Type Inference with structural Subtyping: the faithful description of an efficient constraint solver

Vincent Simonet
Talk Outline

- Structural subtyping
- Solving constraints
- Simplifying constraints
- Experimental results
Structural subtyping

Solving constraints
Simplifying constraints
Implementation

Structural subtyping
What is subtyping?

- A partial order $\leq$ on types and a subsumption rule:
  \[
  \frac{\Gamma \vdash e : t_1 \quad t_1 \leq t_2}{\Gamma \vdash e : t_2}
  \]

- Widely used: object-oriented languages, static analysis, ...

- The definition of the subtyping order itself varies, depending on the application.
Structural subtyping: an example

Ground types

\[ t ::= \text{num} \mid t \rightarrow t \mid t \times t \]
Structural subtyping: an example

**Atoms**

\[ a \in \text{real} \quad \leq_A \quad \text{complex} \quad \leq_A \quad \text{fermat} \quad \leq_A \quad \text{nat} \]

**Ground types**

\[ t ::= \text{num } a \mid t \rightarrow t \mid t \times t \]

**Subtyping order**

\[
\begin{align*}
& a \leq_A a' \\
& \text{num } a \leq \text{num } a' \\
& t_1' \leq t_1 \\
& t_2 \leq t_2' \\
& t_1 \rightarrow t_2 \leq t_1' \rightarrow t_2' \\
& t_1 \times t_2 \leq t_1' \times t_2'
\end{align*}
\]
Structural subtyping (1/2)

In structural subtyping, comparable types must have the same shape and can only differ by their atomic leaves.

- A poset of atoms $a \in (\mathcal{A}, \leq_{\mathcal{A}})$, supposed to be a lattice.

- A set of type constructors $c \in C$.
  Every type constructor is given with an arity, each parameter must be either covariant or contravariant.

- Ground types are defined by the following grammar:

$$t ::= a \mid c(t_1, \ldots, t_n)$$
Structural subtyping (2/2)

The subtyping order \( \leq \) is defined by lifting the atomic order \( \leq_A \) along the type structure:

\[
\frac{a \leq_A a'}{a \leq a'}
\]

if c's \( i^{th} \) parameter is covariant then \( t_i \leq t'_i \)

\[
\frac{c(t_1, \ldots, t_n) \leq c(t'_1, \ldots, t'_n)}{a \approx a'}
\]

\[
\forall i \ t_i \approx t'_i
\]

\( \approx \) is the symmetric transitive closure of \( \leq \).
Type inference for structural subtyping has been widely studied

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Year(s)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mitchell</td>
<td>1981–1984</td>
<td>First algorithms</td>
</tr>
<tr>
<td>Fuh and Mishra</td>
<td>1988-1989</td>
<td>Solving atomic constraints is PSPACE-hard in the general case, but linear in a lattice</td>
</tr>
<tr>
<td>Tiuryn</td>
<td>1992</td>
<td></td>
</tr>
<tr>
<td>Hoang and Mitchell</td>
<td>1995</td>
<td>Type inference is equivalent to constraint resolution</td>
</tr>
<tr>
<td>Rehof</td>
<td>1997</td>
<td>Minimal typings</td>
</tr>
<tr>
<td>Kuncak and Rinard</td>
<td>2003</td>
<td>The first-order theory of structural subtyping is decidable</td>
</tr>
</tbody>
</table>
Our work

However, most of the implementation techniques remains in folklore. This paper proposes:

- A complete account about efficient techniques for constraint resolution in the case of structural subtyping.
- A generic implementation of an efficient constraint solver for structural subtyping.
**Constraint-based type inference**

Type inference systems for subtyping are *constraint-based*:

- Every piece of code is described by a *constrained type scheme*, which is computed by a syntactic analysis:

\[
\sigma ::= \forall \bar{\alpha}[C].\tau \quad \text{(scheme)}
\]

\[
\tau ::= \alpha \mid a \mid c(\tau_1, \ldots, \tau_n) \quad \text{(type)}
\]

\[
C ::= \tau \leq \tau' \mid C_1 \land C_2 \mid \exists \alpha.C \quad \text{(constraint)}
\]

