# Problems and Results in Unification and Type Search

Joe Wells
University of Glasgow
Computer Science Department
jbw@dcs.gla.ac.uk
http://www.dcs.gla.ac.uk/~jbw

June 30, 1998

# Area of Talk

- Type Search:
  - Involves finding a complete typing derivation for a term with incomplete type information.
  - Depends on:
    * The typed calculus, e.g., simply typed $\lambda$-calculus, System F, $F_\omega$.
    * The amount of type information, e.g., none, the final type and the types of free variables, or the types of all variables.

- Unification is used for type search problems:
  - performing type search
  - proving undecidability

  This talk focuses on the latter.

# Overview

- Type search and System F. (The variants F+$\eta$, F$_\omega$, and Church-style F are presented later as they are used.)
- Semi-unification (SUP) and 2nd-order unification (2UP).
- Reductions proving undecidability (hints, not proofs):
  - SUP to type checking in Curry-style F and F$_\omega$.
  - 2UP to typability in Church-style F.
  - 2UP to type checking in F$_\omega$.
  - SUP to subtyping in F+$\eta$.
- Open problems.

# Type Search

Type systems are generally rule sets for deriving *judgements* $A \vdash M : \tau$, meaning "$M$ has type $\tau$ using type assumptions $A$".

Type search problems include:

- Typability: $\_ \vdash M : \_$ ?
- Type checking: $A \vdash M : \tau$ ?

Warning: Many call finding types for untyped programs *type checking*.

- Partial type inference: $\_ \vdash C[\_, \ldots, \_] : \_$ ? (Explicitly typed terms.)

Issues: Can the blanks be filled to make the judgements derivable? If yes, what types can be put in the blanks?

# Type Search: Relevance

The primary driving applications for type search are programming languages. The slogan is: "Well typed programs do not go wrong".

Of the type search problems listed, it is most desired that typability is decidable, because this would allow human programmers to write completely type-free programs and rely on the compiler to verify program safety.

Partial type inference has been considered primarily because typability has been proven to be decidable only for relatively restricted systems, e.g., the Hindley/Milner system.

Type checking is interesting partially because sometimes full context information is available and partially because its decidability would imply that there is no need to store full typing derivations — they can be represented by their final judgements.

# System F (Curry Style)

Types (really type expressions): $\mathbb{T} ::= \mathbb{V} \mid \mathbb{T} \to \mathbb{T} \mid \forall\, \mathbb{V}.\, \mathbb{T}$

Inference rules (equivalent to standard):

VAR $A \vdash x : \sigma$ if $A(x) = \sigma$

APP $\dfrac{A \vdash M : \sigma \to \tau; \qquad A \vdash N : \sigma}{A \vdash (M\ N) : \tau}$ ABS $\dfrac{A \cup \{x{:}\sigma\} \vdash M : \tau}{A \vdash (\lambda x.M) : \sigma \to \tau}$

INST/GEN $\dfrac{A \vdash M : \forall \vec{\alpha}.\sigma}{A \vdash M : \forall \vec{\beta}.S(\sigma)}$ if $(\{\vec{\beta}\} \cup \mathrm{DOM}(S)) \cap \mathrm{FTV}(A) = \varnothing$

It can be required that INST/GEN is used exactly once after every other rule. (Makes rules syntax-directed.)

# Variants of System F

Later, the following variants of System F will be presented as they are discussed:

- $\mathsf{F}_\omega$ (Curry-style): Differs from F in its set of types and its notion of type equivalence.
- F+$\eta$: Differs from F by the addition of the subject $\eta$-reduction rule or, equivalently, by the addition of a particular subtyping relation.
- Church-style System F: Differs from F by making type abstraction and application syntactically explicit.

# Unification: Terms

A language $\mathcal{L}$ is a pair $\langle \Sigma, X \rangle$ where $\Sigma = \{f_1^{(i_1)}, f_2^{(i_2)}, \ldots\}$ and $X = \{x_1^{(j_1)}, x_2^{(j_2)}, \ldots\}$. The optional superscript in $f^{(i)}$ and $x^{(i)}$ denotes arity.

Let $X^0 = \{\, x \in X \mid \mathrm{Arity}(x) = 0 \,\}$. The language is *first-order* if $X = X^0$ and is otherwise *second-order*.

