An Algebraic Characterization of Typability in ML with Subtyping

Marcin Benke*

Institute of Informatics
Warsaw University
ul. Banacha 2
02-097 Warsaw, POLAND

e-mail: benke@mimuw.edu.pl

Abstract. We research the problem of typability in the functional language ML extended with both atomic and polymorphic subtyping. The paper analyses the interaction between polymorphism and subtyping and gives an algebraic characterization of the typability problem.

Introduction

Type reconstruction and type-checking are a form of ensuring program correctness. They prevent run-time errors and often even logical errors in program design.

Subtyping is a form of extending the type discipline which gives programmer more freedom while retaining advantages of type correctness. Since the seminal paper of Mitchell [Mit84] it has attracted a lot of interest. Several ways of extending ML polymorphic type discipline with subtyping have been proposed, but the question of complexity of type reconstruction in such systems remains open.

In this paper we and give an algebraic characterization of typability in such system, similar to one that led to establishing exact complexity of the typability in pure ML. We hope that the characterisation proposed in this paper will allow to achieve similar result for ML with subtyping.

Contributions and organization of the paper

In the first section of the paper we analyse the interaction between polymorphic and atomic subtyping. We introduce a system of subtyping for ML types being a natural restriction of the Mitchell's system [Mit88] to ML types, enriched with the subtyping between type constants. In contrast to to the Mitchell's system, the relation induced by our system turns out to be decidable (in polynomial time, actually). Furthermore, we prove that polymorphic and atomic subtyping can be in certain sense separated.

* This work has been partially supported by Polish KBN grant 8 T11C 035 14

W. Thomas (Ed.): FOSSACS'99, LNCS 1578, pp. 104-119, 1999.

In Section 2 we introduce a type system for ML with subtyping. In contrast to the previous work in this area, we try to keep this extension as simple as possible and thus do not propose to extend the syntax of types, but merely to add the subtyping rule. The resulting system turns out to behave very differently to its ancestor and enjoys some interesting (even if not very encouraging from the practical point of view) properties.

The paper [KTU90] provides an algebraic characterization of typability in ML by the "acyclic semiunification problem" (ASUP) — a variant of the unification problem. In section 3 we propose a generalization of ASUP allowing to accommodate subtyping. In the final section we show that such extension is sufficient to provide an algebraic characterization of typability in our system.

1 Preliminaries

1.1 Terms

We concentrate on a subset of ML terms essential for type analysis:

$$M ::= c \mid x \mid \lambda x.M \mid M_1 M_2 \mid \textbf{let } x = M_1 \textbf{ in } M_2$$

(x stands for variables and c for constants)

1.2 Types and type schemes

Given a set of type variables $(\alpha, \beta, \gamma, \ldots)$ and a (finite) set of type constants (like *char,int,real,...*), we define the set of (monomorphic) types

$$\tau ::= \kappa \mid \alpha \mid \tau \to \tau$$

where κ stands for type constants.

Further, we define the set of (polymorphic) type schemes

$$\sigma ::= \forall \alpha_1 \ldots \alpha_n.\tau$$

In the sequel we shall use the abbreviation $\forall \vec{\alpha}.\tau$, and a notational convention that σ (possibly with indices) will stand for type schemes and τ, ρ (possibly with indices) for (monomorphic) types.

If $\alpha \notin FV(\sigma)$ then $\forall \alpha.\sigma$ is called an empty binding. A type scheme containing only empty bindings is called singular.

1.3 Subtype partial orders

We assume there is some predefined partial ordering $\leq_\kappa$ on the type constants.

We introduce a system of subtyping for ML types being a restriction of the Mitchell's system [Mit88] to ML types, enriched with the subtyping between constants. The system derives formulas of the form $\sigma \leq \tau$, where σ and τ are type schemes.

Axioms:

(refl) $\sigma \leq \sigma$

(inst) $\forall \vec{\alpha}.\, \sigma \leq \forall \vec{\beta}.\, \sigma[\vec{\rho}/\vec{\alpha}]$, ρ_i are types; $\beta_i \notin FV(\forall \vec{\alpha}.\, \sigma)$

Rules:

$$\text{(const)} \quad \frac{\kappa_1 \leq_\kappa \kappa_2}{\kappa_1 \leq \kappa_2}$$

$$(\rightarrow) \quad \frac{\rho' \leq \rho \quad \tau \leq \tau'}{\rho \rightarrow \tau \leq \rho' \rightarrow \tau'} \qquad (\forall) \quad \frac{\sigma \leq \sigma'}{\forall \alpha.\, \sigma \leq \forall \alpha.\, \sigma'}$$

$$\text{(trans)} \quad \frac{\sigma \leq \sigma_1 \quad \sigma_1 \leq \sigma'}{\sigma \leq \sigma'}$$

We write $\vdash \sigma \leq \sigma'$ to indicate that $\sigma \leq \sigma'$ is derivable in the above system. We shall also write $\vdash_\sigma$ for derivability without using the axiom **(inst)** (and use $\vdash_\sigma \sigma_1 \leq \sigma_2$ as a a shorthand for $\sigma_1 \leq_\sigma \sigma_2$) and $\vdash_\tau$ for derivability without mentioning type schemes at all.

We shall use the symbol $\preceq$ to denote the relation generated by the **(inst)** axiom.