- It is *well-typed* if and only if the type scheme has an instance, i.e. its *constraint* is *satisfiable*

**type inference is reduced to constraint solving**
Example

let \( f = \lambda x . \lambda y ( \text{bind } s = x + y \text{ in} \)
\text{bind } p = x \times y \text{ in} \)
\((p + s, p - s) \))

\[
\forall \alpha \left[ \begin{array}{c}
\exists \alpha_1 \alpha_2 \alpha_3 . (\alpha_1 = \alpha_2 \rightarrow \alpha_3 \land \alpha_1 \leq \alpha) \\
\land \exists \alpha_4 \alpha_5 \alpha_6 . (\alpha_4 = \alpha_5 \rightarrow \alpha_6 \land \alpha_4 \leq \alpha_3) \\
\land \exists \beta_1 \gamma_1 . (\gamma_1 = \text{num } \beta_1 \land \alpha_2 \leq \gamma_1 \land \alpha_5 \leq \gamma_1) \\
\land \exists \beta_2 \gamma_2 . (\gamma_2 = \text{num } \beta_2 \land \alpha_2 \leq \gamma_2 \land \alpha_5 \leq \gamma_2) \\
\land \exists \beta_3 \gamma_3 . (\gamma_3 = \text{num } \beta_3 \land \gamma_1 \leq \gamma_3 \land \gamma_2 \leq \gamma_3) \\
\land \exists \beta_4 \gamma_4 . (\gamma_4 = \text{num } \beta_4 \land \gamma_1 \leq \gamma_4 \land \gamma_2 \leq \gamma_4) \\
\land \exists \alpha_7 \alpha_8 \gamma_5 . (\gamma_3 \leq \alpha_7 \land \gamma_4 \leq \alpha_8 \\
\land \gamma_5 = \alpha_7 \times \alpha_8 \land \gamma_5 \leq \alpha_6) \cdots \right] \cdot \alpha
\]
Example

\[\text{let } f = \lambda x.\lambda y (\text{bind } s = x + y \text{ in} \\
\quad (\text{bind } p = x \times y \text{ in} \\
\quad \quad (p + s, p - s)) )\]

\[\forall \alpha [\exists \beta. (\alpha = \text{num } \beta \rightarrow \text{num } \beta \rightarrow \text{num } \beta \times \text{num } \beta)].\alpha\]

\[\forall \beta. \text{num } \beta \rightarrow \text{num } \beta \rightarrow \text{num } \beta \times \text{num } \beta\]
Structural subtyping
► Solving constraints
  Simplifying constraints
  Implementation

Solving constraints
Overview

The solving procedure consists in three steps

- **Unification**: discovering sharing between type structures
- **Expansion and decomposition**: making type structure explicit and decomposing inequalities
- Solving the resulting **atomic problem**.
Solving constraints

Example: unification (1/2)

Two unification algorithms are simultaneously performed:

- One for type equality (=)
- One for type structure (≈)

\[
\begin{align*}
\alpha_1 = \alpha_2 & \rightarrow \alpha_3 \\
\alpha_1 \leq \alpha
\end{align*}
\]

\[
\frac{\langle \alpha_1 = \alpha_2 \rightarrow \alpha_3 \rangle \approx \alpha}{\alpha_1 \leq \alpha}
\]
Example: unification (2/2)

\[
\begin{align*}
\langle \alpha_1 = \alpha_2 \rightarrow \alpha_3 \rangle & \approx \alpha \\
\alpha_1 & \leq \alpha \\
\langle \alpha_4 = \alpha_5 \rightarrow \alpha_6 \rangle & \approx \alpha_3 \\
\alpha_4 & \leq \alpha_3 \\
\langle \gamma_5 = \alpha_7 \times \alpha_8 \rangle & \approx \alpha_6 \\
\gamma_5 & \leq \alpha_6 \\
\langle \gamma_1 = \text{num } \beta_1 \rangle & \approx \langle \gamma_2 = \text{num } \beta_2 \rangle \approx \langle \gamma_3 = \text{num } \beta_3 \rangle \approx \langle \gamma_4 = \text{num } \beta_4 \rangle \approx \\
\alpha_2 & \approx \alpha_5 \approx \alpha_7 \approx \alpha_8 \\
\alpha_2, \alpha_5 & \leq \gamma_1, \gamma_2 \land \gamma_1, \gamma_2 & \leq \gamma_3, \gamma_4 \land \gamma_3 & \leq \alpha_7 \land \gamma_4 & \leq \alpha_8 \\
\beta_1 & \approx \beta_2 \approx \beta_3 \approx \beta_4
\end{align*}
\]
Solving constraints