Terms: $\mathcal{T}(\langle \Sigma, X \rangle)$ is least set such that if $Z^{(n)} \in \Sigma \cup X$ and $t_1, \ldots, t_n \in \mathcal{T}(\langle \Sigma, X \rangle)$, then $Z(t_1, \ldots, t_n) \in \mathcal{T}(\langle \Sigma, X \rangle)$.

A term $t$ is *ground* iff $\mathrm{Vars}(t) = \varnothing$ and is *simple* iff for $x(u_1, \ldots, u_n)$ occuring in $t$, each $u_i$ is ground.

# Unification: Substitutions

P-terms: $\mathcal{T}_{\mathrm{p}}(\langle\Sigma, X\rangle) = \mathrm{Seqs}(X^0) \times \mathcal{T}(\langle\Sigma, X\rangle)$.
Let $\mathrm{Arity}(\langle x_1 \ldots x_n, t\rangle) = n$.

Substitutions over $\langle\Sigma, X\rangle$: $S : X \rightharpoonup \mathcal{T}_{\mathrm{p}}(\langle\Sigma, X\rangle)$ where $\mathrm{Arity}(x) = \mathrm{Arity}(S(x))$ for $x \in \mathrm{DOM}(S)$.

A substitution $S$ over $\mathcal{L}$ is extended (usually implicitly) to $\overline{S} : \mathcal{L} \to \mathcal{L}$:

$$\overline{S}(Z(t_1, \ldots, t_n)) = \begin{cases} Z(\overline{S}(t_1), \ldots, \overline{S}(t_n)) & \text{if } Z \notin \mathrm{DOM}(S) \\ \overline{S'}(t) & \text{if } S(Z) = \langle x_1 \ldots x_n, t\rangle \\ & \text{and } S' = \{\, x_i \mapsto \overline{S}(t_i) \mid 1 \le i \le n \,\} \end{cases}$$

$S$ is *first-order* if $\mathrm{DOM}(S) \subseteq X^0$ and is otherwise *second-order*.

# The Language

Generally, the results discussed will use a language $\mathcal{L}^{\rightarrow} = \langle \Sigma, X \rangle$ with one binary function symbol $\rightarrow$, written in infix notation. This will correspond to the use of $\rightarrow$ as a type constructor.

There may be any number of nullary function symbols (constants). One of the results depends on $|\Sigma|$ being large; the others do not.

# Unification Problems

Let $t \leq u$ ("$t$ *matches* $u$") iff $\exists S.\, S(t) = u$ and $\mathrm{Order}(S) = 1$.

An *instance* is a set $E = \{\langle t_1, u_1\rangle, \ldots, \langle t_n, u_n\rangle\}$.

The semi-unification problem (SUP): Given $E$ as above, does there exist 1st-order $S$ such that $S(t_i) \leq S(u_i)$ for $1 \leq i \leq n$?

The $k$th-order unification problem ($k$UP): Given $E$ as above, does there exist $S$ such that $\mathrm{Order}(S) \leq k$ and $S(t_i) = S(u_i)$ for $1 \leq i \leq n$?

The *simple* $k$th-order unification problem (simple-$k$UP): As $k$UP except restricted to instances with only simple terms.

# Unification Problems: Status

Let $P \leq Q$ abbreviate "problem $P$ is Turing-reducible to problem $Q$". Let XUP$(n)$ stand for XUP restricted to instances with at most $n$ pairs and a language with one binary function symbol and no other non-nullary symbols.

