lite
Horseradish
Spicy
SPREADS & SAUCES
BBQ
Buffalo
Marinara
1000 Island
Ranch
ONIONS
Red
Grilled
Crispy Strings
VEGGIES
Lettuce
Tomato
Pickles
Jalapenos
Olive Salad
Mushrooms
Bell Peppers
Coleslaw
Sauerkraut
OILS & SPICES
Oil
Vinegar
Oregano
Parmesan

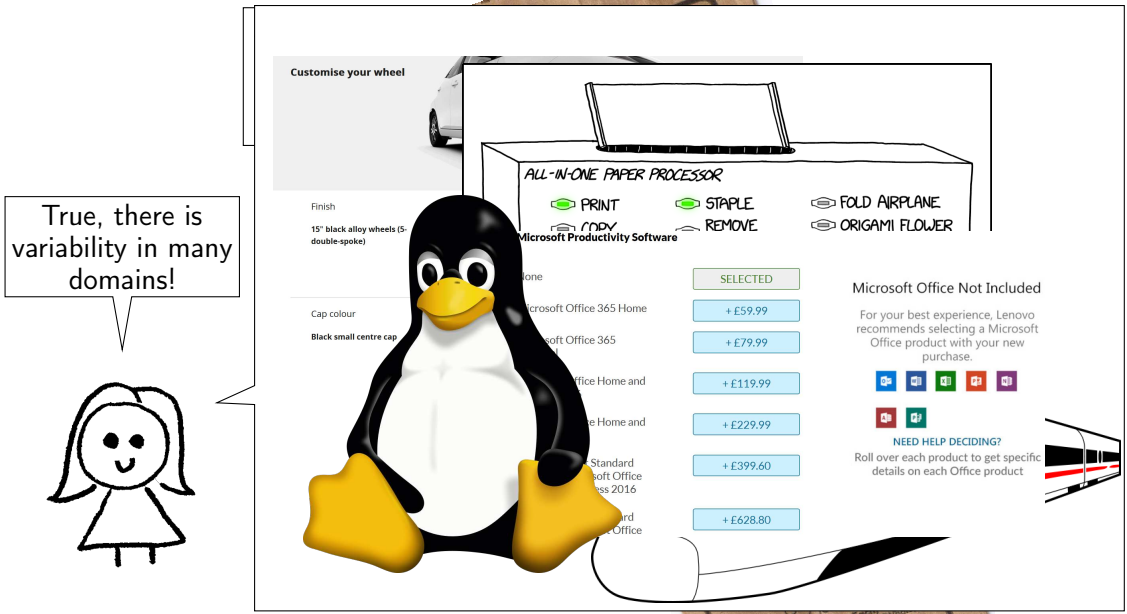
True, there is variability in many domains!

True, there is variability in many domains!

True, there is variability in many domains!

True, there is variability in many domains!




True, there is variability in many domains!
Customise your wheel
Finish
15" black alloy wheels (5-double-spoke)
Cap colour
Black small centre cap
ALL-IN-ONE PAPER PROCESSOR
PRINT
STAPLE
FOLD AIRPLANE
REMOVE
ORIGAMI FLOWER
Microsoft Productivity Software
SELECTED
+ £59.99
+ £79.99
+ £119.99
+ £229.99
+ £399.60
+ £628.80
Microsoft Office Not Included
For your best experience, Lenovo recommends selecting a Microsoft Office product with your new purchase.
NEED HELP DECIDING?
Roll over each product to get specific details on each Office product

But how to *describe* and *analyze* variability across domains?

	Computation	Static Variability
Implementation		
Specification		

	Computation	Static Variability
Implementation	Java, C, Haskell, OCaml, Prolog, ...	
Specification		

	Computation	Static Variability
Implementation	Java, C, Haskell, OCaml, Prolog, ...	
Specification	Turing Machine, Lambda Calculus, Type-0 Grammar, ...	

	Computation	**Static Variability**
Implementation	Java, C, Haskell, OCaml, Prolog, ...	C Preprocessor, Mixins, Aspects, Sandwich Menus, ...
Specification	Turing Machine, Lambda Calculus, Type-0 Grammar, ...	

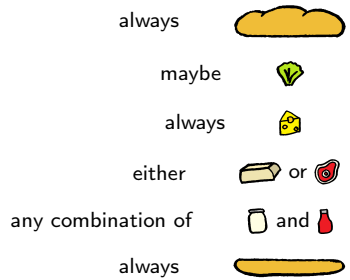
	Computation	**Static Variability**
Implementation	Java, C, Haskell, OCaml, Prolog, ...	C Preprocessor, Mixins, Aspects, Sandwich Menus, ...
Specification	Turing Machine, Lambda Calculus, Type-0 Grammar, ...	?

Core Choice Calculus [Erwig and Walkingshaw, 2011] [Walkingshaw, 2013] – A Lambda Calculus of Variability?

$$
\begin{array}{rcll}
e & ::= & a\langle e, \ldots, e\rangle & \textit{Object Structure} \\
 & | & D\langle e, \ldots, e\rangle & \textit{Choice}
\end{array}
$$

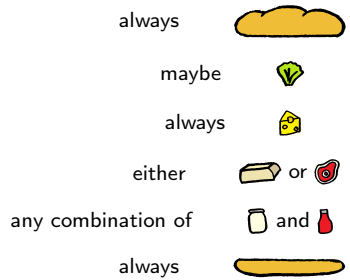
Core Choice Calculus [Erwig and Walkingshaw, 2011] [Walkingshaw, 2013] – A Lambda Calculus of Variability?

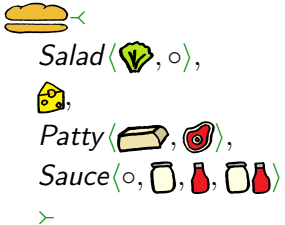
$$
\begin{array}{rcll}
e & ::= & a\langle\!\langle e, \ldots, e \rangle\!\rangle & \textit{Object Structure} \\
 & | & D\langle e, \ldots, e \rangle & \textit{Choice}
\end{array}
$$



Core Choice Calculus [Erwig and Walkingshaw, 2011] [Walkingshaw, 2013] – A Lambda Calculus of Variability?

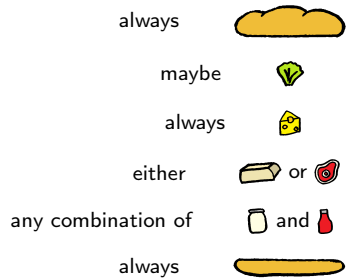
$$
\begin{array}{rcll}
e & ::= & a\prec e, \ldots, e\succ & \textit{Object Structure} \\
 & \mid & D\langle e, \ldots, e\rangle & \textit{Choice}
\end{array}
$$





Core Choice Calculus [Erwig and Walkingshaw, 2011] [Walkingshaw, 2013] – A Lambda Calculus of Variability?

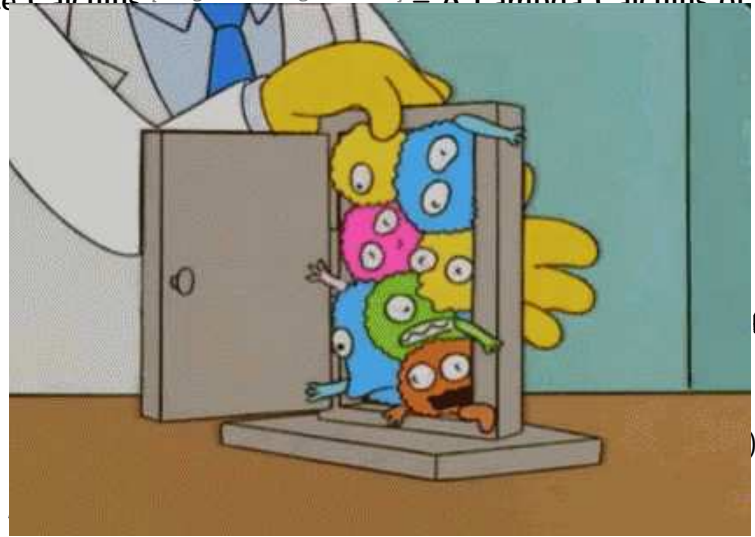
$$
\begin{array}{rcll}
e & ::= & a\langle\!\langle e, \ldots, e \rangle\!\rangle & \textit{Object Structure} \\
 & \mid & D\langle e, \ldots, e \rangle & \textit{Choice}
\end{array}
$$



$$
[\![\langle\!\langle \textit{Salad}\langle \text{🥬}, \circ \rangle, \text{🧀}, \textit{Patty}\langle \text{🥩}, \text{🍅} \rangle, \textit{Sauce}\langle \circ, \text{🫙}, \text{🍾}, \text{🫙}, \text{🍾} \rangle \rangle\!\rangle]\!](c)
$$

$$
= \quad \text{if } c(\textit{Salad}) = 0,\ c(\textit{Patty}) = 0,\ c(\textit{Sauce}) = 2.
$$