Example: expansion and decomposition

\[ \langle \alpha_1 = \alpha_2 \rightarrow \alpha_3 \rangle \approx \langle \alpha = \alpha_9 \rightarrow \alpha_{10} \rangle \]
\[ \alpha_1 \leq \alpha \]

\[ \langle \alpha_4 = \alpha_5 \rightarrow \alpha_6 \rangle \approx \alpha_3 \approx \alpha_{10} \]
\[ \alpha_3 \leq \alpha_{10} \land \alpha_4 \leq \alpha_3 \]

\[ \langle \gamma_5 = \alpha_7 \times \alpha_8 \rangle \approx \alpha_6 \]
\[ \gamma_5 \leq \alpha_6 \]

\[ \langle \gamma_1 = \text{num } \beta_1 \rangle \approx \langle \gamma_2 = \text{num } \beta_2 \rangle \approx \langle \gamma_3 = \text{num } \beta_3 \rangle \approx \langle \gamma_4 = \text{num } \beta_4 \rangle \approx \]
\[ \alpha_2 \approx \alpha_5 \approx \alpha_7 \approx \alpha_8 \approx \alpha_9 \]
\[ \alpha_9 \leq \alpha_2 \land \alpha_5 \leq \gamma_1, \gamma_2 \land \gamma_1, \gamma_2 \leq \gamma_3, \gamma_4 \land \gamma_3 \leq \alpha_7 \land \gamma_4 \leq \alpha_8 \]

\[ \beta_1 \approx \beta_2 \approx \beta_3 \approx \beta_4 \]
**Solving constraints**

Example: expansion and decomposition

\[ \langle \alpha_1 = \alpha_2 \rightarrow \alpha_3 \rangle \approx \langle \alpha = \alpha_9 \rightarrow \alpha_{10} \rangle \]

\[ \langle \alpha_4 = \alpha_5 \rightarrow \alpha_6 \rangle \approx \langle \alpha_3 = \alpha_{11} \rightarrow \alpha_{12} \rangle \approx \langle \alpha_{10} = \alpha_{13} \rightarrow \alpha_{14} \rangle \]

\[ \alpha_3 \leq \alpha_{10} \land \alpha_4 \leq \alpha_3 \]

\[ \langle \gamma_5 = \alpha_7 \times \alpha_8 \rangle \approx \langle \gamma_6 = \alpha_{12} \rangle \approx \langle \alpha_{14} \rangle \]

\[ \gamma_5 \leq \alpha_6 \land \alpha_{12} \leq \alpha_{14} \land \alpha_6 \leq \alpha_{12} \]

\[ \langle \gamma_1 = \text{num} \beta_1 \rangle \approx \langle \gamma_2 = \text{num} \beta_2 \rangle \approx \langle \gamma_3 = \text{num} \beta_3 \rangle \approx \langle \gamma_4 = \text{num} \beta_4 \rangle \approx \]

\[ \alpha_2 \leq \alpha_5 \approx \alpha_7 \approx \alpha_8 \approx \alpha_9 \approx \alpha_{11} \approx \alpha_{13} \]

\[ \alpha_9 \leq \alpha_2 \land \alpha_{13} \leq \alpha_{11} \land \alpha_{11} \leq \alpha_5 \land \alpha_2, \alpha_5 \leq \gamma_1, \gamma_2 \land \gamma_1, \gamma_2 \leq \gamma_3, \gamma_4 \land \gamma_3 \leq \alpha_7 \land \gamma_4 \leq \alpha_8 \]

\[ \beta_1 \approx \beta_2 \approx \beta_3 \approx \beta_4 \]
Complexity

This strategy is theoretically exponential, but has a linear complexity under the hypothesis of types of bounded height.