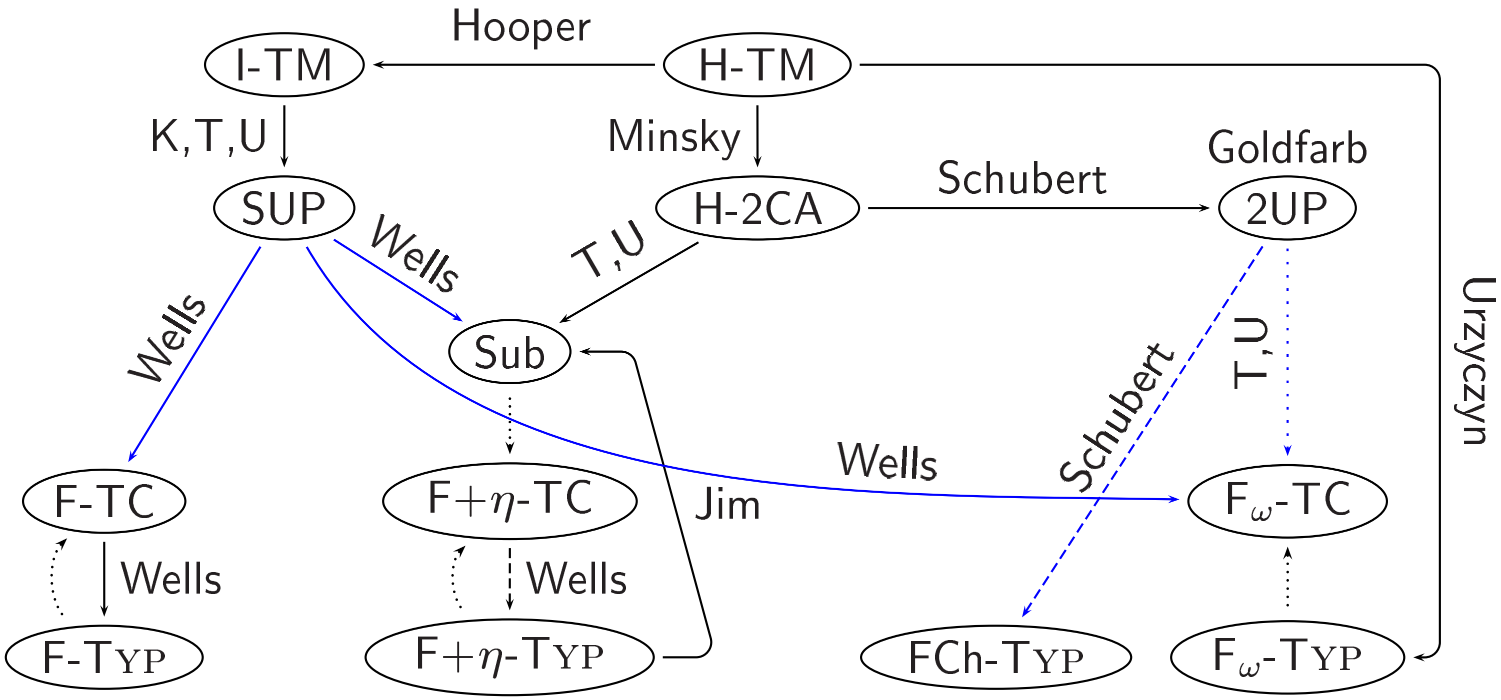
- 1UP $\leq$ SUP and 1UP $\leq$ 2UP. (Trivially.)
- 2UP is undecidable [Goldfarb, 1981], also simple-2UP [Schubert, 1998].
- SUP is undecidable [Kfoury et al., 1993].
- 2UP $\leq$ 2UP$(1)$, simple-2UP $\leq$ simple-2UP$(1)$, and SUP $\leq$ SUP$(2)$, but SUP $\not\leq$ SUP$(1)$.

# Problem Reductions Showing Undecidability

The following problem reductions showing undecidability will now be shown:

- SUP to type checking in Curry-style F and $F_\omega$.
- 2UP to typability in Church-style F.
- 2UP to type checking in $F_\omega$.
- SUP to subtyping in F+$\eta$.

# The Web of Reductions




| | | |
|---|---|---|
| partial, enough for undecidability | folk result | trivial, well known |

# SUP $\leq$ Type Checking for System F

Let $E = \{\langle\sigma_1, \tau_1\rangle, \langle\sigma_2, \tau_2\rangle\}$ be an instance in $\mathcal{L}^{\rightarrow}$.

Let $\{\vec{\alpha}\} = \mathrm{Vars}(E)$ and choose fresh variables $\beta_1$ and $\beta_2$.

Let $M = x(\lambda z.yzz)$.

Define some types:

$$\begin{array}{lll} \rho_x = \forall\alpha.((\alpha \to \alpha) \to \forall\alpha.\alpha) & & \\ \pi_1^0 = \tau_1 \to \beta_1 & \pi_2^0 = \beta_2 \to \tau_2 & \pi_3^0 = \sigma_1 \to \sigma_2 \\ \rho_y = \forall\vec{\alpha}.\forall\vec{\beta}.(\pi_1^0 \to \pi_2^0 \to \pi_3^0) & & \end{array}$$

Let $A = \{x{:}\rho_x, y{:}\rho_y\}$.