It is worthwhile to observe that this is not a conservative restriction of Mitchell's system, i.e. if we allow substituting polymorphic types in **(inst)**, we can infer more inequalities between ML types. A simple example here is the inequality

$$\forall \alpha.((\alpha \rightarrow \beta) \rightarrow \alpha) \leq (\alpha \rightarrow \beta) \rightarrow \gamma$$

which is derivable in the Mitchell's system but (as follows from Theorem 1.7) not in ours. On the other hand an important consequence of allowing only monomorphic instance is the following

Proposition 1.1. *The relation $\leq$ is decidable.*

Proof. This can be deduced from the results contained in [OL96]. In fact one can even prove that it is decidable in polynomial time.

Lemma 1.2. *If $\vdash \sigma \leq \sigma'$ and σ is singular then*

1. *$FV(\sigma) = FV(\sigma')$*
2. *the derivation contains only singular type schemes (in particular σ' is singular).*

Lemma 1.3. *If $\vdash \tau \leq \tau'$ (with τ, τ' monomorphic) then there is a derivation of this inequality which contains only monomorphic types.*

Lemma 1.4. *The relation $\preceq$ is reflexive and transitive.*

Lemma 1.5. *If $\sigma \leq_\sigma \sigma_1 \preceq \sigma'$ then there exists σ_2 such that*

$$\sigma \preceq \sigma_2 \leq_\sigma \sigma'.$$

Lemma 1.6. *If $\sigma_0 \preceq \sigma_1$ then $\forall\gamma.\,\sigma_0 \preceq \forall\gamma.\,\sigma_1$.*

Proof. By the definition of $\preceq$, we have $\sigma_0 \equiv \forall\vec{\alpha}.\,\sigma$ and $\sigma_1 \equiv \sigma[\vec{\rho}/\vec{\alpha}]$ for some σ, ρ. We have to consider two cases:

1. $\gamma \in FV(\forall\vec{\alpha}.\,\sigma)$. Then we have

$$\forall\gamma.\forall\vec{\alpha}.\,\sigma \preceq \forall\vec{\alpha}.\,\forall\gamma.\,\sigma[\gamma/\gamma, \vec{\alpha}/\vec{\alpha}] \equiv \forall\vec{\alpha}.\,\forall\gamma.\,\sigma \preceq \forall\gamma.\sigma[\vec{\rho}/\vec{\alpha}]$$

2. $\gamma \notin FV(\forall\vec{\alpha}.\,\sigma)$. Then we have

$$\forall\gamma.\forall\vec{\alpha}.\,\sigma \preceq \forall\vec{\alpha}.\,\sigma[\gamma/\gamma] \equiv \forall\vec{\alpha}.\,\sigma \preceq \forall\gamma.\,\sigma[\vec{\rho}/\vec{\alpha}]$$

In both cases the thesis follows from the transitivity of $\preceq$.

Theorem 1.7 (Normalization for $\leq$). *If $\vdash \sigma \leq \sigma'$ then there exists σ_1 such that*

$$\sigma \preceq \sigma_1 \leq_\sigma \sigma'$$

Proof. Again we proceed by induction on the derivation. The basic cases i.e. axioms, and (const) are trivial. The rule (trans) can be handled by Lemma 1.5.

If the last rule in the derivation was $(\rightarrow)$ then all components must be monomorphic, and by Lemma 1.3 there is a derivation of the inequality in $\vdash_\sigma$.

Having said that, we only have to consider the case when the last rule was $(\forall)$:

$$\frac{\sigma \leq \sigma'}{\forall\alpha.\,\sigma \leq \forall\alpha.\,\sigma'}$$

By the induction assumption, there exists σ_1 such that

$$\sigma \preceq \sigma_1 \leq_\sigma \sigma'.$$

But then, by Lemma 1.6 we have that

$$\forall\vec{\alpha}.\,\sigma \preceq \forall\vec{\alpha}.\,\sigma_1$$

On the other hand, obviously if $\sigma_1 \leq_\sigma \sigma'$ then $\forall\vec{\alpha}.\,\sigma_1 \leq_\sigma \forall\vec{\alpha}.\,\sigma'$.

Proposition 1.8. *In the $\leq$ ordering, every set of type schemes has a lower bound.*

Proof. It is easy to see that for every type scheme σ

$$\forall\alpha.\alpha \leq \sigma$$

$$\text{(CON)} \qquad \vdash c : \kappa(c) \quad \text{for } c \in K$$

$$\text{(VAR)} \qquad E(x : \sigma) \vdash x : \sigma$$

$$\text{(ABS)} \qquad \frac{E(x : \tau) \vdash M : \rho}{E \vdash \lambda x.M : \tau \to \rho}$$

$$\text{(APP)} \qquad \frac{E \vdash M : \tau \to \rho \quad E \vdash N : \tau}{E \vdash MN : \rho}$$

$$\text{(GEN)} \quad \frac{E \vdash M : \sigma}{E \vdash M : \forall \alpha.\sigma} \qquad \alpha \notin FV(E)$$

$$\text{(INST)} \qquad \frac{E \vdash M : \forall \alpha.\sigma}{E \vdash M : \sigma[\rho/\alpha]}$$

$$\text{(LET)} \quad \frac{E \vdash M : \sigma_0 \quad E(x : \sigma_0) \vdash N : \sigma}{E \vdash \textbf{let } x = M \textbf{ in } N : \sigma}$$

Fig. 1. A reference ML type system, $\vdash_{ML}$

2 Type systems

2.1 The traditional type system for pure ML

We assume that we have a fixed set of constants Q, and for each $c \in Q$ its type $\kappa(c)$, built only with type constants and arrows, is known.