We have to simplify constraints throughout the solving process:

- To improve its efficiency,
- To obtain human readable results.
Simplifying constraints
Overview

Simplification is a subtle problem:

► It must be **correct**!
► It cannot be **time consuming**.

Simplification can be performed:

► **Throughout the expansion process**, in order to reduce the potential number of expanded variables.
► **At the atomic level**: useful in presence of let-polymorphism.

We are **not interested by optimality** (in size) of simplification techniques.
Polarities
[Fuh & Mishra (1989), Trifonov & Smith, Pottier]

Polarities assigned to type variables in a type scheme gives some information about their use:

A negative type variable represents an input of the expression.
A positive type variable represents an output of the expression.

Example:

\[ \forall \alpha_1 \alpha_2 \beta_1 \beta_2 [\alpha_1 \leq \beta_1 \land \alpha_2 \leq \beta_1 \land \alpha_2 \leq \beta_2]. \alpha_1 \rightarrow \alpha_2 \rightarrow \beta_1 \times \beta_2 \]

A negative type variable can receive new lower bounds.
A positive type variable can receive new upper bounds.
Chains reduction
[Eifrig et al. (1995), Aiken and Fähndrich (1996)]

Polarities can be exploited throughout expansion to eliminate variables:

A \text{non-positive} variable can be unified with its unique \text{upper bound}
A \text{non-negative} variable can be unified with its unique \text{lower bound}

Example:

\[ \forall \alpha_1\alpha_2\beta_1\beta_2[\alpha_1 \leq \beta_1 \land \alpha_2 \leq \beta_1 \land \alpha_2 \leq \beta_2].\alpha_1 \rightarrow \alpha_2 \rightarrow \beta_1 \times \beta_2 \]

is equivalent to

\[ \forall \alpha_1\alpha_2\beta_1\beta_2[\alpha_1 \leq \beta_1 \land \alpha_2 \leq \beta_1].\alpha_1 \rightarrow \alpha_2 \rightarrow \beta_1 \times \alpha_2 \]
Example: reducing a chain

\[
\alpha \approx \langle \alpha_1 = \alpha_2 \rightarrow \alpha_3 \rangle
\]
\[
\alpha_1 \leq \alpha
\]

\[
\alpha_3 \approx \langle \alpha_4 = \alpha_5 \rightarrow \alpha_6 \rangle
\]
\[
\alpha_4 \leq \alpha_3
\]

\[
\alpha_6 \approx \langle \gamma_5 = \alpha_7 \times \alpha_8 \rangle
\]
\[
\gamma_5 \leq \alpha_6
\]

\[
\langle \gamma_1 = \text{num } \beta_1 \rangle \approx \langle \gamma_2 = \text{num } \beta_2 \rangle \approx \langle \gamma_3 = \text{num } \beta_3 \rangle \approx \langle \gamma_4 = \text{num } \beta_4 \rangle \approx
\]
\[
\alpha_2 \approx \alpha_5 \approx \alpha_7 \approx \alpha_8
\]
\[
\alpha_2, \alpha_5 \leq \gamma_1 \wedge \alpha_2, \alpha_5 \leq \gamma_2 \wedge \gamma_1, \gamma_2 \leq \gamma_3, \gamma_4 \wedge \gamma_3 \leq \alpha_7 \wedge \gamma_4 \leq \alpha_8
\]

\[
\beta_1 \approx \beta_2 \approx \beta_3 \approx \beta_4
\]
Example: propagating polarities

\[ \langle \alpha = \alpha_1 = \alpha_2 \rightarrow \alpha_3 \rangle \]

\[ \langle \alpha_4 = \alpha_5 \rightarrow \alpha_6 \rangle \approx \alpha_3 \]
\[ \alpha_4 \leq \alpha_3 \]

\[ \langle \gamma_5 = \alpha_7 \times \alpha_8 \rangle \approx \alpha_6 \]
\[ \gamma_5 \leq \alpha_6 \]