**Theorem 1. [Wells [1994]]** *$E$ is semi-unifiable iff $A \vdash M : \forall\alpha.\alpha$ is valid in System F.*

# Proof: Part 1

$A \vdash M : \forall\alpha.\alpha$ is derivable only by a derivation of this shape:

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{B \vdash y : \forall\vec{\alpha}.\forall\vec{\beta}.(\pi_1^0 \to \pi_2^0 \to \pi_3^0)}{B \vdash y : \pi_1^1 \to \pi_2^1 \to \pi_3^1}\,U^1
\qquad
\dfrac{B \vdash z : \forall\vec{\gamma}.\pi_3^3}{B \vdash z : \pi_1^1}\,T_1
}{B \vdash yz : \pi_2^1 \to \pi_3^1}
}{B \vdash yz : \pi_2^2 \to \pi_3^2}\,U^2
\qquad
\dfrac{B \vdash z : \forall\vec{\gamma}.\pi_3^3}{B \vdash z : \pi_2^2}\,T_2
}{B \vdash yzz : \pi_3^2}
}{B \vdash yzz : \forall\vec{\gamma}.\pi_3^3}\,U^3
}{A \vdash (\lambda z.yzz) : (\forall\vec{\gamma}.\pi_3^3) \to (\forall\vec{\gamma}.\pi_3^3)}
}{A \vdash (\lambda z.yzz) : (\forall\vec{\gamma}.\pi_3^4) \to (\forall\vec{\gamma}.\pi_3^4)}
$$

$$
\dfrac{
\dfrac{A \vdash x : \forall\alpha.((\alpha \to \alpha) \to \forall\alpha.\alpha)}{A \vdash x : ((\forall\vec{\gamma}.\pi_3^4) \to (\forall\vec{\gamma}.\pi_3^4)) \to \forall\alpha.\alpha}
\qquad
A \vdash (\lambda z.yzz) : (\forall\vec{\gamma}.\pi_3^4) \to (\forall\vec{\gamma}.\pi_3^4)
}{A \vdash M : \forall\alpha.\alpha}
$$

# Proof: Part 2

Thus, $A \vdash M : \forall\alpha.\alpha$ is derivable iff there is a solution to:

$$\begin{aligned} U^2(T_1(U^3(U^2(U^1(\sigma_1))))) &= U^2(U^1(\tau_1)) \\ T_2(U^3(U^2(U^1(\sigma_2)))) &= U^2(U^1(\tau_2)) \end{aligned}$$

Note: A solution is in *substitutions of System F*, which can have $\forall$-bindings in their range!

Because substitutions of F are closed under function composition, the above has a solution iff the following does:

$$\begin{aligned} R_1(R_0(\sigma_1)) &= R_0(\tau_1) \\ R_2(R_0(\sigma_2)) &= R_0(\tau_2) \end{aligned}$$

# Proof: Part 3

It has been shown that $A \vdash M : \forall\alpha.\alpha$ is derivable in System F iff there is a solution in substitutions of System F to:

$$R_1(R_0(\sigma_1)) = R_0(\tau_1) \qquad R_2(R_0(\sigma_2)) = R_0(\tau_2) \tag{1}$$

Obviously, if there is a solution to (1) in unification substitutions, there is one in F-substitutions. Suppose $R_0$, $R_1$, and $R_2$ are F-substitutions solving (1). Choose a fresh variable $\beta$ and define:

$$\mathrm{er}\,(\alpha) = \alpha \qquad \mathrm{er}\,(\sigma \to \tau) = \mathrm{er}\,(\sigma) \to \mathrm{er}\,(\tau) \qquad \mathrm{er}\,(\forall\alpha.\sigma) = \mathrm{er}\,(\sigma[\alpha{:=}\beta])$$

Then $S_0 \circ \mathrm{er}$, $S_1 \circ \mathrm{er}$, and $S_2 \circ \mathrm{er}$ solve (1) in unification substitutions.

# $\mathsf{F}_\omega$ (Curry-Style)

Kinds: $\mathbb{K} ::= * \mid \mathbb{K} \to \mathbb{K}$ Constructor variables: $\mathbb{V} = \bigcup_{K \in \mathbb{K}} \mathbb{V}_K$.