The system depicted in Figure 1 will serve as a reference type system for ML [DM82,CDDK86,KTU90]. We shall use the symbol $\vdash_{\mathrm{ML}}$ to denote derivability in this system.

The simplest way to extend this system with subtyping is by adding the subsumption rule

$$\text{(SUB)} \quad \frac{E \vdash M : \tau \quad \tau \leq \rho}{E \vdash M : \rho}$$

In the sequel by $\vdash_{\mathrm{ML}_\leq}$ we shall understand the system $\vdash_{\mathrm{ML}}$ with the subsumption rule.

We shall say that a derivation is *normal* if subsumption rule is applied only to variables and constants.

Lemma 2.1. *If $E \vdash_{\mathrm{ML}_\leq} M : \tau$ then there is a normal derivation ending with this judgement.*

2.2 An alternative type system for ML

Kfoury et. al, in [KTU89,KTU94] suggest an alternative (equivalent) type inference system for ML, which is better suited for complexity studies:[1]

$$(\text{CON}) \qquad \vdash c : \kappa(c) \quad \text{for } c \in K$$

$$(\text{VAR}) \qquad (x : \sigma) \vdash x : \tau \qquad \text{for } \sigma \preceq \tau$$

$$(ABS) \qquad \frac{E(x : \tau) \vdash M : \rho}{E \vdash \lambda x.M : \tau \to \rho}$$

$$(APP) \qquad \frac{E \vdash M : \tau \to \rho \quad E \vdash N : \tau}{E \vdash MN : \rho}$$

$$(LET) \quad \frac{E \vdash M : \rho \quad E(x : \forall \vec{\alpha}.\rho) \vdash N : \tau}{E \vdash \textbf{let } x = M \textbf{ in } N : \tau} \quad \alpha$$

Fig. 2. An alternative type system for ML, $\vdash_{ML*}$

Proposition 2.2. *For every term M, it is typable in $\vdash_{\text{ML}^*}$ iff it is typable in $\vdash_{\text{ML}}$.*

For proof, cf. [CDDK86,KTU90].

Subtyping can be added here by replacing the instance relation in the axiom with the subtyping relation defined in the section 1.3...

$$E(x : \sigma) \vdash x : \tau \qquad \text{if } \vdash \sigma \leq \tau$$

... and modifying an axiom for constants:

$$(\text{CON}) \qquad \vdash c : \tau \qquad \text{if } \vdash \kappa(c) \leq \tau$$

We shall denote derivability in the resulting system by the symbol $\vdash_{\text{ML}^*_\leq}$.

Theorem 2.3. *For every environment E, term M and monomorphic type τ, if $\vec{\alpha} \subseteq FV(\tau) - FV(E)$ then*

$$E \vdash_{\text{ML}_\leq} M : \forall \vec{\alpha}.\tau \text{ iff } E \vdash_{\text{ML}^*_\leq} M : \tau$$

In other words for every term M, it is typable in $\vdash_{\text{ML}^*_\leq}$ iff it is typable in $\vdash_{\text{ML}_\leq}$.

[1] This system is called $\vdash^*$ in [KTU90]

Proof. The right-to-left implication becomes obvious when we observe that $\vdash_{\mathrm{ML}^\star_\leq}$ can be viewed as a subset of $\vdash_{\mathrm{ML}_\leq}$, and that under given assumption generalization over $\vec{\alpha}$ is allowed in $\vdash_{\mathrm{ML}_\leq}$.

The proof of the left-to-right implication may proceed by induction on derivation; the only difference from the Proposition 2.2 lies in the rule (SUB):

$$\text{(SUB)} \quad \frac{E \vdash M : \forall \vec{\alpha}.\, \tau \quad \forall \vec{\alpha}.\, \tau \leq \forall \vec{\beta}.\, \tau'}{E \vdash M : \forall \vec{\beta}.\, \tau'}$$

Because of the normalization theorem for $\leq$, it is sufficient to consider the cases when $\forall \vec{\alpha}.\, \tau \preceq \forall \vec{\beta}.\, \tau'$ and when $\forall \vec{\alpha}.\, \tau \leq_\sigma \forall \vec{\beta}.\, \tau'$.

In the first case it follows that $\tau' \equiv \tau[\vec{\rho}/\vec{\alpha}]$. By the induction assumption we have that

$$E \vdash_{\mathrm{ML}^\star_\leq} M : \tau$$

and want to conclude that

$$E \vdash_{\mathrm{ML}^\star_\leq} M : \tau[\vec{\rho}/\vec{\alpha}].$$

It is easy to see that

$$E[\vec{\rho}/\vec{\alpha}] \vdash_{\mathrm{ML}^\star_\leq} M : \tau[\vec{\rho}/\vec{\alpha}]$$

but since α's cannot be possibly free in E, we have that $E[\vec{\rho}/\vec{\alpha}] = E$.
Now consider the case when

$$\vdash_\sigma \forall \vec{\alpha}.\, \tau \leq \forall \vec{\beta}.\, \tau'$$

It is easy to prove that in this case $\vec{\alpha} = \vec{\beta}$ and $\vdash_\sigma \tau \leq \tau'$ with τ, τ' monomorphic. A routine check that in this case if $E \vdash_{\mathrm{ML}^\star_\leq} M : \tau$ then $E \vdash_{\mathrm{ML}^\star_\leq} M : \tau'$ is left to the reader.