Core Choice Calculus [Erwig and Walkingshaw, 2011] – A Lambda Calculus of Variability?



always

maybe

always

either

any combination of

always

$= 0,$

$= 0,$

$= 2.$

Choice Calculus
[Erwig and Walkingshaw, 2011]
[Walkingshaw, 2013]

Artifact Trees
[Linsbauer et al., 2017]

Feature Structure Trees
[Apel et al., 2013]

Algebraic Decision Trees
[Bahar et al., 1993]

Gruler's Language
[Gruler, 2010]

Clone and Own

Variability-Aware ASTs
[Kästner et al., 2008]

Variational IMP
[Midtgaard et al., 2015]

. . .

But how do these languages relate? Can we transfer research results based on one formalism to the others?

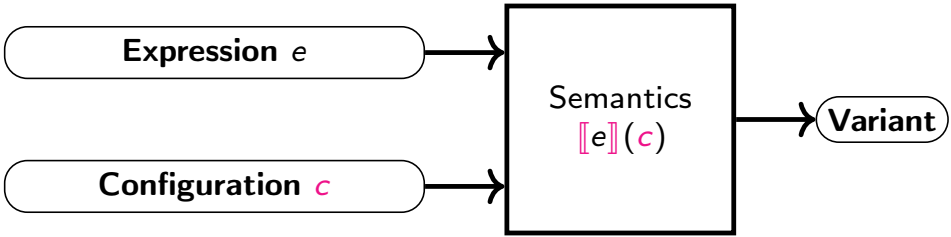
Variability-Aware ASTs

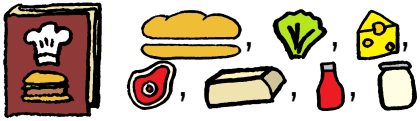
[Kästner et al., 2008]

Variational IDP

[Midtgaard

. . .


Expression e
Configuration c
Semantics
$[\![e]\!](c)$
Variant



Expression e

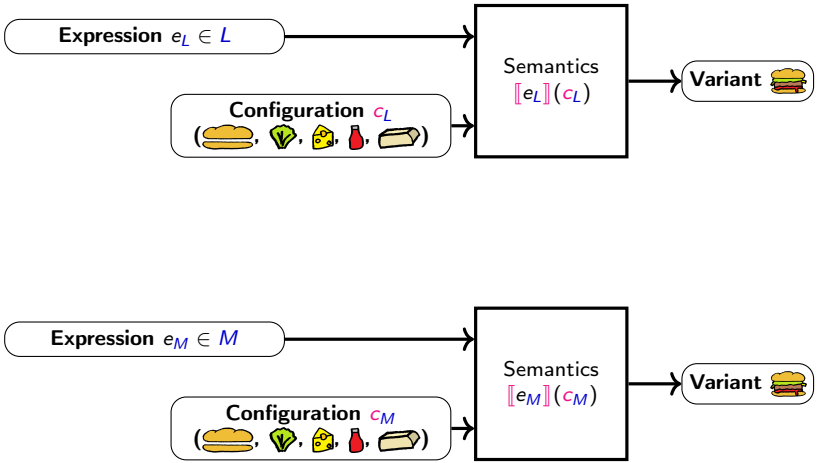
Configuration c

Semantics

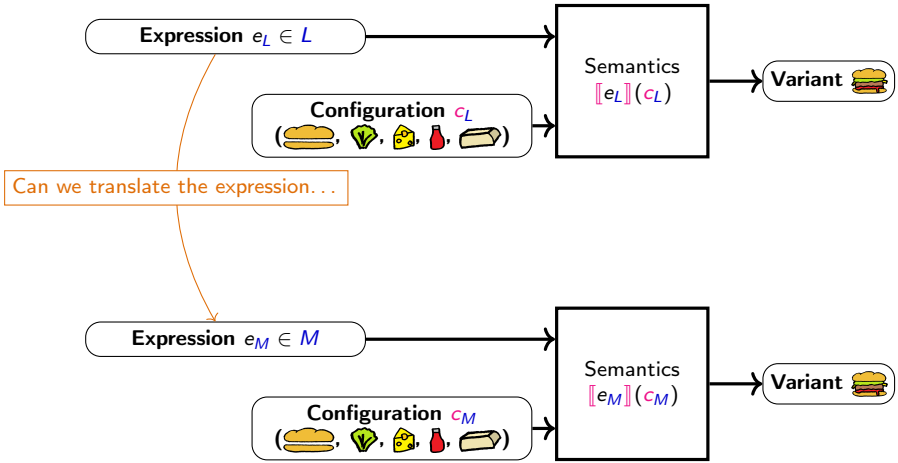
Variant

How to Compare Variability Languages L and M semantically?

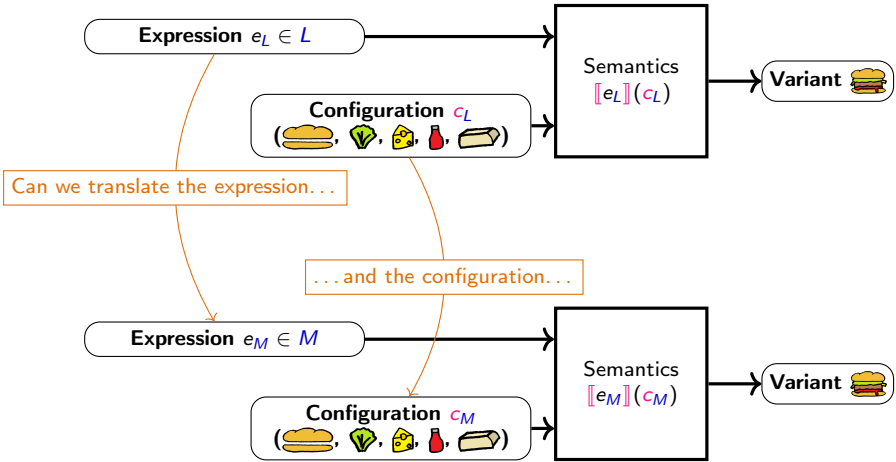
How to Compare Variability Languages L and M semantically?



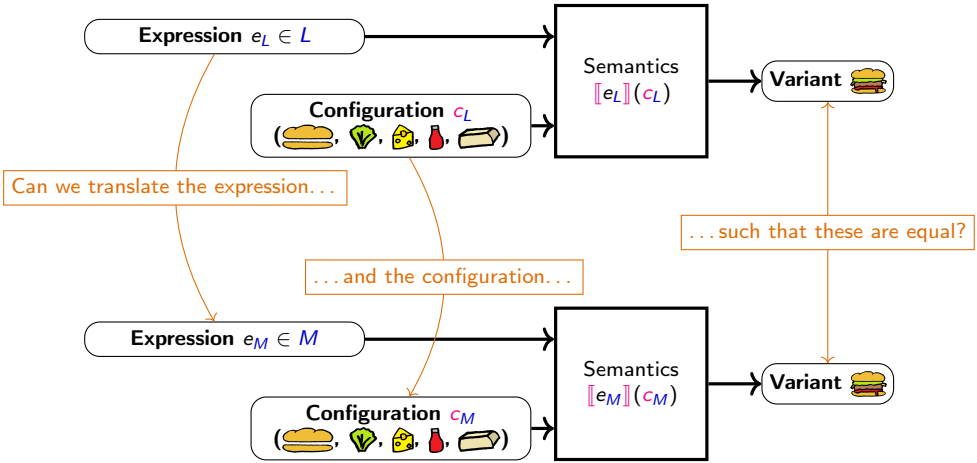
How to Compare Variability Languages L and M semantically?