Constructors: For each $K$, the *constructors* $\mathbb{C}_K$ of kind $K$ is least set such that

$$\begin{array}{llll}
\gamma \in \mathbb{V}_K & & & \Rightarrow \gamma \in \mathbb{C}_K \\
\varphi_1 \in \mathbb{C}_{K_1 \to K_2} & \wedge & \varphi_2 \in \mathbb{C}_{K_1} & \Rightarrow (\varphi_1 \varphi_2) \in \mathbb{C}_{K_2} \\
\gamma \in \mathbb{V}_{K_1} & \wedge & \varphi \in \mathbb{C}_{K_2} & \Rightarrow (\lambda \gamma . \varphi) \in \mathbb{C}_{K_1 \to K_2} \\
\gamma \in \mathbb{V}_K & \wedge & \varphi \in \mathbb{C}_* & \Rightarrow (\forall \gamma . \varphi) \in \mathbb{C}_* \\
\varphi_1 \in \mathbb{C}_* & \wedge & \varphi_2 \in \mathbb{C}_* & \Rightarrow (\varphi_1 \to \varphi_2) \in \mathbb{C}_*
\end{array}$$

$\alpha\beta$-equivalent constructors of same kind are identified.

Types: $\mathbb{T} = \mathbb{C}_*$.

Inference rules are same as for System F.

# SUP $\leq$ Type Checking for F$_\omega$

Let $E = \{\langle\sigma_1, \tau_1\rangle, \langle\sigma_2, \tau_2\rangle\}$, $A$, and $M$ be as in the theorem for System F.

**Theorem 2. [Wells [1997]]** *$E$ is semi-unifiable iff $A \vdash M : \forall\alpha.\alpha$ is derivable in F$_\omega$.*

The proof mostly proceeds the same as before, but there are some difficulties.

# Proof: Part 1

Identically to before, it can be shown that $A \vdash M : \forall\alpha.\alpha$ is derivable in $\mathsf{F}_\omega$ iff there is a solution in substitutions of $\mathsf{F}_\omega$ to:

$$R_1(R_0(\sigma_1)) = R_0(\tau_1) \qquad R_2(R_0(\sigma_2)) = R_0(\tau_2)$$

Extracting a semi-unifier from the $\mathsf{F}_\omega$-substitutions has two problems:

1. The range of the $\mathsf{F}_\omega$-substitutions may contain quantification, constructor abstractions, and constructor applications.

2. The two sides of each equation may be $\alpha\beta$-equivalent rather than identical.

# Proof: Part 2

Let $[\varphi]$ denote the $\alpha\beta$-equivalence class of constructor $\varphi$.

For each equivalence class $[\varphi]$, let $\delta_{[\varphi]}$ be a fresh constructor variable.

Define

$$\operatorname{erase}(\varphi) \equiv \begin{cases} \operatorname{erase}(\sigma) \to \operatorname{erase}(\tau) & \text{if } \varphi \equiv \sigma \to \tau, \\ \delta_{[\varphi]} & \text{otherwise.} \end{cases}$$

Define $\widetilde{S} = S \cup \{\delta_{[\varphi]} \mapsto S(\varphi) \mid \varphi \in \mathbb{C}_{\mathbb{K}}\}$.

It holds that

$$\operatorname{erase}(\widetilde{S_1}(\operatorname{erase}(S_0(\sigma_1)))) = \operatorname{erase}(S_0(\tau_1))$$
$$\operatorname{erase}(\widetilde{S_2}(\operatorname{erase}(S_0(\sigma_2)))) = \operatorname{erase}(S_0(\tau_2))$$

# System F (Church-Style)

Types are the same. Inference rules VAR, APP, ABS are same; INST/GEN is replaced by TAPP and TABS:

$$\text{TAPP}\quad \frac{A \vdash M : \forall\alpha.\tau}{A \vdash (M\,[\sigma]) : \tau[\alpha{:=}\sigma]}$$

$$\text{TABS}\quad \frac{A \vdash M : \tau; \qquad \alpha \notin \mathrm{FTV}(A)}{A \vdash (\Lambda\alpha.M) : \forall\alpha.\tau}$$

