Aleksy Schubert

Second-order unification and functional arity

Aleksy Schubert

Faculty of Mathematics, Informatics and Mechanics,
University of Warsaw

2nd of May, 2023

The context of the problem

- Programmers do not want to type a lot.

The context of the problem

- Programmers do not want to type a lot.
- Therefore need ways to automatically infer some coding information.

The context of the problem

- Programmers do not want to type a lot.
- Therefore need ways to automatically infer some coding information.
- In particular types are mostly optional.

The context of the problem

- Programmers do not want to type a lot.
- Therefore need ways to automatically infer some coding information.
- In particular types are mostly optional.
- Application of a function `M` to an argument `N`

$$\texttt{M N}$$

introduces a unification constraint.

The context of the problem

- Programmers do not want to type a lot.
- Therefore need ways to automatically infer some coding information.
- In particular types are mostly optional.
- Application of a function `M` to an argument `N`

$$\texttt{M N}$$

  introduces a unification constraint.
- In STLC:

$$X_M \doteq X_N \to X_{MN}$$

  where X_M, X_N, X_{MN} are unification variables.

In polymorphic systems

- Types are more complicated

$$A, B ::= C \mid X \mid A \to B \mid \forall X.A$$

In polymorphic systems

- Types are more complicated

$$A, B ::= C \mid X \mid A \to B \mid \forall X.A$$

- Applications of a function $\mathtt{M}$ to an argument $\mathtt{N}$ is more complicated

$$\mathtt{M}\ \mathtt{A}_1 \ldots \mathtt{A}_n \mathtt{N}$$

In polymorphic systems

- Types are more complicated

$$A, B ::= C \mid X \mid A \to B \mid \forall X.A$$

- Applications of a function $\mathtt{M}$ to an argument $\mathtt{N}$ is more complicated

$$\mathtt{M}\ \mathtt{A}_1 \ldots \mathtt{A}_n \mathtt{N}$$

- This introduces a unification constraint

$$F_{\mathtt{M}} \mathtt{A}_1 \ldots \mathtt{A}_n \doteq X_{\mathtt{N}} \to X_{\mathtt{M}\ \mathtt{A}_1 \ldots \mathtt{A}_n \mathtt{N}}$$

where $F_{\mathtt{M}}$ is a second-order unification variable, $X_{\mathtt{N}}, X_{\mathtt{M}\ \mathtt{A}_1 \ldots \mathtt{A}_n \mathtt{N}}$ are unification variables.

If we are (un)lucky...

- Constraints fall within the second-order unification language.

If we are (un)lucky...

- Constraints fall within the second-order unification language.
- If we are (un)lucky – second-order abstract syntax is necessary.

What kind of programming language?

$$M, N ::= x \mid \lambda x.M \mid \Lambda X.M \mid MA \mid MN$$

Type application critical for unification.
Type application omitted $\Rightarrow$ seminunification.

Additional restrictions

- Motivation for types:

Additional restrictions

- Motivation for types:
  - Expression of intent with regard to the program.

Additional restrictions

- Motivation for types:
  - Expression of intent with regard to the program.
  - Gentler writing of the code.

Additional restrictions

- Motivation for types:
  - Expression of intent with regard to the program.
  - Gentler writing of the code.
- Too big types are ineffective.

Additional restrictions

- Motivation for types:
  - Expression of intent with regard to the program.
  - Gentler writing of the code.
- Too big types are ineffective.
- Example (J.B.Wells):

$$
\begin{array}{l}
b : \forall\gamma.(\gamma \to \gamma) \to \beta, \\
c : \forall.(\mu_1 \to \delta_1) \to (\delta_2 \to \mu_2) \to (\tau_2 \to \tau_2), \\
\vdash \\
b(\lambda x.cxx)
\end{array}
$$

where $\tau_1 \leq \mu_1, \quad \tau_2 \leq \mu_2$ is an instance of the seminunification problem.

Additional restrictions

- All types in inference can be at most of size n.

Additional restrictions

- All types in inference can be at most of size n.
- Quantified variables restricted to occur only up to certain depth (Giannini, Ronchi Della Rocca).

Additional restrictions

- All types in inference can be at most of size n.
- Quantified variables restricted to occur only up to certain depth (Giannini, Ronchi Della Rocca).

- What if we bound the arity or functional rank?

Additional restrictions

- All types in inference can be at most of size n.
- Quantified variables restricted to occur only up to certain depth (Giannini, Ronchi Della Rocca).

- What if we bound the arity or functional rank?
  - $\text{arity}(c) = 0$ for a constant c and $\text{arity}(A_1 \to \cdots \to A_n \to c) = \max(\text{arity}(A_1), \ldots, \text{arity}(A_n), n)$,

Additional restrictions

- All types in inference can be at most of size n.
- Quantified variables restricted to occur only up to certain depth (Giannini, Ronchi Della Rocca).