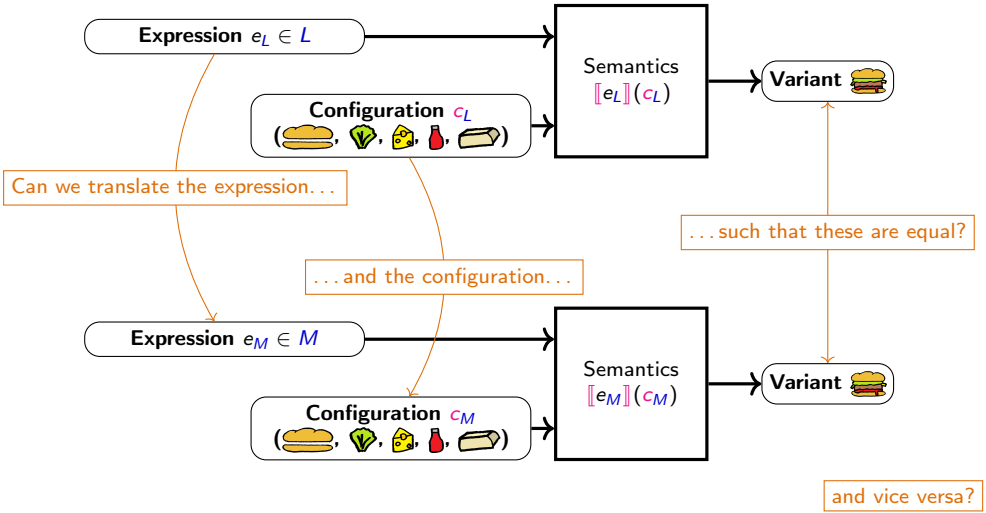
How to Compare Variability Languages L and M semantically?



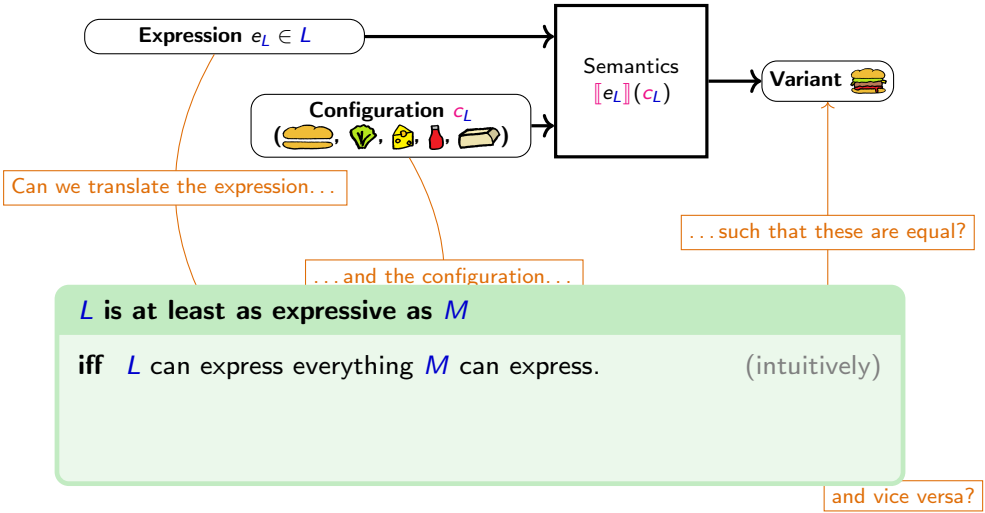
How to Compare Variability Languages L and M semantically?



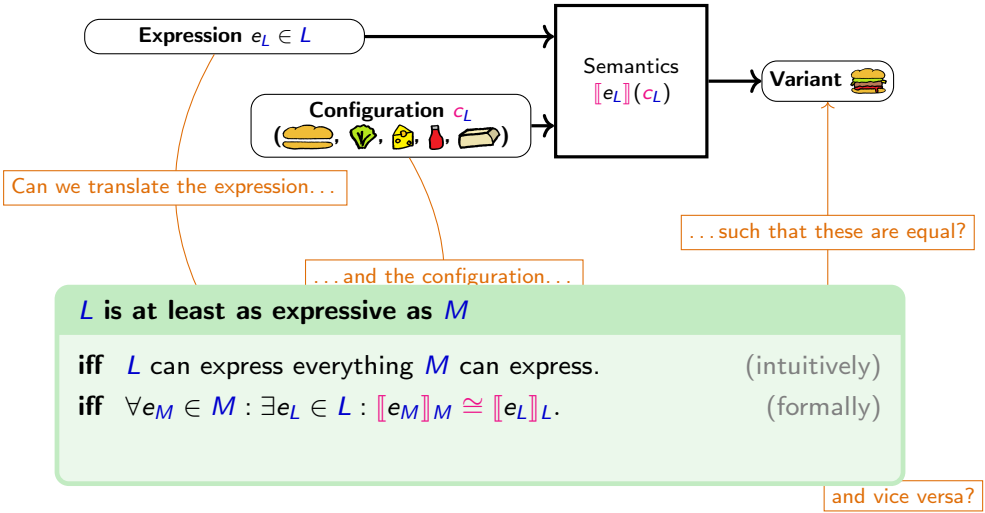
How to Compare Variability Languages L and M semantically?



How to Compare Variability Languages L and M semantically?



How to Compare Variability Languages L and M semantically?

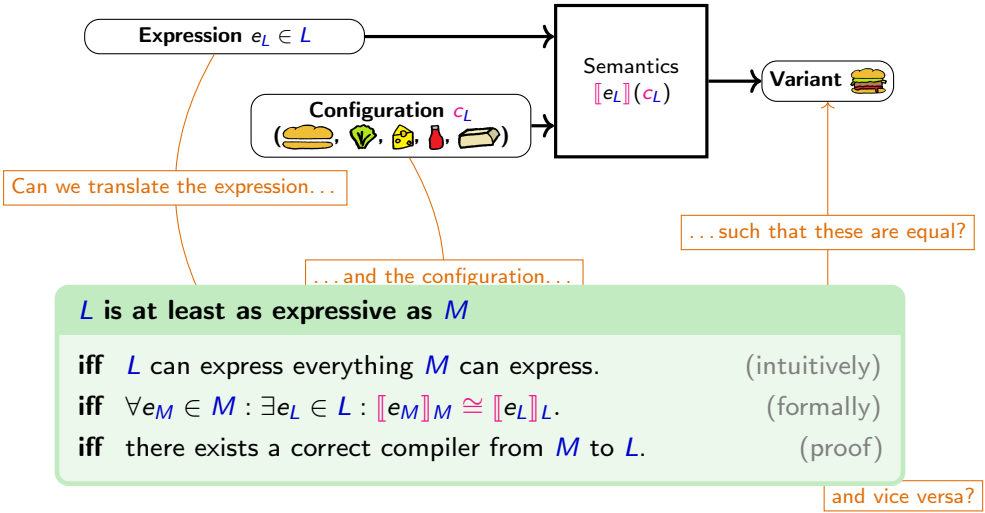


L is at least as expressive as M

iff L can express everything M can express. (intuitively)

iff $\forall e_M \in M : \exists e_L \in L : [\![e_M]\!]_M \cong [\![e_L]\!]_L$. (formally)

How to Compare Variability Languages L and M semantically?