Lemma 2.4. *Let M be an arbitrary term, x a free variable, occurring k times ($k \geq 1$) in M, and let $N = \lambda x_1 \ldots x_k.M'$, where M' is a term obtained from M by replacing subsequent occurrences of x with $x_1, \ldots, x_k$ respectively. Then M is typable iff N is, or, more precisely*

1. *If $E(x : \sigma) \vdash M : \tau$ then $E \vdash N : \rho_1 \to \cdots \to \rho_k \to \tau$ for some $\rho_1, \ldots, \rho_k$ such that $\sigma \leq \rho_i$ for $i = 1, \ldots, k$.*
2. *If $E \vdash N : \rho_1 \to \cdots \to \rho_k \to \tau$, then there is σ such that $E(x : \sigma) \vdash M : \tau$.*

2.3 Example

The example depicted in Fig. 3 illustrates an important difference between ML and $\mathrm{ML}_\leq$:

Assume we have two type constants i, r (one can think of them as representing for example *int* and *real*), with $i \leq r$, an atomic constant pi of type r and a functional constant *round* of type $r \to i$. Now consider the following term:

$$\textbf{let } t = (\lambda f.\, \lambda x.\, f\ (f\ x)) \textbf{ in } t\ round\ pi$$

Assumptions:
r and i are atomic type constants with $i \leq r$.

pi is an object constant of type r

$round$ is an object constant of type $r \to i$

$E_{f,x} = \{f : r \to i, x : r\}$

$E_t = \{t : (r \to i) \to r \to i\}$

Derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{E_{f,x} \vdash f : r \to i \quad \dfrac{E_{f,x} \vdash f : r \to r \quad E_{f,x} \vdash x : r}{E_{f,x} \vdash fx : r}}{E_{f,x} \vdash f(fx) : i}}{f : r \to i \vdash \lambda x.f(fx) : r \to i}}{\vdash \lambda f.\lambda x.f(fx) : (r \to i) \to r \to i} \quad \dfrac{\dfrac{E_t \vdash t : (r \to i) \to r \to i \quad \vdash round : r \to i}{E_t \vdash t\ round : r \to i} \quad E_t \vdash pi : r}{E_t \vdash t\ round\ pi : i}}{\vdash \lambda f.\lambda x.f(fx) : (r \to i) \to r \to i}}{\vdash \textbf{ let } t = \lambda f.\lambda x.f(fx) \textbf{ in } \text{t round pi : i}}$$

Fig. 3. A type derivation in $ML_{\leq}$

This term is not typable in ML but is typable in $ML_{\leq}$. In fact all its normal typings mention only monomorphic types. One of this typings is presented in Fig. 3 Here, the context in which t is used, has 'forced' inferring a monomorphic type for it, even though its definition allows to infer an universal type, e.g. $\forall\alpha.\alpha \to \alpha$.

This example illustrates consequences of the fact that $ML_{\leq}$ has no principal types property, and shows that this type system is in a way 'not compositional', whereas for ML, the following holds:

Proposition 2.5. *For any terms* N_1, N_2

$$E \vdash_{\mathrm{ML}} \ \textbf{\textit{let}}\ x = N_1\ \textbf{\textit{in}}\ N_2 : \tau$$

iff

$$E(x:\sigma) \vdash_{\mathrm{ML}} N_2 : \tau$$

where σ *is a principal type for* N_1.

3 Subtyping and Semi-Unification

3.1 Semi-unification

The Semi-Unification Problem (SUP) can be formulated as follows: An instance Γ of SUP is a set of pairs of (monomorphic) types. A substitution S is a *solution* of $\Gamma = \{(t_1, u_1), \ldots, (t_n, u_n)\}$ iff there are substitutions $R_1, \ldots, R_n$ such that

$$R_1(S(t_1)) = S(u_1), \ldots, R_n(S(t_n)) = S(u_n)$$

The problem is to decide, whether given instance has a solution.

A variation of this problem including subtyping can be formulated by redefining solution of Γ as follows: S is a *sub-solution* of Γ iff there are substitutions $R_1, \ldots, R_n$ such that

$$R_1(S(t_1)) \leq S(u_1), \ldots, R_n(S(t_n)) \leq S(u_n)$$

The Semi Sub-Unification Problem (SSUP) is to decide whether given instance has a sub-solution.