\[ \langle \gamma_1 = \text{num} \beta_1 \rangle \approx \langle \gamma_2 = \text{num} \beta_2 \rangle \approx \langle \gamma_3 = \text{num} \beta_3 \rangle \approx \langle \gamma_4 = \text{num} \beta_4 \rangle \approx \alpha_2 \approx \alpha_5 \approx \alpha_7 \approx \alpha_8 \]
\[ \alpha_2, \alpha_5 \leq \gamma_1 \land \alpha_2, \alpha_5 \leq \gamma_2 \land \gamma_1, \gamma_2 \leq \gamma_3, \gamma_4 \land \gamma_3 \leq \alpha_7 \land \gamma_4 \leq \alpha_8 \]

\[ \beta_1 \approx \beta_2 \approx \beta_3 \approx \beta_4 \]
Example: removing a non-polar variable

\[ \langle \alpha = \alpha_4 = \alpha_2 \rightarrow \alpha_3 \rangle \]

\[ \alpha_3 \approx \langle \alpha_4 = \alpha_5 \rightarrow \alpha_6 \rangle \]
\[ \alpha_4 \leq \alpha_3 \]

\[ \alpha_6 \approx \langle \gamma_5 = \alpha_7 \times \alpha_8 \rangle \]
\[ \gamma_5 \leq \alpha_6 \]

\[ \langle \gamma_1 = \text{num} \beta_1 \rangle \approx \langle \gamma_2 = \text{num} \beta_2 \rangle \approx \langle \gamma_3 = \text{num} \beta_3 \rangle \approx \langle \gamma_4 = \text{num} \beta_4 \rangle \approx \]
\[ \alpha_2 \approx \alpha_5 \approx \alpha_7 \approx \alpha_8 \]

\[ \alpha_2, \alpha_5 \leq \gamma_1 \land \alpha_2, \alpha_5 \leq \gamma_2 \land \gamma_1, \gamma_2 \leq \gamma_3, \gamma_4 \land \gamma_3 \leq \alpha_7 \land \gamma_4 \leq \alpha_8 \]

\[ \beta_1 \approx \beta_2 \approx \beta_3 \approx \beta_4 \]
Example: reducing a chain

\[ \langle \alpha = \alpha_2 \rightarrow \alpha_3 \rangle \]

\[ \alpha_3 \approx \langle \alpha_4 = \alpha_5 \rightarrow \alpha_6 \rangle \]
\[ \alpha_4 \leq \alpha_3 \]

\[ \alpha_6 \approx \langle \gamma_5 = \alpha_7 \times \alpha_8 \rangle \]
\[ \gamma_5 \leq \alpha_6 \]

\[ \langle \gamma_1 = \text{num } \beta_1 \rangle \approx \langle \gamma_2 = \text{num } \beta_2 \rangle \approx \langle \gamma_3 = \text{num } \beta_3 \rangle \approx \langle \gamma_4 = \text{num } \beta_4 \rangle \approx \]
\[ \alpha_2 \approx \alpha_5 \approx \alpha_7 \approx \alpha_8 \]

\[ \alpha_2, \alpha_5 \leq \gamma_1 \wedge \alpha_2, \alpha_5 \leq \gamma_2 \wedge \gamma_1, \gamma_2 \leq \gamma_3, \gamma_4 \wedge \gamma_3 \leq \alpha_7 \wedge \gamma_4 \leq \alpha_8 \]

\[ \beta_1 \approx \beta_2 \approx \beta_3 \approx \beta_4 \]
Simplifying constraints