Note that the inference rules accept a different set of terms, a subset of:

$$\mathbf{\Lambda}_{\mathrm{t}} ::= \mathcal{V} \mid \lambda\mathcal{V}.\mathbf{\Lambda}_{\mathrm{t}} \mid \mathbf{\Lambda}_{\mathrm{t}}\mathbf{\Lambda}_{\mathrm{t}} \mid \Lambda\mathbb{V}.\mathbf{\Lambda}_{\mathrm{t}} \mid \mathbf{\Lambda}_{\mathrm{t}}\,[\mathbb{T}]$$

# 2UP $\leq$ Typability for Church-Style System F

Let $E = \{\langle s, t\rangle\}$ be an instance in $\mathcal{L}^{\rightarrow}$ where $s$ and $t$ are simple.

**Theorem 3. [Schubert [1998]]** *There is a $\lambda$-term $M_E$ effectively constructed from $E$ such that $E$ is 2nd-order unifiable iff $M_E$ is typable.*

$M_E$ is constructed using several techniques:

- A $\lambda$-term context can simulate fixing all or part of the type environment.
- Shared $\lambda$-term variable occurrences can force two subterms of a $\lambda$-term to have the same type.
- The above two techniques are used in an inductive process to force the type of a $\lambda$-term variable to represent the result of a single fixed 2nd-order substitution applied to a unification term.

# Proof: Fixing Part of Type Environment

Let $[M_1, \ldots, M_n] \equiv v_{\mathrm{cons}} M_1 [M_2, \ldots, M_n]$ and $[] \equiv v_{\mathrm{nil}}$, where $v_{\mathrm{cons}}$ and $v_{\mathrm{nil}}$ are fixed $\lambda$-term variables.

Let $\{c_1, \ldots, c_k\}$ be the set of constants in our signature $\Sigma$.

Let $B = \{v_{\mathrm{cons}} : \alpha \to \alpha \to \alpha, v_{\mathrm{nil}} : \alpha, v_{\mathrm{res}}^{c_1}{:}\gamma_{c_1}, \ldots, v_{\mathrm{res}}^{c_n}{:}\gamma_{c_n}\}$.

Let $C \equiv \Lambda\alpha\vec{\gamma}.\lambda v_{\mathrm{cons}} : \alpha \to \alpha \to \alpha.\lambda v_{\mathrm{nil}} : \alpha.\lambda v_{\mathrm{res}}^{c_1}{:}\gamma_{c_1}. \cdots \lambda v_{\mathrm{res}}^{c_n}{:}\gamma_{c_n}.\Box$.

**Lemma 1.** *$C[M]$ is typable iff $M$ is typable under the environment $A \cup B$ for some $A$ such that $\alpha, \vec{\gamma} \notin \mathrm{FTV}(A)$.*

All subsequent work will be done inside context $C$.

# Proof: More Techniques

- Suppose $M$ contains subterms $(xN)$ and $(xP)$. Then the derived types of the subterm occurrences $N$ and $P$ must be the same. Furthermore, if $P \equiv y$ or $N \equiv y$ for $\lambda$-term variable $y$, then the environment type of $y$ must be the same.

  (Remember: All type application is explicit.)

- In the subterm $N \equiv [M_1, \ldots, M_n]$, the derived types of $M_1$, ..., $M_n$, and $N$ are all $\alpha$.

  (Remember: $B(v_{\mathrm{cons}}) = \alpha \to \alpha \to \alpha$ and $B(v_{\mathrm{nil}}) = \alpha$.)

# Proof: Building Sets of Constraints

Let $u^* \equiv \begin{cases} [v^u_{\mathrm{eq}}(v^x_{\mathrm{subst}}(u_1{}^\dagger)\ldots(u_n{}^\dagger)), v^u_{\mathrm{eq}}v^u_{\mathrm{res}}] & \text{if } u = x(u_1,\ldots,u_n) \\ [] & \text{if } u = c_i \\ [v^u_{\mathrm{eq}}(v^u_{\mathrm{res}}v^{u_1}_{\mathrm{res}}), v^u_{\mathrm{eq}}v^{u_2}_{\mathrm{res}}, u_1{}^*, u_2{}^*] & \text{if } u = u_1 \to u_2 \end{cases}$.

Let $u^\dagger \equiv \begin{cases} (u_1{}^\dagger) \to (u_2{}^\dagger) & \text{if } u = u_1 \to u_2 \\ \gamma_c & \text{if } u = c \end{cases}$.