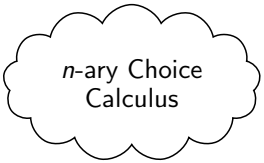
Binary Choice Calculus

Option Calculus

Algebraic Decision Trees

Core Choice Calculus

Feature Structure Trees



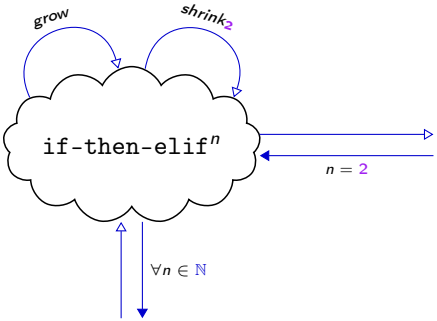
Binary Choice
Calculus

Option
Calculus

Algebraic
Decision Trees

Core Choice
Calculus

Feature
Structure Trees

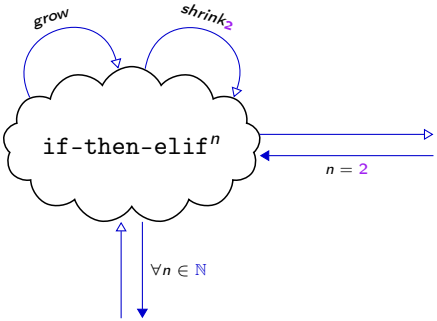


if-then-else

Algebraic
Decision Trees

Option
Calculus

Feature
Structure Trees

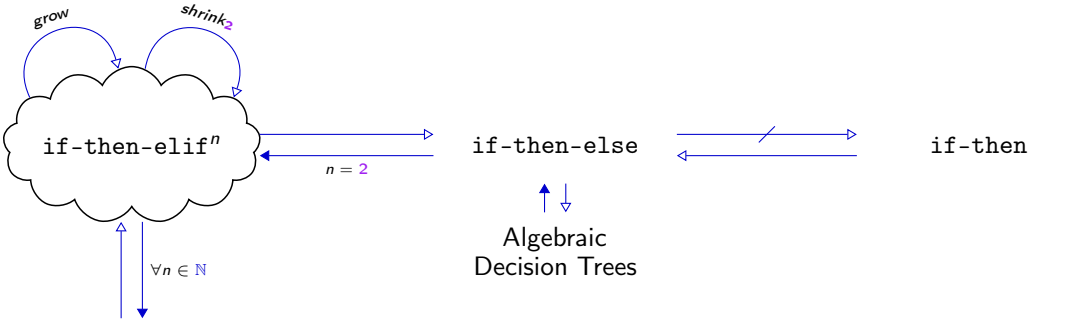


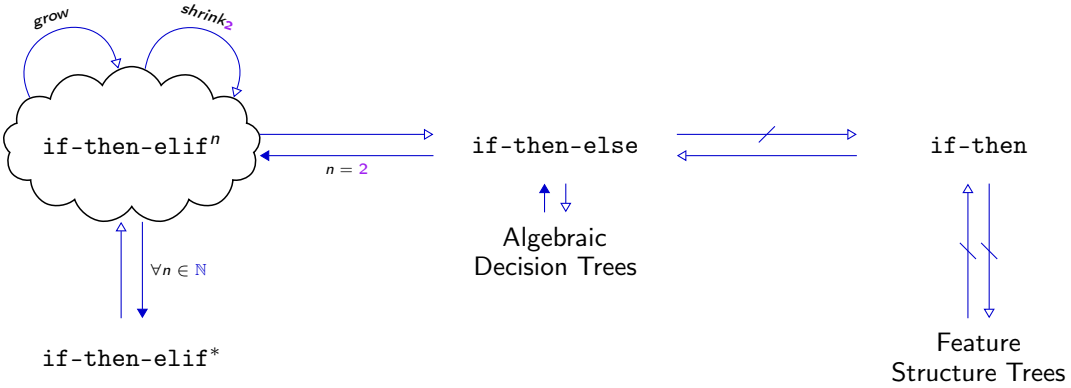
`if-then-else`

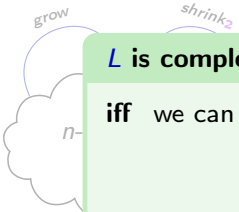
Algebraic
Decision Trees

Option
Calculus

Feature
Structure Trees


grow
shrink2
if-then-elif^n
n = 2
if-then-else
if-then
Algebraic
Decision Trees
∀n ∈ ℕ
if-then-elif*
Feature
Structure Trees


grow
shrink2
if-then-elif^n
n = 2
∀n ∈ ℕ
if-then-elif*
if-then-else
Algebraic
Decision Trees
if-then
Feature
Structure Trees



L is complete

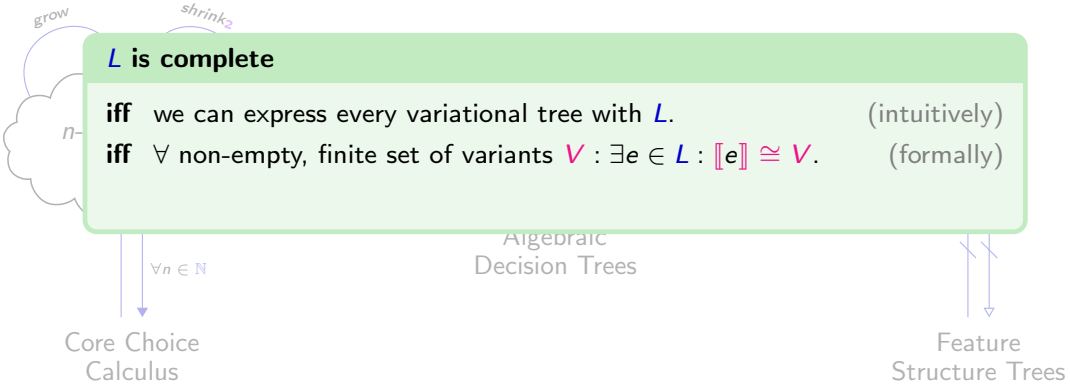
iff we can express every variational tree with L. (intuitively)

$\forall n \in \mathbb{N}$

Algebraic Decision Trees

Core Choice Calculus

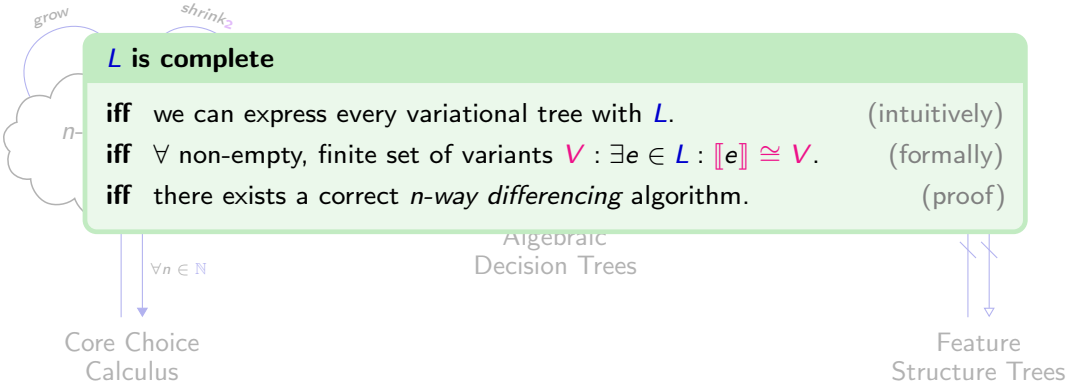
Feature Structure Trees

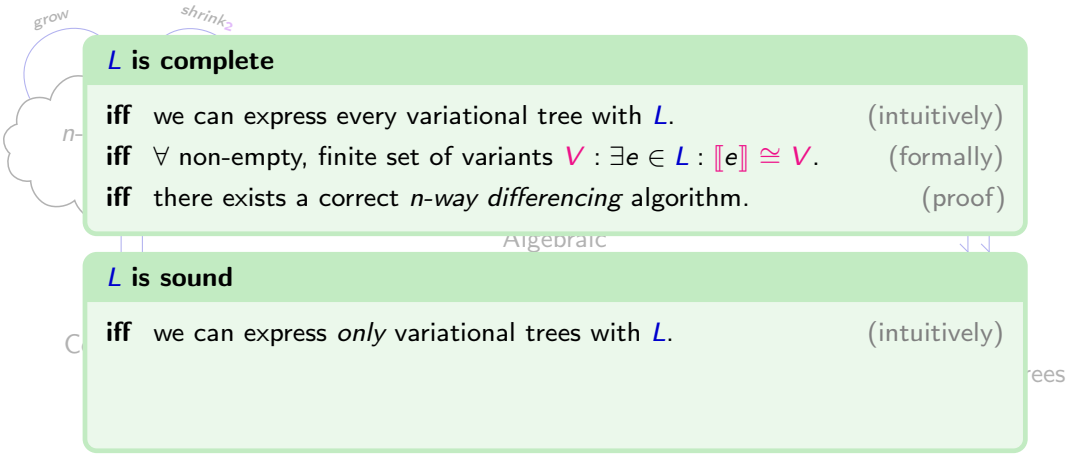


L is complete

iff we can express every variational tree with L. (intuitively)

iff $\forall$ non-empty, finite set of variants $V : \exists e \in L : [\![e]\!] \cong V$. (formally)


grow
shrink2
n-
L is complete
iff we can express every variational tree with L. (intuitively)
iff $\forall$ non-empty, finite set of variants $V : \exists e \in L : [\![e]\!] \cong V$. (formally)
iff there exists a correct *n-way differencing* algorithm. (proof)
$\forall n \in \mathbb{N}$
Algebraic
Decision Trees
Core Choice
Calculus
Feature
Structure Trees