Example: reducing a chain

\[\langle \alpha = \alpha_2 \rightarrow \alpha_3 \rangle\]

\[\langle \alpha_3 = \alpha_5 \rightarrow \alpha_6 \rangle\]

\[\alpha_6 \approx \langle \gamma_5 = \alpha_7 \times \alpha_8 \rangle\]

\[\gamma_5 \leq \alpha_6\]

\[\langle \gamma_1 = \text{num } \beta_1 \rangle \approx \langle \gamma_2 = \text{num } \beta_2 \rangle \approx \langle \gamma_3 = \text{num } \beta_3 \rangle \approx \langle \gamma_4 = \text{num } \beta_4 \rangle \approx \alpha_2 \approx \alpha_5 \approx \alpha_7 \approx \alpha_8\]

\[\alpha_2, \alpha_5 \leq \gamma_1 \wedge \alpha_2, \alpha_5 \leq \gamma_2 \wedge \gamma_1, \gamma_2 \leq \gamma_3, \gamma_4 \wedge \gamma_3 \leq \alpha_7 \wedge \gamma_4 \leq \alpha_8\]

\[\beta_1 \approx \beta_2 \approx \beta_3 \approx \beta_4\]
Reducing two chains

\[ \langle \alpha = \alpha_2 \rightarrow \alpha_3 \rangle \]

\[ \langle \alpha_3 = \alpha_5 \rightarrow \alpha_6 \rangle \]

\[ \langle \alpha_6 = \alpha_7 \times \alpha_8 \rangle \]

\[ \langle \gamma_1 = \text{num } \beta_1 \rangle \approx \langle \gamma_2 = \text{num } \beta_2 \rangle \approx \langle \gamma_3 = \text{num } \beta_3 \rangle \approx \langle \gamma_4 = \text{num } \beta_4 \rangle \approx \]

\[ \alpha_2 \approx \alpha_5 \approx \alpha_7 \approx \alpha_8 \]

\[ \alpha_2, \alpha_5 \leq \gamma_1, \gamma_2 \land \gamma_1, \gamma_2 \leq \gamma_3, \gamma_4 \land \gamma_3 \leq \alpha_7 \land \gamma_4 \leq \alpha_8 \]

\[ \beta_1 \approx \beta_2 \approx \beta_3 \approx \beta_4 \]
Simplifying constraints

Example: reducing two chains

\[ \langle \alpha = \alpha_2 \rightarrow \alpha_3 \rangle \]

\[ \langle \alpha_3 = \alpha_5 \rightarrow \alpha_6 \rangle \]

\[ \langle \alpha_6 = \alpha_7 \times \alpha_8 \rangle \]

\[ \langle \gamma_1 = \text{num} \beta_1 \rangle \approx \langle \gamma_2 = \text{num} \beta_2 \rangle \approx \langle \gamma_3 = \alpha_7 \text{ num} \beta_3 \rangle \approx \langle \gamma_4 = \alpha_8 \text{ num} \beta_4 \rangle \approx \langle \alpha_2 \rangle \approx \langle \alpha_5 \rangle \]

\[ \alpha_2, \alpha_5 \leq \gamma_1, \gamma_2 \wedge \gamma_1, \gamma_2 \leq \gamma_3, \gamma_4 \]

\[ \beta_1 \approx \beta_2 \approx \beta_3 \approx \beta_4 \]
**Simplifying constraints**

**Example: expanding variables**

\[
\langle \alpha = \alpha_2 \rightarrow \alpha_3 \rangle
\]

\[
\langle \alpha_3 = \alpha_5 \rightarrow \alpha_6 \rangle
\]

\[
\langle \alpha_6 = \alpha_7 \times \alpha_8 \rangle
\]

\[
\langle \gamma_1 = \text{num } \beta_1 \rangle \approx \langle \gamma_2 = \text{num } \beta_2 \rangle \approx \langle \gamma_3 = \alpha_7 = \text{num } \beta_3 \rangle \approx \langle \gamma_4 = \alpha_8 = \text{num } \beta_4 \rangle \approx \langle \alpha_2 = \text{num } \beta_5 \rangle \approx \langle \alpha_5 = \text{num } \beta_6 \rangle
\]

\[
\alpha_2, \alpha_5 \leq \gamma_1, \gamma_2 \land \gamma_1, \gamma_2 \leq \gamma_3, \gamma_4
\]

\[
\beta_1 \approx \beta_2 \approx \beta_3 \approx \beta_4 \approx \beta_5 \approx \beta_6
\]
Simplifying constraints

