Contents

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: Marcin.Benke@mimuw.edu.pl

December, 1996

Abstract

We research the problem of typability in the functional laguage ML extended with subtyping. The paper analyses the interaction between polymorphism and subtyping and give an algebraic characterization of typability problem, hoping that this will open the way to establishing its exact complexity.

Introduction

Type reconstruction and type-checking are forms of ensuring program correctness. They help to avoid 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 analyse the interaction between polymorphism and subtyping 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 us to achieve similar result for ML with subtyping.

*This work has been partially supported by Polish KBN grant 8 T11C 034 10, ESPRIT BRA "Gentzen" and the EuroFoCS network.

1 Preliminaries

1.1 Terms

We consider only the core of ML relevant to typing issues. We assume that terms are given by the following grammar

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

where x stands for variables and c for (object) 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 containing only empty binding is called quasi-monomorphic.

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], enriched with the subtyping between constants. The system derives formulas of the form $\sigma \leq \sigma'$, where σ and σ' are type schemes.

(refl) $\sigma \leq \sigma$

(inst) $\forall \vec{\alpha}.\, \sigma \leq \forall \vec{\beta}.\, \sigma[\vec{\tau}/\vec{\alpha}],\ \ \beta_i \notin FV(\forall \vec{\alpha}.\, \sigma)$

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

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

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

We write $\vdash \sigma \leq \tau$ to indicate that $\sigma \leq \tau$ is derivable in the above system. We shall also write $\vdash_\sigma$ for derivability without using the axiom **(inst)** 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

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

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

1.1 Proposition
The relation $\leq$ is decidable.

1.2 Lemma
If $\vdash \sigma \leq \sigma'$ and σ is quasi-mono then

1. *$FV(\sigma) = FV(\sigma')$*
2. *the derivation contains only quasi-monotypes (in particular σ' is quasi-monomorphic).*

Proof: By induction on the derivation. The case of **(refl)** is trivial. If the derivation consists of an instance of the axiom **(inst)** then it looks like

$$\forall\vec{\alpha}.\,\sigma \leq \forall\vec{\beta}.\,\sigma[\vec{\rho}/\vec{\alpha}] \quad \beta_i \notin FV(\forall\vec{\alpha}.\,\sigma)$$

Since $\alpha \notin FV(\sigma)$ then $\sigma[\vec{\rho}/\vec{\alpha}] = \sigma$ and $\beta_i \notin FV(\sigma)$. Hence $FV(\forall\vec{\alpha}.\,\sigma) = FV(\forall\vec{\beta}.\,\sigma)$ and $\forall\vec{\beta}.\sigma$ is quasi-mono.

The cases when the derivation ends with $(\to)$ or (const) are trivial. Let us then consider a proof ending with an application of the rule $(\forall)$:

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

If $\forall\vec{\alpha}.\,\sigma$ is quasi-mono then so is σ. By induction assumption $FV(\sigma) = FV(\sigma')$ and the derivation of $\sigma \leq \sigma'$ contains only quasi-monotypes. Thus $\alpha \notin FV(\sigma')$, $FV(\forall\alpha.\,\sigma) = FV(\forall\alpha.\,\sigma')$ and whole derivation contains only quasi-monotypes.

Finally, consider the case when the last rule used was (trans):

$$\frac{\sigma \leq \sigma_1 \quad \sigma_1 \leq \sigma'}{\sigma \leq \sigma'}$$

By the induction assumption $FV(\sigma) = FV(\sigma_1)$ and σ_1 is quasi-mono. Thus we can re-apply the induction assumption to the inequality $\sigma_1 \leq \sigma'$, concluding that $FV(\sigma') = FV(\sigma_1)$ and the derivation contains only quasi-monomorphic types. Hence, the thesis follows. ∎

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

Proof: By the Lemma 1.2, the derivation of $\vdash \tau \leq \tau'$ may contain only quasi-monotypes. We shall prove that erasing all empty bindings in such a derivation yields a valid derivation. As usual, we shall proceed by induction on the derivation. Here the only interesting cases to be considered are **(inst)** and $(\forall)$. Consider an instance of the former:

$$\forall \vec{\alpha}.\, \sigma \leq \forall \vec{\beta}.\, \sigma[\vec{\rho}/\vec{\alpha}] \quad \beta_i \notin FV(\forall \vec{\alpha}.\, \sigma)$$

Since $\alpha_i \notin FV(\sigma)$ for all i, we have $\beta_i \notin FV(\sigma)$ and erasing empty bindings yields an instance of the axiom **(refl)**.

