Modest annotations with intersection types

Aleksy Schubert

Faculty of Mathematics, Informatics and Mechanics,
University of Warsaw

9th of July, 2024

The context of the problem

- Programmers do not want to write a lot.

The context of the problem

- Programmers do not want to write a lot.
- Strongly typed languages impair productivity [Hannenberg'210].

The context of the problem

- Programmers do not want to write a lot.
- Strongly typed languages impair productivity [Hannenberg’210].
- A study on Groovy [Souza and Figueiredo’14] showed that

The context of the problem

- Programmers do not want to write a lot.
- Strongly typed languages impair productivity [Hannenberg'210].
- A study on Groovy [Souza and Figueiredo'14] showed that
  - types in scripting scenarios are not used,

The context of the problem

- Programmers do not want to write a lot.
- Strongly typed languages impair productivity [Hannenberg’210].
- A study on Groovy [Souza and Figueiredo’14] showed that
  - types in scripting scenarios are not used,
  - types in structured programming are used.

The context of the problem

- Programmers do not want to write a lot.
- Strongly typed languages impair productivity [Hannenberg'210].
- A study on Groovy [Souza and Figueiredo'14] showed that
  - types in scripting scenarios are not used,
  - types in structured programming are used.
- Typing with sophisticated properties requires writing a lot. [later studies by Hannenberg's group]

The context of the problem

- Programmers do not want to write a lot.
- Strongly typed languages impair productivity [Hannenberg’210].
- A study on Groovy [Souza and Figueiredo’14] showed that
  - types in scripting scenarios are not used,
  - types in structured programming are used.
- Typing with sophisticated properties requires writing a lot. [later studies by Hannenberg’s group]

Question:

What typing disciplines support

- no types in scripts and
- sophisticated types in complex programs?

A proposed system syntax

- Traditional intersection types

$$\sigma, \tau ::= \alpha \mid \sigma \wedge \tau \mid \sigma \to \tau$$

A proposed system syntax

- Traditional intersection types

$$\sigma, \tau ::= \alpha \mid \sigma \wedge \tau \mid \sigma \to \tau$$

- Slightly non-standard λ-terms

$$M, N ::= x^{\sigma} \mid x \mid \lambda x\!:\!\sigma.M \mid \lambda x.M \mid MN$$

A proposed system syntax

- Traditional intersection types

$$\sigma, \tau ::= \alpha \mid \sigma \wedge \tau \mid \sigma \to \tau$$

- Slightly non-standard λ-terms

$$M, N ::= x^{\sigma} \mid x \mid \lambda x\!:\!\sigma.M \mid \lambda x.M \mid MN$$

- $x : \sigma$ is obligatory in λ when σ contains intersection.

A proposed system syntax

- Traditional intersection types

$$\sigma, \tau ::= \alpha \mid \sigma \wedge \tau \mid \sigma \to \tau$$

- Slightly non-standard λ-terms

$$M, N ::= x^{\sigma} \mid x \mid \lambda x\!:\!\sigma.M \mid \lambda x.M \mid MN$$

- $x : \sigma$ is obligatory in λ when σ contains intersection.
- Investigations on systematic type erasure in System F show that omitting types in λ leads to undecidability

A proposed system rules

$$\frac{}{\Gamma, x : \sigma \vdash x^{\sigma} : \sigma}\,(Var) \qquad \frac{\sigma \in T_{\rightarrow}}{\Gamma, x : \sigma \vdash x : \sigma}\,(VarS)$$

A proposed system rules

$$\frac{}{\Gamma, x:\sigma \vdash x^{\sigma} : \sigma}\,(Var) \qquad \frac{\sigma \in T_{\rightarrow}}{\Gamma, x:\sigma \vdash x : \sigma}\,(VarS)$$

$$\frac{\Gamma, x:\sigma \vdash M : \tau}{\Gamma \vdash \lambda x{:}\sigma.M : \sigma \rightarrow \tau}\,(\rightarrow I) \qquad \frac{\Gamma, x:\sigma \vdash M : \tau \quad \sigma \in T_{\rightarrow}}{\Gamma \vdash \lambda x.M : \sigma \rightarrow \tau}\,(\rightarrow IS)$$

$$\frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\,(\rightarrow E)$$

A proposed system rules

$$\frac{}{\Gamma, x : \sigma \vdash x^{\sigma} : \sigma}\,(Var) \qquad \frac{\sigma \in T_{\to}}{\Gamma, x : \sigma \vdash x : \sigma}\,(VarS)$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x{:}\sigma.M : \sigma \to \tau}\,(\to I) \qquad \frac{\Gamma, x : \sigma \vdash M : \tau \quad \sigma \in T_{\to}}{\Gamma \vdash \lambda x.M : \sigma \to \tau}\,(\to IS)$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\,(\to E)$$

$$\frac{\Gamma \vdash M : \sigma \wedge \tau}{\Gamma \vdash M : \sigma}\,(\wedge E1) \qquad \frac{\Gamma \vdash M : \sigma \wedge \tau}{\Gamma \vdash M : \tau}\,(\wedge E2)$$

Comments on the system design

- The system is based on the Church style ITD [Liquori and Ronchi Della Rocca'07].