Proposition 3.1 ([KTU93]). *SUP is undecidable.*

Corollary 3.2. *SSUP is undecidable.*

3.2 Acyclic semi-unification

An instance Γ of semi-unification is *acyclic* if for some $n \geq 1$, there are integers $r_1, \ldots, r_n$ and $n+1$ disjoint sets of variables, $V_0, \ldots, V_n$, such that the pairs of Γ can be placed in n columns (possibly of different height; column i contains r_i pairs):

$$\begin{array}{cccc} (t^{1,1}, u^{1,1}) & (t^{2,1}, u^{2,1}) & \cdots & (t^{n,1}, u^{n,1}) \\ (t^{1,2}, u^{1,2}) & (t^{2,2}, u^{2,2}) & \cdots & (t^{n,2}, u^{n,2}) \\ \vdots & \vdots & \ddots & \vdots \\ (t^{1,r_1}, u^{1,r_1}) & (t^{2,r_2}, u^{2,r_2}) & \cdots & (t^{n,r_n}, u^{n,r_n}) \end{array}$$

where:

$$\begin{aligned} V_0 &= \mathrm{FV}(t^{1,1}) \cup \cdots \cup \mathrm{FV}(t^{1,r_1}) \\ V_i &= \mathrm{FV}(u^{i,1}) \cup \cdots \cup \mathrm{FV}(u^{i,r_i}) \cup \\ &\quad \cup \mathrm{FV}(t^{i+1,1}) \cup \cdots \cup \mathrm{FV}(t^{i+1,r_{i+1}}) \text{ for } 1 \leq i < n \\ V_n &= \mathrm{FV}(u^{n,1}) \cup \cdots \cup \mathrm{FV}(u^{n,r_n}) \end{aligned}$$

The set of terms with variables from V_i is also called *zone* i.

The Acyclic Semi-Unification Problem (ASUP) is the problem of deciding whether given acyclic instance has a solution.

Here again, subtyping can be introduced to yield an Acyclic Semi-Sub-Unification problem: whether a sub-solution of given acyclic instance exists.

Proposition 3.3 ([KTU90]). *ASUP is DEXPTIME-complete.*

4 The equivalence between $\mathbf{ML}_{\leq}$ and ASSUP

This section is devoted to the proof of the following

Theorem 4.1. *ASSUP is* log*-space equivalent to* $\mathbf{ML}_{\leq}$ *typability.*

The reduction from $\mathrm{ML}_{\leq}$ to ASSUP can be inferred from the original reduction from ML to ASUP given in [KTU90]. The differences are mostly of technical nature, therefore we omit it here[2] and focus on the reduction from ASSUP to typability.

4.1 Constraining terms

For every type variable α_i we introduce object variables w_i, v_i. We assume that for every type constant $\kappa \in Q$ we there is a constant c_κ of this type, and a constant $c_{(\kappa \to \kappa)}$ of type $\kappa \to \kappa$. Now, for every monomorphic type τ, we define a term M_τ and a context $C_\tau[\,]$ with one hole simultaneously by induction on τ (bear in mind that $K = \lambda x.\lambda y.x$):

1. $M_\kappa = c_\kappa$
   $C_\kappa[\,] = c_{(\kappa \to \kappa)}[\,]$

[2] The readers keen on details are referred to [Ben96].

2. $M_{\alpha_i} = v_i w_i$
   $C_{\alpha_i}[\,] = v_i[\,]$
3. $M_{\tau_1 \to \tau_2} = \lambda x.K M_{\tau_2} C_{\tau_1}[x]$
   $C_{\tau_1 \to \tau_2}[\,] = C_{\tau_2}[[\,] M_{\tau_1}]$

Intuitively, M_τ can be used wherever enforcing τ as a lower bound is needed. Dually, the context $C_\tau[\,]$ imposes τ as an upper bound for types of the term placed inside it. These intuitions are formalised in the following

Lemma 4.2. *Let $\tau, \rho_1, \ldots, \rho_\ell, \rho_1^1, \ldots, \rho_\ell^1, \rho_1^2, \ldots, \rho_\ell^2$ be arbitrary types such that*

$$\mathrm{FV}(\tau) \subseteq \{\alpha_1, \ldots, \alpha_\ell\}.$$

$$\rho_i^1 \leq \rho_i^2 \text{ for } 1 \leq i \leq \ell$$

Furthermore, let S be a substitution such that $\rho_i^1 \leq S(\alpha_i) \leq \rho_i^2$ for $1 \leq i \leq \ell$. Then for any term N and environment E such that

$$E \supseteq \{v_i : \rho_i^1 \to \rho_i^2, w_i : \rho_i \mid 1 \leq i \leq \ell\}$$

we have for every type τ'':

1. *If*
   $$E \vdash M_\tau : \tau''$$
   then $S(\tau) \leq \tau''$
2. *If*
   $$E \vdash C_\tau[N] : \tau''$$
   then
   $$E \vdash N : S(\tau)$$

Lemma 4.3. *Let $\tau, \rho_1, \ldots, \rho_\ell$ be arbitrary types such that*

$$\mathrm{FV}(\tau) \subseteq \{\alpha_1, \ldots, \alpha_\ell\}.$$

Furthermore, let S be a substitution such that $S(\alpha_i) = \rho_i$ for $1 \leq i \leq \ell$. Then for any term N and environment E such that

$$E \supseteq \{v_i : \rho_i \to \rho_i, w_i : \rho_i \mid 1 \leq i \leq \ell\}$$

we have

1. *$E \vdash M_\tau : S(\tau)$*
2. *If $E \vdash N : S(\tau)$, then there exists τ'' such that $E \vdash C_\tau[N] : \tau''$*

4.2 The encoding

Consider an instance Γ of ASSUP. Without loss of generality we may assume that all the columns of Γ have an equal number of inequalities r.

