Nominal Anti-Unification with Atom-Variables

FSCD 2022, Haifa, Israel

Manfred Schmidt-Schauß[1]

Daniele Nantes-Sobrinho[2]

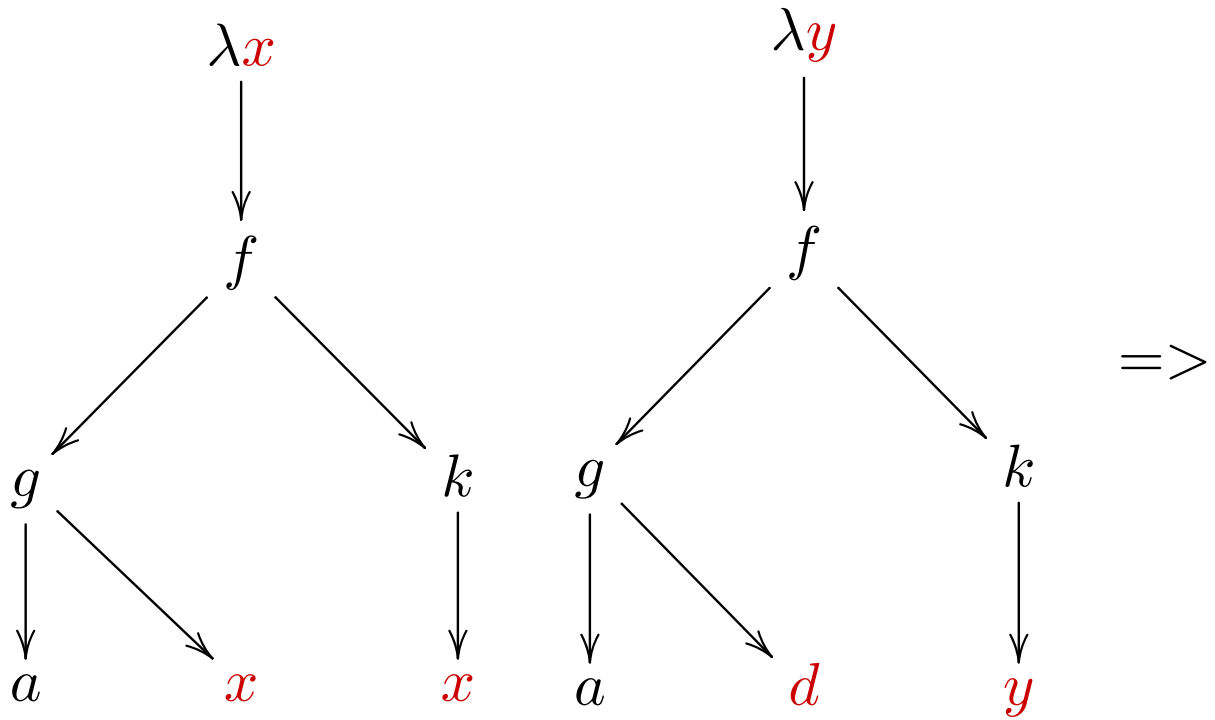
1: Goethe-University Frankfurt, Germany

2: Department of Computing, Imperial College London, London, UK