The term $u^*$ constitutes a set of constraints that forces the type of $v^u_{\mathrm{res}}$ to represent the result of applying to $u$ the ground 2nd-order substitution represented by the environment types for the $\lambda$-term variables $v^x_{\mathrm{subst}}$ for each $x$ occurring in $u$.

The type $u^\dagger$ for ground $u$ represents the term $u$ exactly.

# Proof: Conclusion

Let $N \equiv [v_{\mathrm{eq}}^{s,t} v_{\mathrm{res}}^{s}, v_{\mathrm{eq}}^{s,t} v_{\mathrm{res}}^{t}, s^*, t^*]$.

Let $M_E \equiv C[N]$.

$E$ is 2nd-order unifiable iff $M_E$ is typable in Church-style System F.

# 2UP $\leq$ Type Checking for $\mathsf{F}_\omega$

Let $E = \{\langle\sigma, \tau\rangle\}$ be an instance in $\mathcal{L}^{\rightarrow}$.

Let $\{\vec{\gamma}\} = \mathrm{Vars}(\sigma) \cup \mathrm{Vars}(\tau)$.

A term $x(t_1, \ldots, t_n)$ is treated as the type $xt_1 \ldots t_n$.

Let $A = \{y{:}\forall\alpha.(\alpha \to \alpha) \to \forall\alpha.\alpha, x{:}\forall\vec{\gamma}.\sigma \to \tau\}$.

**Theorem 4.** *$E$ is 2nd-order unifiable iff $A \vdash yx : \alpha$ is derivable in $\mathsf{F}_\omega$.*

Similar techniques to before are used to exclude junk from a $\mathsf{F}_\omega$-substitution to obtain a 2nd-order substitution.

Urzyczyn [1997] has a significantly more complicated variant. He calls the fact that there is a reduction a folk result and attributes it to at least himself and Tiuryn, as well as possibly others.

# F+$\eta$

F+$\eta$ is System F extended with the *subject $\eta$-reduction* rule:

$$(\eta)\quad \frac{A \vdash M : \tau; \qquad M \to_\eta N}{A \vdash N : \tau}$$

Remember that $\eta$-reduction is: $\lambda x.Mx \to_\eta M$ if $x \notin \mathrm{FV}(M)$

A more convenient presentation is to replace the INST/GEN rule by:

$$(\leq)\quad \frac{A \vdash M : \sigma; \qquad \sigma \leq \tau}{A \vdash M : \tau}$$

The relation $\leq$ was initially axiomatized by Mitchell [1988] and later by Longo et al. [1995].

# F+$\eta$: Subtyping

The following axiomatization [Wells, 1995] of $\leq$ is easier to prove with:

(tvar) $\alpha \leq \alpha$

(bot) $(\forall\alpha.\alpha) \leq \sigma$

$(\widetilde{\rightarrow})$ $$\frac{\tau_L \leq \forall\vec{\beta}.(\sigma_L[\vec{\alpha}{:=}\vec{\rho}]); \qquad \forall\vec{\beta}.(\sigma_R[\vec{\alpha}{:=}\vec{\rho}]) \leq \tau_R}{\forall\vec{\alpha}.(\sigma_L \to \sigma_R) \leq \forall\vec{\gamma}.(\tau_L \to \tau_R)}$$

where $\vec{\beta} \notin \mathrm{FTV}(\sigma_L \to \sigma_R)$ and $\vec{\gamma} \notin \mathrm{FTV}(\forall\vec{\alpha}.(\sigma_L \to \sigma_R))$

Syntax-directed: $\sigma$'s shape uniquely determines the final rule in deriving $\sigma \leq \tau$.

# SUP $\leq$ Subtyping for F+$\eta$

Let $E = \{\langle\sigma_1, \tau_1\rangle, \langle\sigma_2, \tau_2\rangle\}$ be an instance in $\mathcal{L}^{\to}$.