Let type variables in zone i, for $i = 0, 1, \ldots, n$, be:

$$\alpha_{i,1},\ \alpha_{i,2},\ \ldots\ ,\ \alpha_{i,\ell_i}$$

for some $\ell_i \geq 1$, corresponding to which we introduce object variables:

$$v_{i,1}, w_{i,1},\ v_{i,2}, w_{i,2},\ \ldots,\ v_{i,\ell_i}, w_{i,\ell_i}$$

The notation M_τ and $C_\tau[\,]$ introduced earlier relative to singly subscripted variables, α_i and v_i, w_i, is now extended to doubly subscripted variables, $\alpha_{i,j}$ and $v_{i,j}, w_{i,j}$. We can assume that all the zones have an equal number ℓ of variables, i.e.,

$$\ell \ = \ \ell_0 \ = \ \ell_1 \ = \ \cdots \ = \ \ell_n$$

With these assumptions about Γ, let us introduce some building blocks which shall be used in the construction of the term M_Γ:

In the sequel by $C_i^\ell[\,]$ we shall mean the context

$$\lambda w_{i,1}. \ldots \lambda w_{i,\ell}.(\lambda v_{i,1}. \ldots \lambda v_{i,\ell}.[\,])\ \underbrace{I \ldots I}_{\ell \text{ times}}$$

where $I = \lambda x.x$.

Define

$$N_1' = \lambda z.z M_{t^{1,1}} \ldots M_{t^{1,r}}$$
$$N_1 = C_0^\ell[N_1']$$

Further for $1 \leq j \leq r$, define

$$P_{1,j} = \lambda p_1 \ldots \lambda p_l.x_i\, p_1 \ldots p_l(\lambda y_1 \ldots \lambda y_r.C_{u^{1,j}}[y_j])$$

and for $2 \leq i \leq n$ and $1 \leq j \leq r$

$$P_{i,j} = \lambda p_1 \ldots \lambda p_l.x_i\, p_1 \ldots p_l(\lambda z_1 \ldots \lambda z_r \lambda y_1 \ldots \lambda y_r.C_{u^{i,j}}[y_j])$$

and

$$N_i' = \lambda z.z P_{i-1,1} \ldots P_{i-1,r} M_{t^{i,1}} \ldots M_{t^{i,r}} \tag{1}$$
$$N_i = C_i^\ell[N_i'] \tag{2}$$

Finally we define the term M_Γ as follows:

$$\begin{aligned} M_\Gamma \equiv\ & \textbf{let}\ \ x_1 \ = \ N_1 \quad \textbf{in} \\ & \textbf{let}\ \ x_2 \ = \ N_2 \quad \textbf{in} \\ & \qquad \vdots \\ & \textbf{let}\ \ x_{n-1} = N_{n-1}\ \textbf{in} \\ & \textbf{let}\ \ x_n \ = \ N_n \quad \textbf{in}\ \ N_{n+1} \end{aligned}$$

4.3 Soundness of the encoding

Theorem 4.4. *If Γ has a sub-solution then M_Γ is typable.*

Proof. Suppose Γ has a sub-solution S:

$$S = [\alpha_{i,j} := \rho_{i,j} \mid i = 0, \ldots, n, \text{ and } j = 1, \ldots, \ell\,]$$

There are therefore substitutions $R_{i,j}$ such that: $R_{i,j}(S(t^{i,j})) \leq S(u^{i,j})$, for every $i = 1, \ldots, n$ and $j = 1, \ldots, r$. We shall show that M_Γ is typable. For $i = 1, \ldots, n+1$, define the environment E_i:

$$E_i = \{v_{i-1,j} : \rho_{i-1,j} \to \rho_{i-1,j}, w_{i-1,j} : \rho_{i-1,j} \mid 1 \leq j \leq \ell\}$$

For $i = 1, \ldots, n$ and $j = 1, \ldots, r$, by Lemma 4.3:

$$E_i \vdash M_{t^{i,j}} : S(t^{i,j})$$

and for every $i = 2, \ldots, n+1$ and $j = 1, \ldots, r$ there exists a (monomorphic) type $\psi_{i,j}$, such that

$$E_i(y_j : S(u^{i-1,j})) \vdash C_{u^{i-1,j}}[y_j] : \psi_{i,j}$$

We shall prove that $\vdash N_1 : \xi_1$, where:

$$\tau_1 = \rho_{0,1} \to \cdots \to \rho_{0,\ell} \to (S(t^{1,1}) \to \cdots \to S(t^{1,r}) \to \beta_1) \to \beta_1$$

where β_1 is a type variable.