Departamento de Matemática, Universidade de Brasília, Brazil

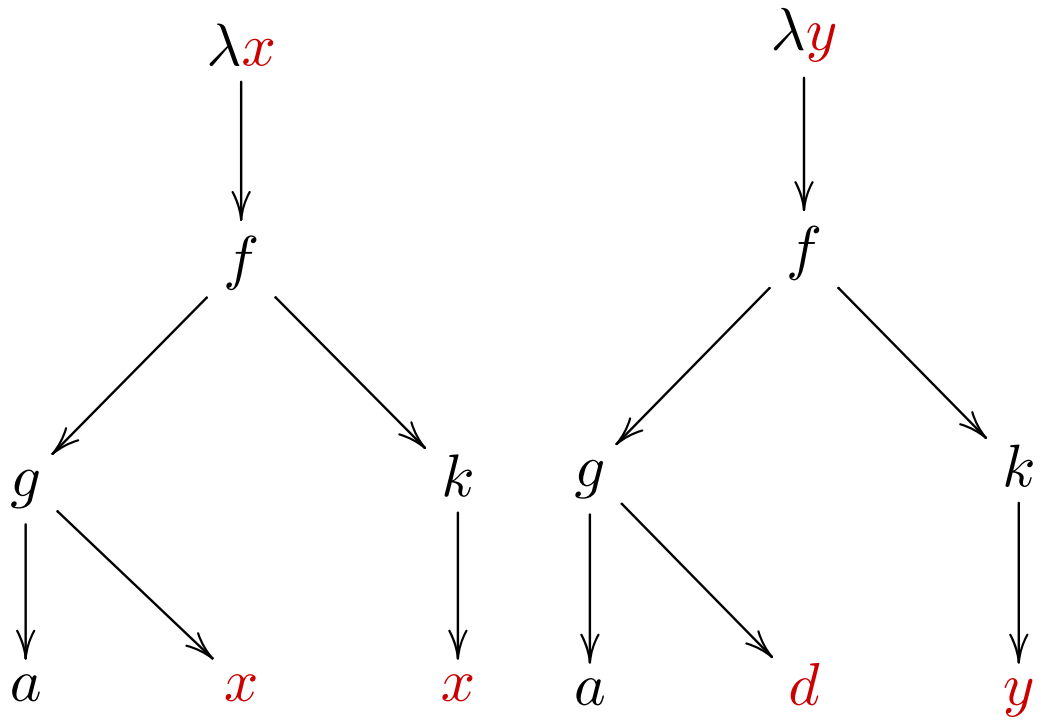
Examples for Term Generalization of lambda-expressions

Generalize the two terms:



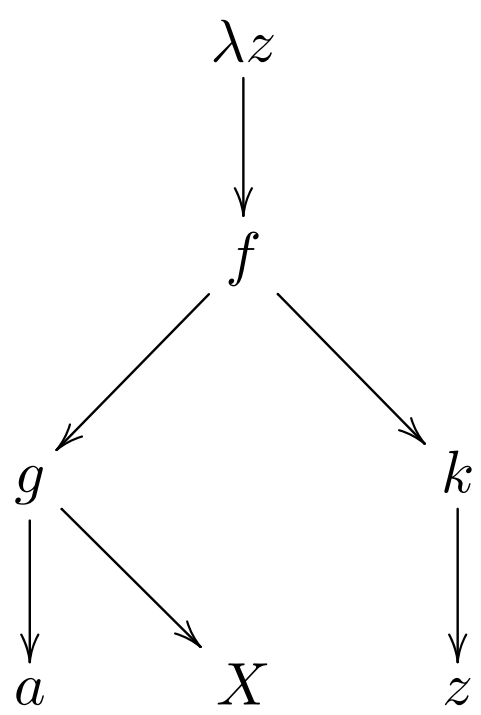
Examples for Term Generalization of lambda-expressions

Generalize the two terms:



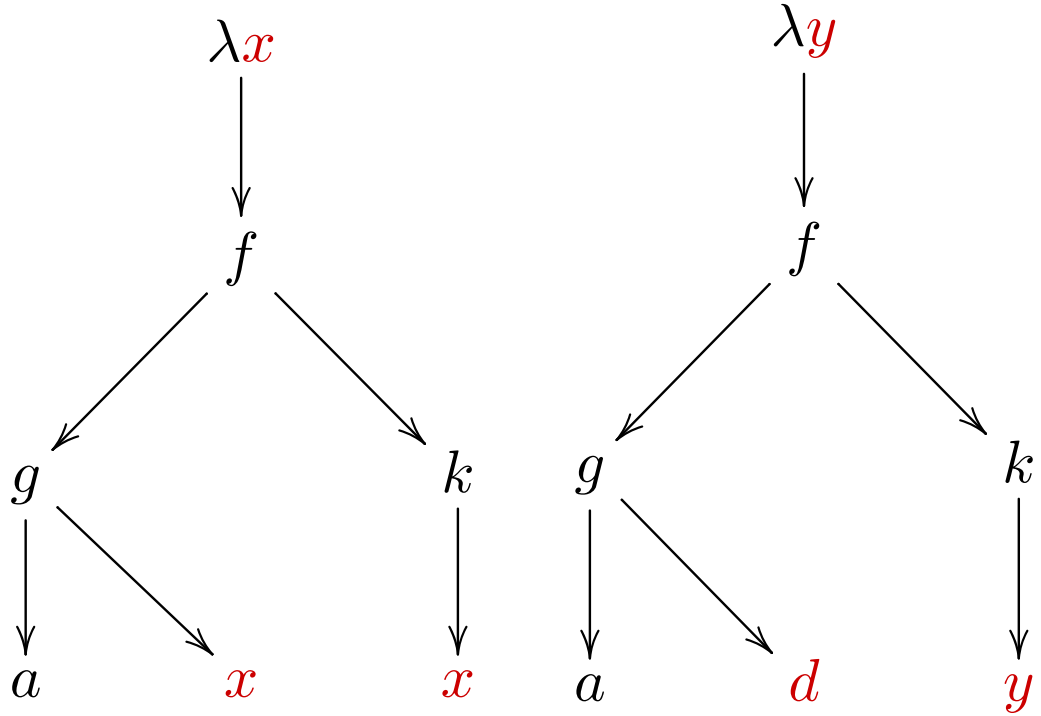
=>

Generalization:



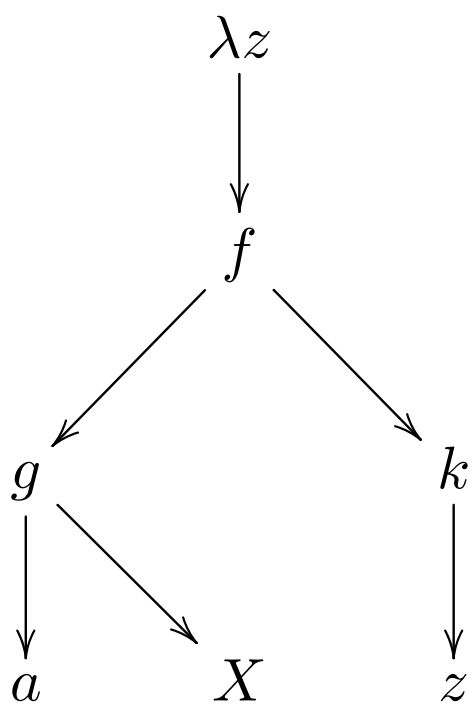
Examples for Term Generalization of lambda-expressions

Generalize the two terms:



=>

Generalization:



Task of Anti-Unification: Given a set of terms.
Compute a representation
of all least general generalizations (lgg-s)

Nominal Term Generalization

Our contribution:

- Anti-Unification,
  in an extended lambda-calculus NL_a modulo $\sim_\alpha$
  with grammar:

$$S ::= a \mid f(S_1, \ldots, S_n) \mid \lambda a.S$$

- Our general language for the input and algorithm is NL_A:

$$\begin{array}{lll} s & ::= & W \mid \pi{\cdot}X \mid f(s_1, \ldots, s_n) \mid \lambda W.s \\ W & ::= & \pi{\cdot}A \\ \pi & ::= & \emptyset \mid (W_1\ W_2){\circ}\pi \end{array}$$

A: atom-variables standing for atoms
π: permutations (of atoms)
W: suspensions

Some Related Work

Plotkin, A note on inductive generalization, MI, 1971

Urban,Pitts,Gabbay Nominal unification, CSL 2003

Baumgartner, Kutsia, Levy, Villaret. Nominal anti-unification, RTA 2015

S, Sabel, Kutz: Nominal unification with Atom-Variables, JSC 2019

Related Work

Baumgartner et.al.: Nominal anti-unification algorithm.

