Perl vs. Python
Perl vs. Python

TIMTOWTDI

vs.

TIOOWTDI
Perl vs. Python

“There Is More Than One Way To Do It”

vs.

“There Is Only One Way To Do It”
Warmup

Arithmetic expressions over one variable $x$: meaning in $\mathbb{N} \rightarrow \mathbb{N}$

$$a, b ::= n \in \mathbb{N} \mid x \mid a + b \mid a \times b$$
Warmup

Arithmetic expressions over one variable $x$: meaning in $\mathbb{N} \rightarrow \mathbb{N}$

$$a, b ::= n \in \mathbb{N} \mid x \mid a + b \mid a \times b$$

This representation has redundancies: $2 + 2$ and $4$, same meaning.

TIMTOWTDI
Warmup

Arithmetic expressions over one variable $x$: meaning in $\mathbb{N} \rightarrow \mathbb{N}$

$$a, b ::= n \in \mathbb{N} \mid x \mid a + b \mid a \times b$$

This representation has redundancies: $2 + 2$ and $4$, same meaning.

TIMTOWTDI

Polynomials:

$$\sum_{0 \leq k \leq d} c_k x^k$$

More **canonical** representation: $2 + 2$ and $4$ both become $4x^0$.

TIOOWTDI
Warmup

Arithmetic expressions over one variable $x$: meaning in $\mathbb{N} \rightarrow \mathbb{N}$

$$a, b ::= n \in \mathbb{N} \mid x \mid a + b \mid a \times b$$

This representation has redundancies: $2 + 2$ and $4$, same meaning.

TIMTOWTDI

Polynomials:

$$\sum_{0 \leq k \leq d} c_k x^k$$

More canonical representation: $2 + 2$ and $4$ both become $4x^0$.

TIOOWTDI

Helps for application: does $a$ asymptotically dominate $b$?
Warmup

Arithmetic expressions over one variable $x$: meaning in $\mathbb{N} \to \mathbb{N}$

$$a, b ::= n \in \mathbb{N} \mid x \mid a + b \mid a \times b$$

This representation has redundancies: $2 + 2$ and $4$, same meaning.
TIMTOWTDI

Polynomials:

$$\sum_{0 \leq k \leq d} c_k x^k$$

More **canonical** representation: $2 + 2$ and $4$ both become $4x^0$.
TIOOWTDI

Helps for application: does $a$ asymptotically dominate $b$?
Less convenient to write: $P \times Q$. 
Representation and structure

Representations are human-designed.

Good representations reveal the *structure* of formal objects.

**Canonical** representations (no redundancies at all) precisely capture/expose this structure.
What about PL?

For programming languages, clear notion of equivalence given by contextual equivalence.

But representations are under-studied.

What is a canonical representation of the programs of your language?

Some applications:
  - Equivalence algorithms.
  - Program synthesis.
Logic

Logicians have studied proof representations for decades.

- Natural deduction
Logic

Logicians have studied *proof representations* for decades.

- Natural deduction
- Sequent calculus
Logic

Logicians have studied **proof representations** for decades.

- Natural deduction
- Sequent calculus
- Tableaux
Logic

Logicians have studied **proof representations** for decades.

- Natural deduction
- Sequent calculus
- Tableaux
- Matrices/connections
Logic

Logicians have studied **proof representations** for decades.

- Natural deduction
- Sequent calculus
- Tableaux
- Matrices/connections
- Expansion proofs
Logic

Logicians have studied **proof representations** for decades.

- Natural deduction
- Sequent calculus
- Tableaux
- Matrices/connections
- Expansion proofs
- Proof nets
Logic

Logicians have studied **proof representations** for decades.

- Natural deduction
- Sequent calculus
- Tableaux
- Matrices/connections
- Expansion proofs
- Proof nets
- Focusing
Logic

Logicians have studied **proof representations** for decades.

- Natural deduction
- Sequent calculus
- Tableaux
- Matrices/connections
- Expansion proofs
- Proof nets
- Focusing
- Multi-focusing

Eliminates redundancies, clarifies the structure of proof search, restricts the search space.
Logic

Logicians have studied **proof representations** for decades.

- Natural deduction
- Sequent calculus
- Tableaux
- Matrices/connections
- Expansion proofs
- Proof nets
- Focusing
- Multi-focusing
Logic

Logicians have studied *proof representations* for decades.

- Natural deduction
- Sequent calculus
- Tableaux
- Matrices/connections
- Expansion proofs
- Proof nets
- Focusing
- Multi-focusing

Eliminates redundancies, clarifies the structure of proof search, restricts the search space.
Contribution

A new Curry-Howard connection.

“The structure of programs corresponds to the structure of proof search.”
Contribution

A new Curry-Howard connection.

“The structure of programs corresponds to the structure of proof search.”

To find good program representations, go read logic papers.
Focusing

(Andreoli 1992)

Gives canonical representations for impure \(\lambda\)-calculi.

(Nice sequent syntax in Munch-Maccagnoni (2013).)
(Andreoli 1992)

Gives canonical representations for impure $\lambda$-calculi.
(Zeilberger 2009)

Nice sequent syntax in Munch-Maccagnoni (2013).
(Scherer and Rémy 2015)
Combines \textbf{backward} and \textbf{forward} proof-search.

Gives canonical representation of the \textbf{pure} simply-typed $\lambda$-calculus.

Application: equivalence of programs with sums and the empty type (Scherer 2017).
Program synthesis

Types with a unique inhabitant (Scherer and Rémy 2015): correct-by-construction synthesis.


Gabriel Scherer and Didier Rémy (2015). “Which simple types have a unique inhabitant?” *ICFP*.


11