Let $\overline{\rho}$ stand for $\rho \to \rho$ and $\perp$ for $\forall\alpha.\alpha$. Define:

$$
\begin{array}{rcl}
\mu & = & \forall\beta.((\beta \to ((\beta \to \beta \to \perp) \to \perp) \to \beta) \to \perp) \\
\pi & = & \pi_1 \to \perp \\
\pi_1 & = & \forall\vec{\alpha}.\forall\vec{\gamma}.(\quad ((\overline{\sigma_1 \to \sigma_2}) \to \perp) \\
 & & \qquad \to ((((\overline{\tau_1 \to \alpha_1}) \to \perp) \to ((\overline{\alpha_2 \to \tau_2}) \to \perp) \to \perp) \to \perp) \\
 & & \qquad \to ((\overline{\sigma_1 \to \sigma_2}) \to \perp))
\end{array}
$$

**Theorem 5. [Wells [1995]]** *$E$ is semi-unifiable iff $\mu \leq \pi$.*

A much more complicated reduction from two-counter automata is by Tiuryn and Urzyczyn [1996].

# Proof: Ideas

Some hints as to why the proof works:

- Using the syntax-directed axiomatization allows descending into the two types in parallel, generating substitutions whenever a $\forall$-binding is passed and subtyping constraints whenever a leaf is reached in one of the types.

- Various facts about bicoercible types (i.e., $\sigma \leq \tau$ and $\tau \leq \sigma$) are used, including that (1) they must have the same underlying tree skeleton and (2) if a variable has both positive and negative occurrences (which the construction does deliberately) then quantified leaf groups and negative/positive quantifier position must be the same.

- The technique of earlier proofs for eliminating higher-order junk from the substitutions is used again.

# Open Problems

- The proof of undecidability of semi-unification [Kfoury et al., 1993], although straightforward to check, is considered to be quite hard to understand. A simpler proof would be good.
- A straightforward reduction from 2UP to SUP or vice versa would be nice. The former would aid in producing a simpler proof of the undecidability of SUP.
- A reduction from type checking to typability of $F_\omega$ would be nice, because the proof of undecidability of the latter problem is by a very difficult Turing-machine encoding [Urzyczyn, 1997].
- Second-order semi-unification seems like it could be interesting.

# Conclusion

- Proving undecidability of type search problems by reduction from undecidable unification problems often produces small, easy to understand proofs.
- Unification can be useful when used in a direction somewhat opposite of the usual.
- You all know how to do this now. Go forth and solve problems. Report back next year.

# References


Warren D. Goldfarb. Undecidability of the second-order unification problem. *Theor. Comp. Sc.*, 13:225–230, 1981.

Assaf J. Kfoury, Jerzy Tiuryn, and Paweł Urzyczyn. The undecidability of the semi-unification problem. *Inf. & Comput.*, 102(1):83–101, 1993.

Giuseppe Longo, Kathleen Milsted, and Sergei Soloviev. A logic of subtyping. In *Proc. 10th Ann. IEEE Symp. Logic in Computer Sci.*, pages 292–299, 1995.

John C. Mitchell. Polymorphic type inference and containment. *Inf. & Comput.*, 76(2/3):211–249, 1988.

Aleksy Schubert. Second-order unification and type inference for Church-style polymorphism. In *Conf. Rec. POPL '98: 25th ACM Symp. Principles of Prog. Languages*, 1998.

Jerzy Tiuryn and Paweł Urzyczyn. The subtyping problem for second-order types is undecidable. Tech. rep., Univ. of Warsaw, 1995.

Jerzy Tiuryn and Paweł Urzyczyn. The subtyping problem for second-order types is undecidable. In *Proc. 11th Ann. IEEE Symp. Logic in Computer Sci.*, 1996. Shorter proceedings version of Tiuryn and Urzyczyn [1995].

Paweł Urzyczyn. Type reconstruction in $\mathbf{F}_\omega$. *Math. Struc. in Comp. Sc.*, 7 (4):329–358, 1997.

J. B. Wells. Typability and type checking in the second-order $\lambda$-calculus are equivalent and undecidable. In *Proc. 9th Ann. IEEE Symp. Logic in Computer Sci.*, 1994. Obsoleted by Wells [199X].

J. B. Wells. The undecidability of Mitchell's subtyping relation. Tech. Rep. 95-019, Comp. Sci. Dept., Boston Univ., 1995.

J. B. Wells. Typability and type checking in $\lambda 2$-Curry and related problems. Material to appear in forthcoming book by Barendregt and Statman, 1997.

J. B. Wells. Typability and type checking in System F are equivalent and undecidable. *Ann. Pure & Appl. Logic*, 199X. To appear. Obsoletes Wells [1994].