Local Type Inference for Polarised System F with Existentials

ANONYMOUS AUTHOR(S)

CHANGE!! This paper addresses the challenging problem of type inference for Impredicative System F with existential types, a critical aspect of many programming languages. While System F serves as the basis for type systems in numerous languages, existing type inference techniques for Impredicative System F are undecidable due to the presence of existential (∃) and polymorphic (∀) types. Consequently, current algorithms are often ad-hoc and sub-optimal. This paper presents novel contributions in the form of a local type inference algorithm for Impredicative System F with existential types. The algorithm introduces innovative techniques, such as a unique combination of unification and anti-unification, a full correctness proof, and the use of control structures inspired by Call-By-Push-Value . Additionally, the paper discusses a type inference framework that allows the algorithm to be applied to different type systems, offering insights into the under-researched area of impredicative existential type inference.

CCS Concepts: • **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

Additional Key Words and Phrases: Type Inference, System F, Call-by-Push-Value, Polarized Typing, Focalisation, Subtyping

ACM Reference Format:
Anonymous Author(s). 2018. Local Type Inference for Polarised System F with Existentials. *J. ACM* 37, 4, Article 111 (August 2018), 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

2 OVERVIEW

2.1 The Language

The types of $F^{\pm}\exists$ are given in fig. 1. They are stratified into two syntactic categories (polarities): positive and negative, similarly to the Call-By-Push-Value system [Levy 2006]. The negative types represent computations, and the positive types represent values:

– α^- is a negative type variable, which can be taken from a context or introduced by ∃.
– a function $P \to N$ takes a value as input and returns a computation;
– a polymorphic abstraction $\forall\overrightarrow{\alpha^+}.N$ quantifies a computation over a list of positive type variables $\overrightarrow{\alpha^+}$. The polarities are chosen to follow the definition of functions.
– a shift $\uparrow P$ allows a value to be used as a computation, which at the term-level corresponds to a pure computation **return** v.
+ α^+ is a positive type variable, taken from a context or introduced by ∀.
+ $\exists\overrightarrow{\alpha^-}.P$, symmetrically to ∀, binds negative variables in a positive type P.
+ a shift $\downarrow N$, symmetrically to the up-shift, thunks a computation, which at the term-level corresponds to $\{c\}$.

0004-5411/2018/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

Negative declarative types

$$N,\ M,\ K \quad ::= \quad \alpha^- \mid {\uparrow}P \mid P \to N \mid \forall\overrightarrow{\alpha^+}.N$$

Positive declarative types

$$P,\ Q,\ R \quad ::= \quad \alpha^+ \mid {\downarrow}N \mid \exists\overrightarrow{\alpha^-}.P$$

Fig. 1. Declarative Types of $F^{\pm}\exists$

Definitional Equalities. For simplicity, we assume alpha-equivalent terms equal. This way, we assume that substitutions do not capture bound variables. Besides, we equate $\forall\overrightarrow{\alpha^+}.\forall\overrightarrow{\beta^+}.N$ with $\forall\overrightarrow{\alpha^+}, \overrightarrow{\beta^+}.N$, as well as $\exists\overrightarrow{\alpha^-}.\exists\overrightarrow{\beta^-}.P$ with $\exists\overrightarrow{\alpha^-}, \overrightarrow{\beta^-}.P$, and lift these equations transitively and congruently to the whole system.

3 DECLARATIVE SYSTEM

4 ALGORITHM

5 PROOF

6 EXTENSIONS

7 CONCLUSION

[Botlan et al. 2003] [Dunfield et al. 2020]

REFERENCES

Didier Le Botlan and Didier Rémy (Aug. 2003). "MLF Raising ML to the Power of System F." In: *ICFP '03*. Uppsala, Sweden: ACM Press, pp. 52–63.

Jana Dunfield and Neel Krishnaswami (Nov. 2020). "Bidirectional Typing." In: arXiv: 1908.05839.

Paul Blain Levy (Dec. 2006). "Call-by-Push-Value: Decomposing Call-by-Value and Call-by-Name." In: *Higher-Order and Symbolic Computation* 19.4, pp. 377–414. DOI: 10.1007/s10990-006-0480-6.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009