L is complete

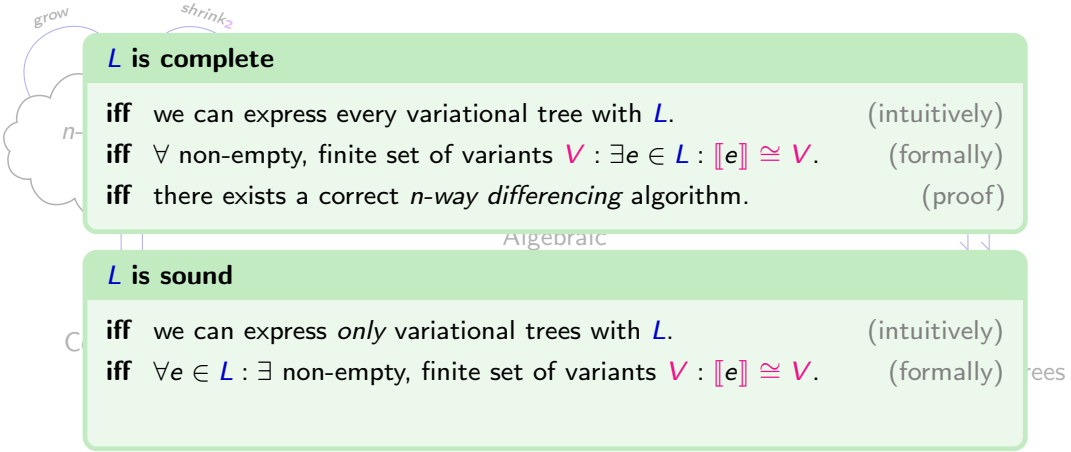
iff we can express every variational tree with L. (intuitively)

iff $\forall$ non-empty, finite set of variants $V : \exists e \in L : [\![e]\!] \cong V$. (formally)

iff there exists a correct *n-way differencing* algorithm. (proof)

L is sound

iff we can express *only* variational trees with L. (intuitively)



L is complete

iff we can express every variational tree with L. (intuitively)

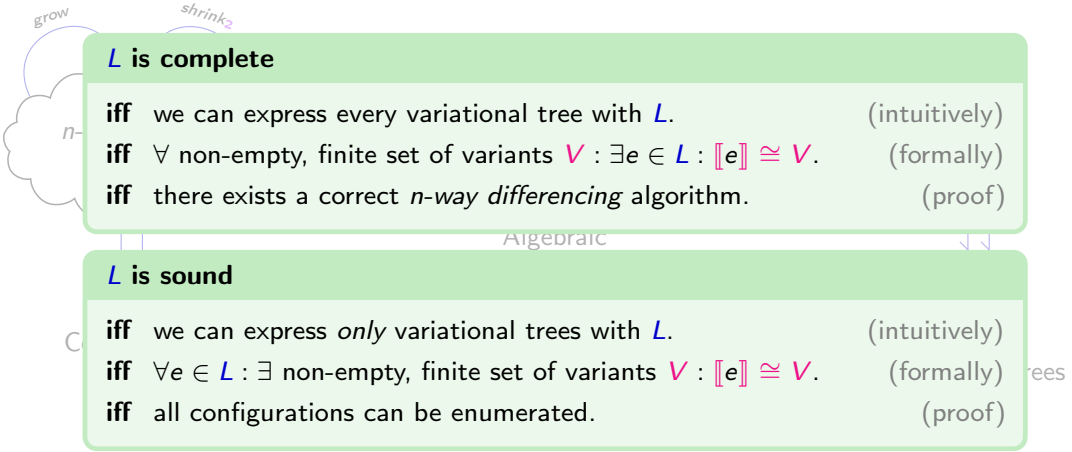
iff $\forall$ non-empty, finite set of variants $V : \exists e \in L : [\![e]\!] \cong V$. (formally)

iff there exists a correct *n-way differencing* algorithm. (proof)

L is sound

iff we can express *only* variational trees with L. (intuitively)

iff $\forall e \in L : \exists$ non-empty, finite set of variants $V : [\![e]\!] \cong V$. (formally)



L is complete

iff we can express every variational tree with L. (intuitively)

iff $\forall$ non-empty, finite set of variants $V : \exists e \in L : [\![e]\!] \cong V$. (formally)

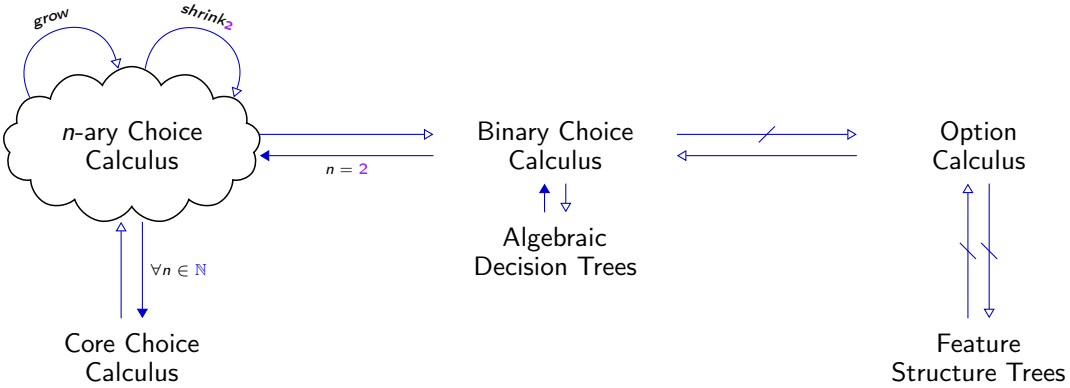
iff there exists a correct *n-way differencing* algorithm. (proof)

L is sound

iff we can express *only* variational trees with L. (intuitively)

iff $\forall e \in L : \exists$ non-empty, finite set of variants $V : [\![e]\!] \cong V$. (formally)

iff all configurations can be enumerated. (proof)


grow
shrink2
n-ary Choice Calculus
n = 2
∀n ∈ ℕ
Core Choice Calculus
Binary Choice Calculus
Algebraic Decision Trees
Option Calculus
Feature Structure Trees