Let

$$\chi_1 = S(t^{1,1}) \to \cdots \to S(t^{1,r}) \to \beta_1$$

The desired property of N_1 is easily seen from the following derivation:

$$\dfrac{\dfrac{\dfrac{E_1(z : \chi_1) \vdash z : \chi_1 \quad E_1(z : \chi_1) \vdash M_{t^{1,1}} : S(t^{1,1})}{\vdots}}{E_1(z : \chi_1) \vdash z M_{t^{1,1}} \ldots M_{t^{1,r}} : \beta_1}}{E_1 \vdash N_1' : \chi_1 \to \beta_1}$$

$$\dfrac{\vdots}{\vdash C_1^\ell[N_1'] : \xi_1}$$

By an argument similar to the one about N_1, one may prove that

$$\begin{aligned} x_1 : \xi_1 \vdash N_2 : \rho_{1,1} \to \cdots \to \rho_{1,\ell} \to \\ (\theta_{1,1} \to \cdots \to \theta_{1,r} \to \\ S(t^{2,1}) \to \cdots \to S(t^{2,r}) \to \beta_2) \to \beta_2 \end{aligned}$$

we shall call this type ξ_2. Similarly, for $i = 3, \ldots, n+1$ and $j = 1, \ldots, r$, using the fact that $R_{i-1,j}(S(t^{i-1,j})) \leq S(u^{i-1,j})$, it is not difficult to check that:

$$\{x_{i-1} : \forall \bar{\varphi}.\, \xi_{i-1}\} \vdash N_i : \xi_i$$

where

$$\xi_i = \rho_{i-1,1} \to \cdots \to \rho_{i-1,\ell} \to (\theta_{i-1,1} \to \cdots \to \theta_{i-1,r} \to S(t^{i,1}) \to \cdots \to S(t^{i,r}) \to \beta_i) \to \beta_i$$

$$\theta_{i,j} = R_{i,j}(\rho_{i,1}) \to \cdots \to R_{i,j}(\rho_{i,\ell}) \to \psi_{i,j}$$

for some type $\psi_{i,j}$.

4.4 Completeness of the encoding

Theorem 4.5. *If M_Γ is typable then Γ has a sub-solution, i.e. there exist substitutions $S, R_{1,1}, \ldots, R_{n,r}$ such that*

$$R_{i,j}(S(t_{i,j})) \leq S(u_{i,j}) \text{ for } 1 \leq i \leq n, \quad 1 \leq j \leq r$$

Proof. Suppose that M_Γ is typable. This means that, for $i = 1, \ldots, n+1$, N_i is typable in an environment E_i of the form:

$$E_i \;=\; \{x_1 : \sigma_1, \ldots, x_{i-1} : \sigma_{i-1}\}$$

where $\sigma_1, \ldots \sigma_n$ are type schemes Although the only free variable in N_i is x_{i-1}, E_i must include a type assumption for every variable whose binding includes N_i in its scope. Note that $\emptyset = E_1 \subset \cdots \subset E_{n+1}$.

Let V_i denote the set of variables in zone i of Γ. Note that

$$FV(N_i) = \begin{cases} V_0 & \text{if } i = 1 \\ V_{i-1} \cup \{x_{i-1}\} & \text{otherwise} \end{cases} \tag{3}$$

If $N_i = C_i^\ell[N_i']$ is typable in E_i, then there exists an environment E_i^v and types $\xi_i, \rho_{i,1}^1, \ldots, \rho_{i,\ell}^1, \rho_{i,1}^2, \ldots, \rho_{i,\ell}^2$ such that

$$\rho_{i,j}^1 \leq \rho_{i,j}^2 \tag{4}$$

$$E_i^v(v_{i,j}) = \rho_{i,j}^1 \to \rho_{i,j}^2 \tag{5}$$

$$E_i^v \vdash N_i : \xi_i \tag{6}$$

Take any S such that $\rho_{i,j}^1 \leq S(\alpha_{i,j}) \leq \rho_{i,j}^1$ for $1 \leq i \leq n$, $1 \leq j \leq \ell$. The existence of such S follows from acyclicity of Γ and (3). Note that S and E_i^v satisfy assumptions of Lemma 4.2.

First let us focus on the term N_1. The type ξ_1 must be of the form

$$\xi_1 = \rho_{0,1} \to \cdots \to \rho_{0,\ell} \to (\tau_{1,1} \to \cdots \to \tau_{1,r} \to \varphi_1) \to \psi_1$$

where

$$S(t^{1,j}) \leq \tau_{1,j} \text{ for } 1 \leq j \leq r \tag{7}$$

$$\varphi_1 \leq \psi_1 \tag{8}$$

Now consider the term $P_{1,j}$. There exist an environment $E^y_{1,j} \supseteq E^v_1$ and a type $\psi_{1,j}$ such that

$$E^y_{1,j} \vdash C_{u^{i,j}} : \psi_{1,j}$$
$$E^y_{1,j} \vdash y : S(u_{1,j})$$

Since $P_{1,j}$ occurs in a context **let** $x_1 = N_1$ **in** $\ldots$, the occurrence of x_1 in it must be assigned a type ξ^j_1 that is an instance of $\forall\bar{\varphi}.\,\xi_1$. Therefore by the Lemma 1.7, there exists a substitution $R_{1,j}$ such that

$$R_{1,j}(\xi_1) \leq \xi^j_1. \tag{9}$$

From this, it follows that ξ^j_1 must be of the form

$$\xi^j_1 = \rho^j_{0,1} \to \cdots \to \rho^j_{0,\ell} \to (\tau^j_{1,1} \to \cdots \to \tau^j_{1,r} \to \varphi^j_1) \to \psi^j_1$$

since x_1 occurs in $P_{1,j}$ in the context $x_1 p_1 \ldots p_\ell(\lambda y_1 \ldots \lambda y_r . C_{u^1,j}[y_j]$, by Lemma 4.2 we have