Comments on the system design

- The system is based on the Church style ITD [Liquori and Ronchi Della Rocca'07].
- $(\wedge I)$ rule is not included [Kurata and Takahashi'95]

Comments on the system design

- The system is based on the Church style ITD [Liquori and Ronchi Della Rocca'07].
- $(\wedge I)$ rule is not included [Kurata and Takahashi'95]
  - Can placement of the rule be automatically and efficiently inferred?

Comments on the system design

- The system is based on the Church style ITD [Liquori and Ronchi Della Rocca'07].
- $(\wedge I)$ rule is not included [Kurata and Takahashi'95]
    - Can placement of the rule be automatically and efficiently inferred?
    - Automatic introduction may be complicated to understand.

Comments on the system design

- The system is based on the Church style ITD [Liquori and Ronchi Della Rocca'07].
- $(\wedge I)$ rule is not included [Kurata and Takahashi'95]
    - Can placement of the rule be automatically and efficiently inferred?
    - Automatic introduction may be complicated to understand.
    - It may trigger intersection types for variables without type annotations.

Decidability of the type reconstruction

- Generation of unification constraints.

Decidability of the type reconstruction

- Generation of unification constraints.
- Constraint use a special kind of second-order variables.

Decidability of the type reconstruction

- Generation of unification constraints.
- Constraint use a special kind of second-order variables.
- Reduction algorithm to solve constraints.

Decidability of the type reconstruction

- Generation of unification constraints.
- Constraint use a special kind of second-order variables.
- Reduction algorithm to solve constraints.
- One needs an additional ordering to solve the constraints.

Projection variables

A substitution S assigns to a second-order *projection* variable F expressions

$$A, B := \square \mid \blacksquare \mid A \wedge B$$

The result of application of the substitution to an expression Fσ is $A(\sigma)$ where

- $\square(\sigma) = \sigma$,
- $\blacksquare(\sigma)$ is undefined,
- $A_0 \wedge A_1(\sigma_0 \wedge \sigma_1) = A_i(\sigma_i)$ in case $A_{1-i}(\sigma_{1-1})$ is undefined for $i \in \{0, 1\}$,
- $A_0 \wedge A_1(\sigma_0 \wedge \sigma_1) = A_0(\sigma_0) \wedge A_1(\sigma_1)$ in case $A_i(\sigma_i)$ is defined for all $i \in \{0, 1\}$,

Constraint generation

Given Γ, M we define the function $\mathsf{Constr}(\Gamma; M)$ by induction on M as follows.

1. $\mathsf{Constr}(\Gamma; x^{\sigma}) = \{\Gamma(x) \doteq \sigma, \sigma \doteq X_x\}$,
2. $\mathsf{Constr}(\Gamma; x) = \{\Gamma(x) \doteq X_x^{\rightarrow}\}$,
3. $\mathsf{Constr}(\Gamma; MN; \vec{X}) =$
   $\texttt{let } E_M = \mathsf{Constr}(\Gamma; M) \texttt{ and } E_N = \mathsf{Constr}(\Gamma; N)$
   $\texttt{in } \{\mathsf{F}_M(X_M) \doteq \mathsf{F}_N(X_N) \to X_{MN}\} \cup E_M \cup E_N$,
4. $\mathsf{Constr}(\Gamma; \lambda x{:}\sigma.M) = \texttt{let } E_M = \mathsf{Constr}(\Gamma, x{:}\sigma; M)$
   $\texttt{in } \{X_{\lambda x:\sigma.M} \doteq \sigma \to X_M\} \cup E_M$,
5. $\mathsf{Constr}(\Gamma; \lambda x.M) = \texttt{let } E_M = \mathsf{Constr}(\Gamma, x : X_x^{\rightarrow}; M)$
   $\texttt{in } \{X_{\lambda x.M} \doteq X_x^{\rightarrow} \to X_M\} \cup E_M$.

Conclusions

- Explicit intersection types and implicit simple types lead to decidable type-checking, type reconstruction.
- What is the exact complexity?

The End