Binary Choice Calculus

Algebraic Decision Trees

Clone and Own

Option Calculus

Feature Structure Trees

Binary Choice Calculus

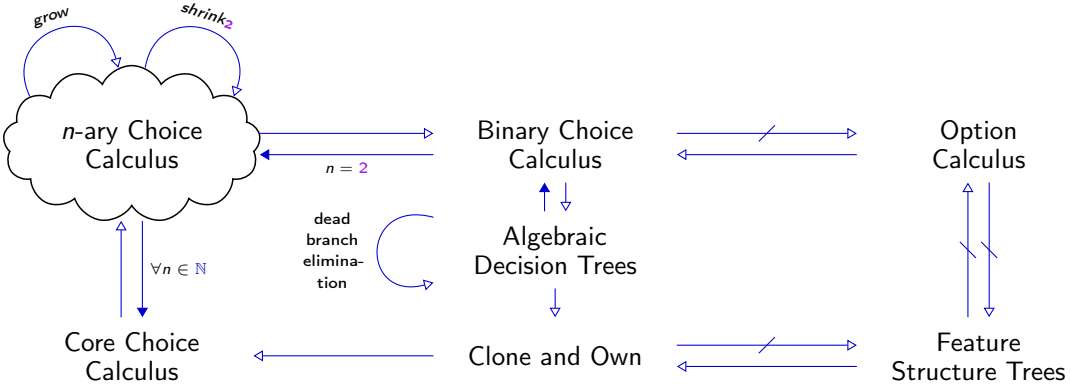
Option Calculus

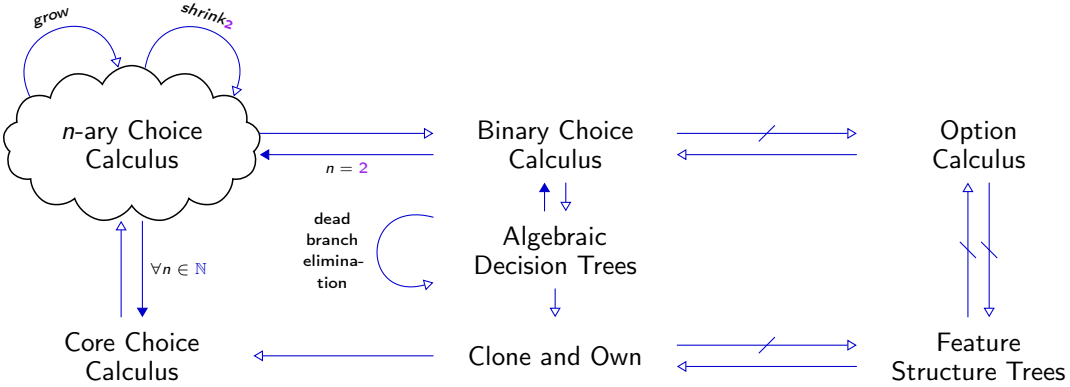
Algebraic Decision Trees

Feature Structure Trees

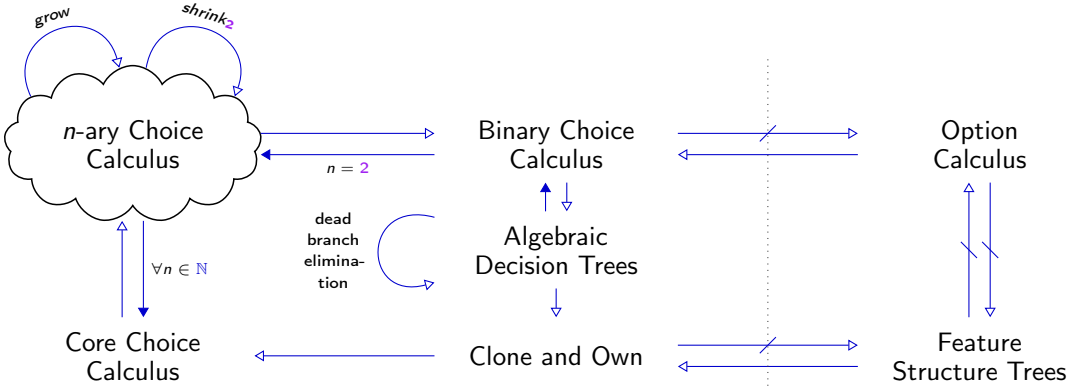
Clone and Own

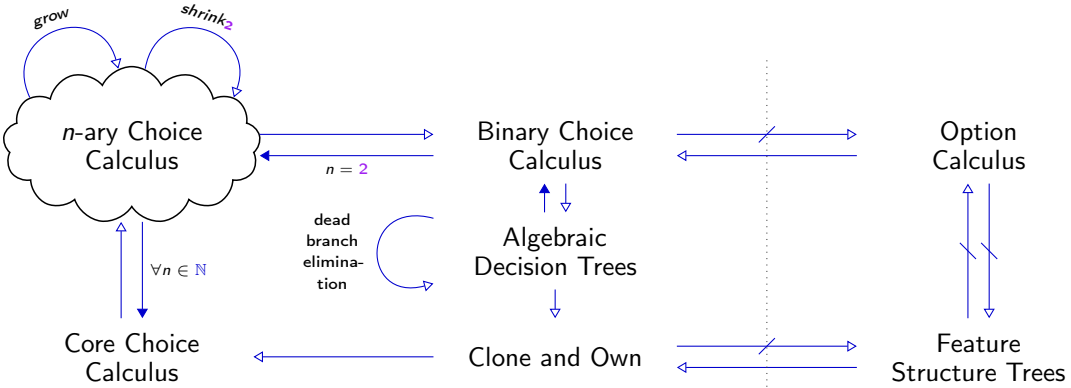
We prove this sound and complete.


grow
shrink2
n-ary Choice Calculus
n = 2
Binary Choice Calculus
Option Calculus
dead branch elimina-tion
Algebraic Decision Trees
∀n ∈ ℕ
Core Choice Calculus
Clone and Own
Feature Structure Trees

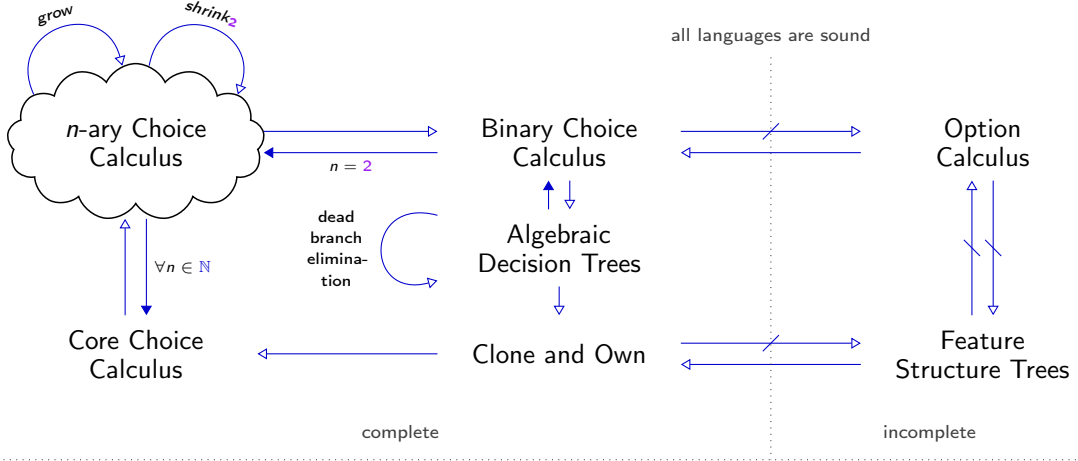


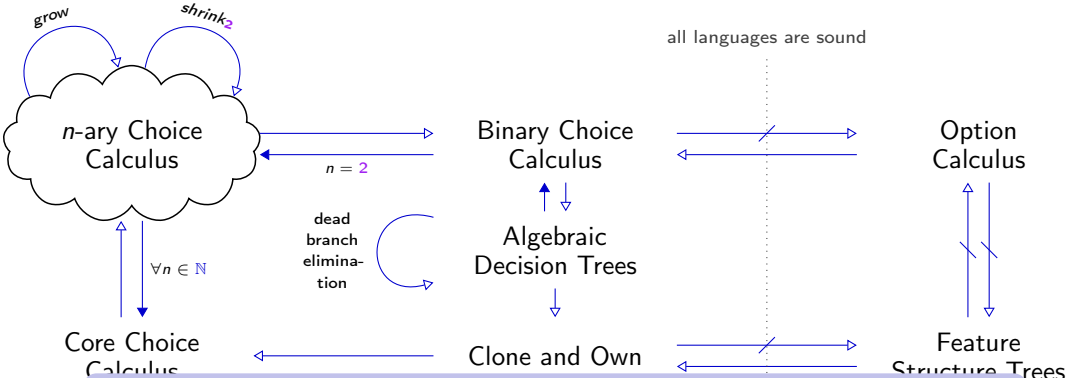
$$\frac{\mathsf{Complete}(L) \qquad M \succeq L}{\mathsf{Complete}(M)}$$


grow
shrink2
n-ary Choice Calculus
n = 2
dead branch elimina-tion
∀n ∈ ℕ
Core Choice Calculus
Binary Choice Calculus
Algebraic Decision Trees
Clone and Own
Option Calculus
Feature Structure Trees
complete
incomplete