- Baumgartner et. al.'s algorithm: results in a single generalization. Striking Problem in Baumgartner et.al.: There are no *least* general generalizations!
- Since there are infinitely properly increasing chains of more and more general generalizations.
- The problem:
  It is always possible to refine constraints by adding components of the form $a\#X$, where a is an atom that does not occur in the problem.

Our apporach:

- Exploit atom-variables to avoid infinite increasing chains;
- And provide a more general formulation

NL_A. Terms-in-Context

A term-in context is a pair (t, C), where

t is a NL_A- term:

C is a NL_A freshness context, i.e. a set of constraints.
(of the form: $A\#s$.)

AtomAntiUnification: State: (L, M, Δ, θ)

L	Term-pairs to generalize with a default generalization-variable
M	unsolved term-pairs for later merge
Δ	The freshness context (i.e. constraints)
θ	Substitution; i.e. refinement of the generalization vars.

AtomAntiUnification: Rules of the algorithm

(Dec): Decomposition

$$\frac{\{X{:}f(s_1,\ldots,s_n) \triangleq f(t_1,\ldots,t_n)\} \uplus \Gamma, M, \nabla, \theta}{\Gamma \uplus \{X_1{:}s_1 \triangleq t_1, \ldots, X_n{:}s_n \triangleq t_n\}, M, \nabla, \theta \cup \{X \mapsto f(X_1,\ldots,X_n)\}}$$

where X_i are fresh variables

(Abs): Abstraction

$$\frac{\{X{:}\lambda W_1.s \triangleq \lambda W_2.t\} \uplus \Gamma, M, \nabla, \theta}{\begin{array}{l}\Gamma \uplus \{Y{:}(W_1\ B)\cdot s \triangleq (W_2\ B)\cdot t\}, M, \nabla \cup \{B\#\lambda W_1.s, B\#\lambda W_2.t\},\\ \theta \cup \{X \mapsto \lambda B.Y\}\end{array}}$$

where Y is a fresh variable, and B is a fresh atom-variable

(SusA): SuspensionA

$$\frac{\{X{:}W_1 \triangleq W_2\} \uplus \Gamma, M, \nabla, \theta \quad \nabla \vDash W_1 = W_2}{\Gamma, M, \nabla, \theta \cup \{X \mapsto W_1\}}$$

(SusYY): SuspensionYY

$$\frac{\{X{:}\pi_1\cdot Y \triangleq \pi_2\cdot Y\} \uplus \Gamma, M, \nabla, \theta \quad \nabla \vDash \pi_1 = \pi_2}{\Gamma, M, \nabla, \theta \cup \{X \mapsto \pi_1\cdot Y\}}$$

AtomAntiUnification: Further Rules of the algorithm

(Mer): Merging

$$\frac{\Gamma, \{Z_1{:}s_1 \triangleq t_1, Z_2{:}s_2 \triangleq t_2\} \uplus M, \nabla, \theta \quad \textsc{Eqvm}(\{(s_1, t_1) \preceq (s_2, t_2)\}, \nabla) = \pi}{\Gamma, M \cup \{Z_1{:}s_1 \triangleq t_1\}, \nabla, \theta \cup \{Z_2 \mapsto \pi{\cdot}Z_1\}}$$

where (Z_1, Z_2) is (X, Y) or (A, B)

(Solve)

$$\frac{\{X{:}s \triangleq t\} \uplus \Gamma, M, \nabla, \theta}{\Gamma, M \cup \{X{:}s \triangleq t\}, \nabla, \theta}$$

If $\mathit{Head}(s) \neq \mathit{Head}(t)$ and if s and t are not both suspensions of atom-variables.