Example: decomposing inequalities

\[
\langle \alpha = \alpha_2 \rightarrow \alpha_3 \rangle
\]

\[
\langle \alpha_3 = \alpha_5 \rightarrow \alpha_6 \rangle
\]

\[
\langle \alpha_6 = \alpha_7 \times \alpha_8 \rangle
\]

\[
\langle \gamma_1 \geq \text{num} \beta_1 \rangle \approx \langle \gamma_2 \geq \text{num} \beta_2 \rangle \approx \langle \gamma_3 = \alpha_7 = \text{num} \beta_3 \rangle \approx \langle \gamma_4 = \alpha_8 = \text{num} \beta_4 \rangle \approx \langle \alpha_2 = \text{num} \beta_5 \rangle \approx \langle \alpha_5 = \text{num} \beta_6 \rangle
\]

\[
\alpha_2, \alpha_5 \leq \gamma_1, \gamma_2 \land \gamma_1, \gamma_2 \leq \gamma_3, \gamma_4
\]

\[
\beta_1 \approx \beta_2 \approx \beta_3 \approx \beta_4 \approx \beta_5 \approx \beta_6
\]

\[
\beta_5, \beta_6 \leq \gamma_1, \gamma_2 \land \gamma_1, \gamma_2 \leq \beta_3, \beta_4
\]
Summary of expansion

- On this example, expansion introduces only 2 variables.
- In the absence of simplification, they would have been 25.
- It remains to simplify the atomic graph.
Example: polarised closure
[Fuh & Mishra (1989)]

Polarised closure removes non-polar variables in the atomic graph, and keeps only paths from negative variables to positive ones.
**Simplifying constraints**

**Example: minimisation**


Two negative variables with the same successors are equivalent to two positive variables with the same predecessors.

\[
\begin{align*}
\beta_6 & \quad \beta_4 \\
\beta_5 & \quad \beta_3 \\
\beta_6 = \beta_5 & \quad \beta_4 = \beta_3 \\
\beta_6 = \beta_4 & \quad \beta_5 \neq \beta_3
\end{align*}
\]
Example: final result

\[ \langle \alpha = \alpha_2 \rightarrow \alpha_3 \rangle \]

\[ \langle \alpha_3 = \alpha_5 \rightarrow \alpha_6 \rangle \]

\[ \forall \alpha. [\exists \cdots]. \alpha \]

\[ \forall \alpha[\exists \beta_5.(\alpha = \text{num } \beta_5 \rightarrow \text{num } \beta_5 \rightarrow \text{num } \beta_5 \times \text{num } \beta_5)]. \alpha \]

\[ \forall \beta_5. \text{num } \beta_5 \rightarrow \text{num } \beta_5 \rightarrow \text{num } \beta_5 \times \text{num } \beta_5 \]

\[ \langle \alpha_2 = \alpha_5 = \alpha_7 = \alpha_8 = \text{num } \beta_5 \rangle \]

\[ \langle \beta_5 = \beta_6 = \beta_3 = \beta_4 \rangle \]
Implementation

- Structural subtyping
- Solving constraints
- Simplifying constraints

Implementation
The Dalton Library

The Dalton Library is a real-size implementation in Objective Caml of these algorithms.

http://cristal.inria.fr/~simonet/soft/dalton/

It comes as a functor parametrized by a series of modules describing the client’s type system.

It has been used within the Flow Caml System, an information flow analyzer for the Caml Language

http://cristal.inria.fr/~simonet/soft/flowcaml/
### Experimental results

<table>
<thead>
<tr>
<th></th>
<th>Typing the code of Caml Light library</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.s.t. nodes</td>
<td>14002</td>
<td>22996</td>
</tr>
<tr>
<td>Unification solver</td>
<td>0.346 s</td>
<td>0.954 s</td>
</tr>
<tr>
<td>Structural subtyping solver</td>
<td>0.966 s</td>
<td>2.213 s</td>
</tr>
<tr>
<td>/</td>
<td>2.79</td>
<td>2.31</td>
</tr>
</tbody>
</table>
Conclusion

- An efficient implementation type inference engine for structural subtyping
- whose correctness is proved.