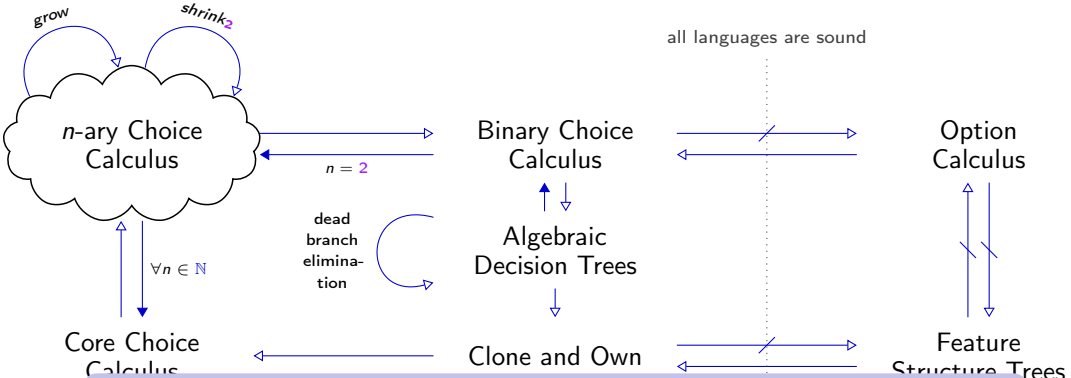
$$\frac{\mathsf{Sound}(L) \qquad L \succeq M}{\mathsf{Sound}(M)}$$


grow
shrink2
n-ary Choice Calculus
n = 2
dead branch elimina-tion
∀n ∈ ℕ
Core Choice Calculus
Binary Choice Calculus
Algebraic Decision Trees
Clone and Own
all languages are sound
Option Calculus
Feature Structure Trees
complete
incomplete



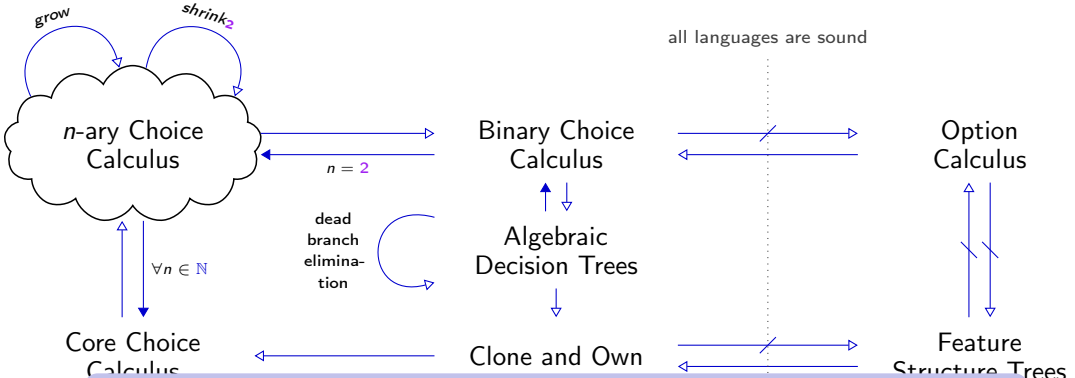
Conclusions

- Choice calculus is indeed a lambda calculus of variation



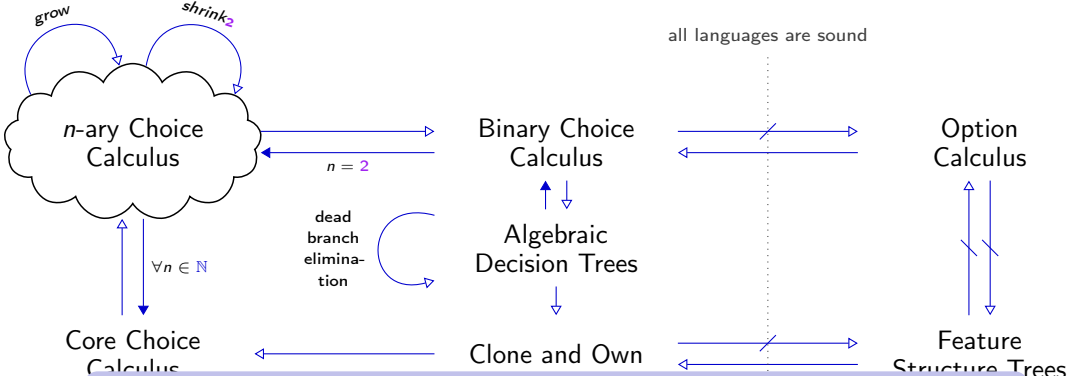
Conclusions

- Choice calculus is indeed a lambda calculus of variation as well as algebraic decision trees and clone-and-own.



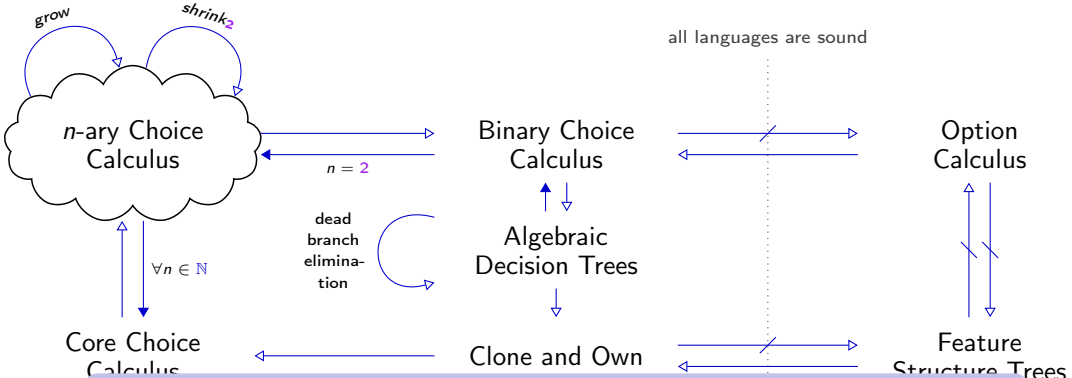
Conclusions

- Choice calculus is indeed a lambda calculus of variation as well as algebraic decision trees and clone-and-own.
- There are ≥ 3 classes of expressiveness, arising from different syntactical restrictions.



Conclusions

- Choice calculus is indeed a lambda calculus of variation as well as algebraic decision trees and clone-and-own. $\Rightarrow$ **Are all languages equally good?**
- There are ≥ 3 classes of expressiveness, arising from different syntactical restrictions.

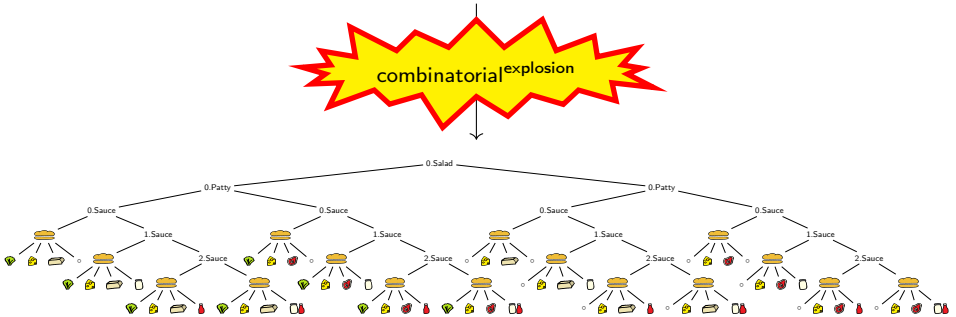


Conclusions

- Choice calculus is indeed a lambda calculus of variation as well as algebraic decision trees and clone-and-own. ⇒ **Are all languages equally good?**
- There are ≥ 3 classes of expressiveness, arising from different syntactical restrictions. ⇒ **What is the value of incomplete languages?**

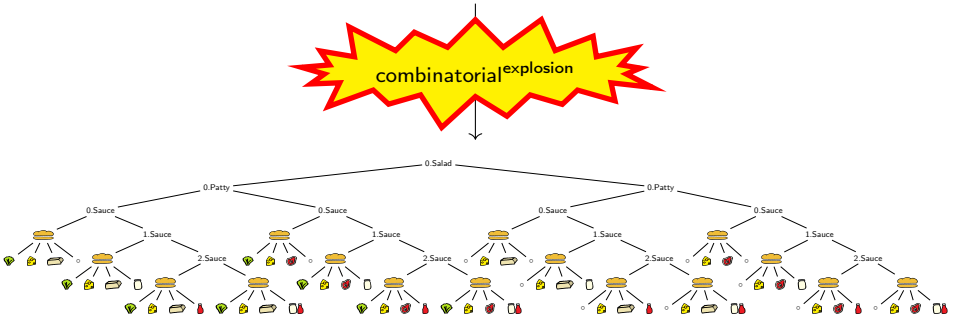
Are all complete languages equally useful?