- What if we bound the arity or functional rank?
  - $\mathrm{arity}(c) = 0$ for a constant c and $\mathrm{arity}(A_1 \to \cdots \to A_n \to c) = \max(\mathrm{arity}(A_1), \ldots, \mathrm{arity}(A_n), n)$,
  - $\mathrm{rank}(c) = 0$ for a constant c, and $\mathrm{rank}(A \to B) = \max(\mathrm{rank}(A) + 1, \mathrm{rank}(B))$.

Result

- Type-checking and type-inference in domain-free languages is undecidable when arity or rank are restricted.

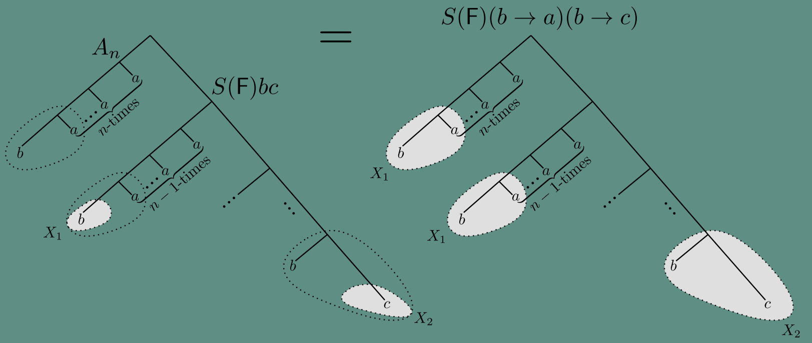
Result

- Type-checking and type-inference in domain-free languages is undecidable when arity or rank are restricted.

Example

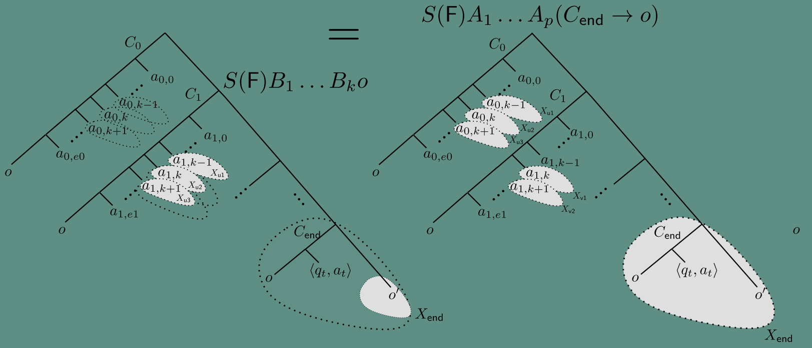
Consider SOU instance

$$A_n \to \mathsf{F}bc \quad \doteq \quad \mathsf{F}(b \to a)(b \to c)$$



(Misleading) Machine simulation

$$C_0 \to \mathsf{F} B_1 \dots B_k o \doteq \mathsf{F} A_1 \dots A_p (C_{n-1} \to o)$$



Verification of machine consistency

$$(\mathsf{F}\mathsf{s}1 \ldots \mathsf{s}p\, o') \to \mathsf{G}D'_1 \ldots D'_l \doteq \mathsf{G}D_1 \ldots D_l$$

$\mathsf{s} \to \langle b, o\rangle \rightsquigarrow \langle \mathsf{s}, 1\rangle$ in case $\pi_3(\mathsf{s}) = b$,
$\mathsf{s} \to o \rightsquigarrow \langle \mathsf{s}, 1\rangle$ in case $\pi_3(\mathsf{s}) = \bullet$, or $\pi_2(\mathsf{s}) = \bullet$,
$\mathsf{s} \to \langle \mathsf{s}', 1\rangle \rightsquigarrow \langle \mathsf{s}, 2\rangle$ in case $\pi_3(\mathsf{s}) = \pi_2(\mathsf{s}')$ and $\pi_1(\mathsf{s}') = \pi_2(\mathsf{s})$,
$\mathsf{s} \to \langle \mathsf{s}', 2\rangle \rightsquigarrow \langle \mathsf{s}, 3\rangle$ in case $\pi_3(\mathsf{s}) = \pi_2(\mathsf{s}')$ and $\pi_1(\mathsf{s}') = \pi_2(\mathsf{s})$,
$a \to \langle \mathsf{s}, 3\rangle \rightsquigarrow \langle a, 4\rangle$ in case $\pi_1(\mathsf{s}) = a$, $a \to \langle b, 4\rangle \rightsquigarrow \langle a, 4$

Conclusions

- Restriction to bounded types leads to decidable type-checking, type reconstruction.
- Restriction to types with bounded arity/rank may lead to undecidable type-checking, type reconstruction.

The End