(SolveYY)

$$\frac{\{X{:}\pi_1{\cdot}Y \triangleq \pi_2{\cdot}Y\} \uplus \Gamma, M, \nabla, \theta \quad \nabla \not\vdash \pi_1 = \pi_2}{\Gamma, M \cup \{X{:}\pi_1{\cdot}Y \triangleq \pi_2{\cdot}Y\}, \nabla, \theta}$$

(SolveAB)

$$\frac{\{X{:}W_1 \triangleq W_2\} \uplus \Gamma, M, \nabla, \theta \qquad \nabla \not\vdash W_1 = W_2}{\Gamma, M \cup \{A{:}W_1 \triangleq W_2\}, \nabla, \theta \cup \{X \mapsto A\}}$$

A is a fresh atom-variable.

AtomAntiUnification: Properties

Properties of AtomAntiUnification

- It computes a single generalization
- The algorithm requires simple exponential time to compute a solution.
  The number of rule applications is polynomial,
  and the solution requires polynomial space.

AtomAntiUnification problems

- requires exponential time:
  checking equivalence of permutations,
  and merging
- It is only weakly complete:
  It is complete if one ignores the freshness constraints.

An example as a hurdle to completeness of AtomAntiUnification

- Input: $(\emptyset, \{X : (f(A), A, B) \triangleq (c, A, B)\})$
- AtomAntiUnification computes the generalization $(\emptyset, (Y, A, B))$, however, it is not an lgg.

- Adding $A\#Y$ or $B\#Y$ violates the generalization property: $(f(a), a, a)$ not covered.
- $(\{B\#\lambda A.Y\}, (Y, A, B))$ is the lgg: $(\{B\#\lambda A.Y\}, (Y, A, B))$ allows the instance $(f(a), a, a)$, since $a\#\lambda a.f(a)$ holds.

How to get completeness?

Pros and Cons of using and refining freshness constraints

- (+) For a fixed input problem: The tree of properly refining freshness constraint C is finite.
- (-) However, we did not succeed in constructing a (terminating) algorithm that can construct maximal properly decreasing chains of constraints.

EQR-freshness contexts

We propose an extension of the expressive power of freshness constraints: **EQR-Constraints**

Permit more general freshness constraints based on **eq**uivalence **r**elations of atom-variables induced by semantic instantiations.

Example for EQR-freshness constraints:

The constraint $A\#\lambda B.A$
has an equivalent EQR-freshness context:

$$(\{A = B\} \implies \texttt{True}) \\ \wedge \\ (\{A \neq B\} \implies \texttt{False})$$

EQR-freshness contexts

- EQR-freshness contexts are strictly more general than freshness contexts (in NL_A)
- EQR-freshness contexts may be large. (exponentially larger than (usual) freshness contexts.
- There is an algorithm for refining EQR-freshness contexts using the same set of atom-variables
- The refinement steps are sufficiently fine-grained and terminate.

The Anti-Unification-Algorithm for EQR-Constraints

AtomAntiUnifLgg:

1. Run AtomAntiUnification.
2. Then: Iteratively refine the EQR-freshness constraints and check.

Theorem Algorithm AtomAntiUnifLgg terminates, and is complete and it computes a single lgg of given terms.
The complexity is exponential

Revisit Baumgartner, Kutsia, ...

Slightly changing their approach and algorithm:
Result: a unitary nominal anti-unification algorithm with usual freshness constraints.

idea: Apply the distinct-names-assumptions
(OR: do not take atom names too literally)

Conclusion and Further Work

- We constructed a nominal anti-unification algorithm
that is complete and outputs a single least general generalisation.
(for atom-variables, and with EQR-constraints)

- We constructed a unitary nominal anti-unification algorithm
improving the algorithm and result of Baumgartner, Kutsia, Levy, Villaret, RTA 2015. (Only atoms, no atom-variables)

Further Work

- Is AtomAntiUnification with refinement of freshness constraints finitary or unary?

- Extending the applicability of the methods to other higher-order languages. E.g. Haskell with recursive let constructs