$$\tau^j_{1,j} \leq S(u^{1,j})$$

Since $\tau_{1,j}$ occurs positively in ξ_1, we have $R_{1,j}(S(t^{1,j}) \leq R_{1,j}(\tau_{1,j})$. In turn $R_{1,j}(\tau_{1,j}) \leq \tau^j_{1,j}$ by 9. From this inequalities, we conclude that

$$R_{1,j}(S(t^{1,j})) \leq S(u^{1,j}).$$

By the same token, for $2 \leq i \leq n+1$,[3] the type ξ_i must be of the form

$$\begin{aligned}\xi_i = {} & \rho_{i-1,1} \to \cdots \to \rho_{i-1,\ell} \to \\ & (\theta_{i-1,1} \to \cdots \to \theta_{i-1,r} \to \tau_{i,1} \to \cdots \to \tau_{i,r} \to \varphi_i) \to \psi_i\end{aligned}$$

and for all j the occurrence of x_i in $P_{i,j}$ must be assigned a type of the form

$$\begin{aligned}\xi^j_i = {} & \rho^j_{i-1,1} \to \cdots \to \rho^j_{i-1,\ell} \to \\ & (\theta^j_{i-1,1} \to \cdots \to \theta^j_{i-1,r} \to \tau^j_{i,1} \to \cdots \to \tau^j_{i,r} \to \varphi^j_i) \to \psi^j_i\end{aligned}$$

and there must be a substitution $R_{i,j}$ such that

$$R_{i,j}(\xi_i) \leq \xi^j_i. \tag{10}$$

From this and from the construction of $P_{i,j}$ we can again conclude that in fact $R_{i,j}(S(t^{i,j})) \leq R_{i,j}(\tau_{i,j}) \leq \tau^j_{i,j}$ and hence

$$R_{i,j}(S(t^{i,j})) \leq S(u^{i,j})$$

for all i and j.

[3] Note that there is no x_{n+1}, but there is N_{n+1}.

References

[Ben93] Marcin Benke. Efficient type reconstruction in the presence of inheritance (extended abstract). In *Proc. Int. Symp. MFCS 1993*. Springer Verlag, 1993.

[Ben96] Marcin Benke. An algebraic characterization of typability in ML with subtyping. Technical Report TR96-14(235), Institute of Informatics, Warsaw University, December 1996. Available from `http://zls.mimuw.edu.pl/~ben/Papers/`.

[Ben98] Marcin Benke. *Complexity of type reconstruction in programming languages with subtyping*. PhD thesis, Warsaw University, 1998.

[CDDK86] D. Clément, J. Despeyroux, T. Despeyroux, and G. Kahn. A simple applicative language: Mini-ML. In *Proc. ACM Conference on Lisp and Functional Programming*, pages 13–27, 1986.

[DM82] Luis Damas and Robin Milner. Principal type-schemes for functional programs. In *Conf. Rec. ACM Symp. Principles of Programming Languages*, pages 207–211, 1982.

[FM89] You-Chin Fuh and Prateek Mishra. Polymorphic subtype inference: closing the theory-practice gap. In *Proc. Theory and Practice of Software Development*, pages 167–184, March 1989.

[FM90] You-Chin Fuh and Prateek Mishra. Type inference with subtypes. *Theoretical Computer Science*, 73:155–176, 1990.

[HM95] My Hoang and John C. Mitchell. Lower bounds on type inference with subtypes. In *Conf. Rec. ACM Symp. Principles of Programming Languages*, 1995.

[Jim96] Trevor Jim. What are principal typings and what are they good for? In *Conf. Rec. ACM Symp. Principles of Programming Languages*, pages 42–53, 1996.

[KTU89] Assaf J. Kfoury, Jerzy Tiuryn, and Paweł Urzyczyn. An analysis of ML typability. Technical Report 89-009, Boston University, 1989.

[KTU90] Assaf J. Kfoury, Jerzy Tiuryn, and Paweł. Urzyczyn. ML typability is DEXPTIME-complete. In *Proc. 15th Colloq. on Trees in Algebra and Programming*, pages 206–220. Springer LNCS 431, 1990.

[KTU93] Assaf J. Kfoury, Jerzy Tiuryn, and Paweł Urzyczyn. The undecidability of the semi-unification problem. *Information and Computation*, 102(1):83–101, January 1993.

[KTU94] Assaf J. Kfoury, Jerzy Tiuryn, and Paweł Urzyczyn. An analysis of ML typability. *Journal of the ACM*, 41(2):368–398, March 1994.

[Mit84] John C. Mitchell. Coercion and type inference. In *Conf. Rec. ACM Symp. Principles of Programming Languages*, pages 175–185, 1984.

[Mit88] John C. Mitchell. Polymorphic type inference and containment. *Information and Computation*, 76(2/3):211–249, 1988. Reprinted in *Logical Foundations of Functional Programming*, ed. G. Huet, Addison-Wesley (1990) 153–194.

[OL96] Martin Odersky and Konstantin Läufer. Putting type annotations to work. In *Conf. Rec. ACM Symp. Principles of Programming Languages*, pages 54–67, 1996.

[Smi94] Geoffrey S. Smith. Principal type schemes for functional programs with overloading and subtyping. *Science of Computer Programming*, 23:197–226, 1994.

[TU96] Jerzy Tiuryn and Paweł Urzyczyn. The subtyping problem for second-order types is undecidable. In *Proc. 11th IEEE Symposium on Logic in Computer Science*, 1996.