Are all complete languages equally useful?



for algebraic decision trees and clone-and-own

Are all complete languages equally useful?

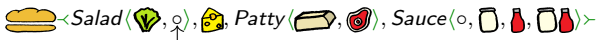


for algebraic decision trees and clone-and-own

No, avoiding duplication of shared sub-terms is essential for scalability and usability.
(known problem of clone-and-own and a basic principle of software engineering)

What is the value of incomplete languages?

What is the value of incomplete languages?



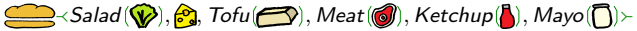
Encoding options with choices requires context-sensitive neutral values ○

What is the value of incomplete languages?

Encoding options with choices requires context-sensitive neutral values ○ or sacrifices on sharing:

What is the value of incomplete languages?

Encoding options with choices requires context-sensitive neutral values ○ or sacrifices on sharing:



Options avoid these drawbacks

What is the value of incomplete languages?

Encoding options with choices requires context-sensitive neutral values ○ or sacrifices on sharing:

Options avoid these drawbacks but cannot express alternatives! (That is why they are incomplete.)

What is the value of incomplete languages?

Encoding options with choices requires context-sensitive neutral values ○ or sacrifices on sharing:

Options avoid these drawbacks but cannot express alternatives! (That is why they are incomplete.)

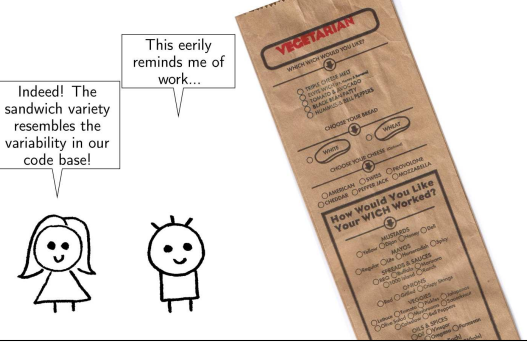
Conclusion: Use choices and options for completeness and usability.

***Everything* is formalized in Agda ...**

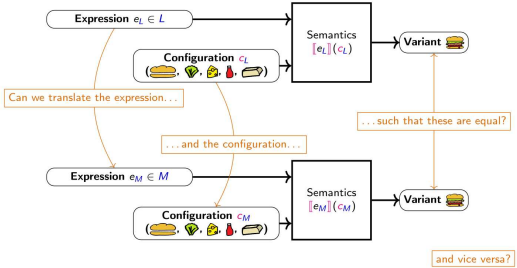
... including *all* languages, definitions, theorems, and proofs,

... including tutorials, examples, and a demo,

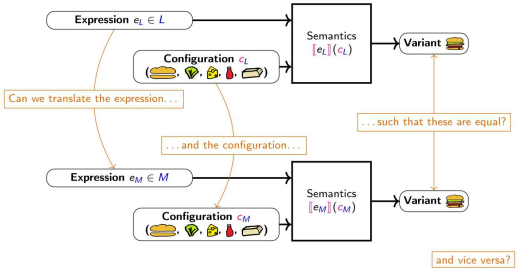
... and where proofs come as executable compilers.

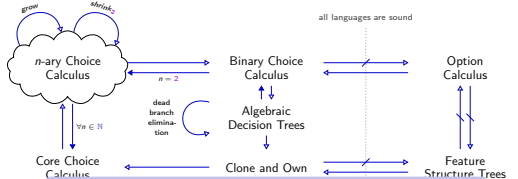

This eerily reminds me of work...
Indeed! The sandwich variety resembles the variability in our code base!
VEGETARIAN
How Would You Like Your WICH Worked?

How to Compare Variability Languages L and M semantically?



How to Compare Variability Languages L and M semantically?

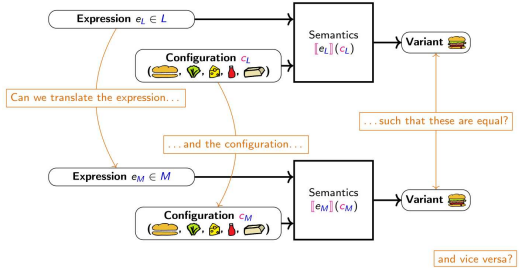


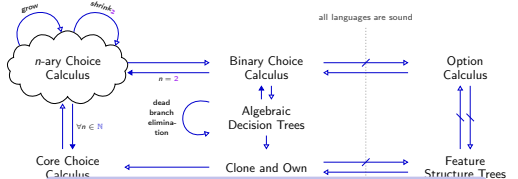


Conclusions

- Choice calculus is indeed a lambda calculus of variation as well as algebraic decision trees and clone-and-own.
- There are ≥ 3 classes of expressiveness, arising from different syntactical restrictions.

How to Compare Variability Languages L and M semantically?





Conclusions

- Choice calculus is indeed a lambda calculus of variation as well as algebraic decision trees and clone-and-own.
- There are ≥ 3 classes of expressiveness, arising from different syntactical restrictions.

Agda


Apel, S., Kästner, C., and Lengauer, C. (2013).
Language-Independent and Automated Software Composition: The FeatureHouse Experience.
IEEE Trans. on Software Engineering (TSE), 39(1):63–79.


Apel, S., Lengauer, C., Möller, B., and Kästner, C. (2010).
An Algebraic Foundation for Automatic Feature-Based Program Synthesis.
Science of Computer Programming (SCP), 75(11):1022–1047.


Bahar, R. I., Frohm, E. A., Gaona, C. M., Hachtel, G. D., Macii, E., Pardo, A., and Somenzi, F. (1993).
Algebraic Decision Diagrams and Their Applications.
In *Proc. Int'l Conf. on Computer-Aided Design (ICCAD)*, pages 188–191. IEEE.


Castro, T., Teixeira, L., Alves, V., Apel, S., Cordy, M., and Gheyi, R. (2021).
A Formal Framework of Software Product Line Analyses.
Trans. on Software Engineering and Methodology (TOSEM), 30(3).


Erwig, M. and Walkingshaw, E. (2011).
The Choice Calculus: A Representation for Software Variation.
Trans. on Software Engineering and Methodology (TOSEM), 21(1):6:1–6:27.


Gruler, A. (2010).
A Formal Approach to Software Product Families.
PhD thesis, TU München.


Kästner, C., Apel, S., and Kuhlemann, M. (2008).
Granularity in Software Product Lines.
In *Proc. Int'l Conf. on Software Engineering (ICSE)*, pages 311–320. ACM.


Linsbauer, L., Lopez-Herrejon, R. E., and Egyed, A. (2017).
Variability Extraction and Modeling for Product Variants.
Software and Systems Modeling (SoSyM), 16(4):1179–1199.


Midtgaard, J., Dimovski, A. S., Brabrand, C., and Wąsowski, A. (2015).
Systematic Derivation of Correct Variability-Aware Program Analyses.
Science of Computer Programming (SCP), 105:145–170.


Walkingshaw, E. (2013).
The Choice Calculus: A Formal Language of Variation.
PhD thesis, Oregon State University.