Now consider an instance of the rule $(\forall)$:

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

Since both quantifiers are empty, after erasure this rule becomes the identity step:

$$\frac{\sigma \leq \sigma'}{\sigma \leq \sigma'}$$

∎

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

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

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

Proof: If $\sigma \leq_\sigma \sigma_1$ then there exist monomorphic types τ, τ_1 such that

$$\begin{aligned} \sigma &\equiv \forall \vec{\alpha}.\tau && (1)\\ \sigma_1 &\equiv \forall \vec{\alpha}.\tau_1 && (2)\\ \tau &\leq_\sigma \tau_1 && (3) \end{aligned}$$

Thus $\sigma' \equiv \forall \vec{\beta}.\, \tau[\vec{\rho}/\vec{\alpha}]$, $\beta_i \notin FV(\sigma_1)$. It is easy to see that $\sigma_1 \equiv \forall \vec{\beta}.\, \tau[\vec{\rho}/\vec{\alpha}]$ fulfils the conditions of the thesis. ∎

1.6 Lemma
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 the following 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 $\preceq$. ∎

1.7 Theorem (Normalisation 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 the lemma 1.5.

If the last rule in the derivation was ($\rightarrow$) then all components must be monomorphic, and by the 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 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 $\forall\vec{\alpha}.\sigma_1 \leq_\sigma \forall\vec{\alpha}.\sigma'$. ∎

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

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 following will serve as a reference type system for ML:

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

$$
\begin{array}{ll}
\text{(CON)} & \vdash c : \kappa(c) \quad \text{for } c \in K \\[1ex]
\text{(VAR)} & E(x:\sigma) \vdash x : \sigma \\[1ex]
\text{(ABS)} & \dfrac{E(x:\tau) \vdash M : \rho}{E \vdash \lambda x.M : \tau \to \rho} \\[1ex]
\text{(APP)} & \dfrac{E \vdash M : \tau \to \rho \quad E \vdash N : \tau}{E \vdash MN : \rho} \\[1ex]
\text{(GEN)} & \dfrac{E \vdash M : \tau}{E \vdash M : \forall \alpha.\tau} \qquad \alpha \notin FV(E) \\[1ex]
\text{(INST)} & \dfrac{E \vdash M : \forall \alpha.\tau}{E \vdash M : \tau[\rho/\alpha]} \\[1ex]
\text{(LET)} & \dfrac{E \vdash M : \sigma \quad E(x:\sigma) \vdash N : \tau}{E \vdash \mathbf{let}\ x = M\ \mathbf{in}\ N : \tau}
\end{array}
$$

Figure 1: A reference ML type system

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

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.

2.1 Lemma
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

[KTU89] suggests an alternative (equivalent) type inference system for ML, which is more suited for complexity studies:[1]

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

For proof, cf. [CDDK86, KTU89].

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

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

[1] This system is called $\vdash^*$ in [KTU89]

$$
\begin{array}{ll}
\text{(VAR)} & (x:\sigma) \vdash x:\tau \qquad \text{for } \sigma \preceq \tau \\[1em]
(ABS) & \dfrac{E(x:\tau) \vdash M:\rho}{E \vdash \lambda x.M : \tau \to \rho} \\[1em]
(APP) & \dfrac{E \vdash M:\tau \to \rho \quad E \vdash N:\tau}{E \vdash MN:\rho} \\[1em]
(LET) & \dfrac{E \vdash M:\rho \quad E(x:\forall\vec{\alpha}.\rho) \vdash N:\tau}{E \vdash \mathbf{let}\ x = M\ \mathbf{in}\ N:\tau}
\end{array}
$$

Figure 2: An alternative type system for ML

...and adding 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 $\vdash_{\mathrm{ML}^\star_\leq}$.

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

$$E \vdash_{\mathrm{ML}_\leq} M : \forall\vec{\alpha}.\tau \textit{ iff } E \vdash_{\mathrm{ML}^\star_\leq} M:\tau$$

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

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 generalisation over $\vec{\alpha}$ is allowed in $\vdash_{\mathrm{ML}_\leq}$.

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

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

Because of the normalisation theorem for $\leq$ (1.7), it will do to consider the cases when $\forall\vec{\alpha}.\tau \preceq \forall\beta.\tau'$ and when $\forall\vec{\alpha}.\tau \leq_\sigma \forall\beta.\tau'$.

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 possibly be free in E, we have that $E[\vec{\rho}/\vec{\alpha}] = E$.

Now consider the case when

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

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

2.4 Lemma

Let M be an arbitrary term, x a free variable, occurring $k \ge 1$ times 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 $\sigma \le \rho_i$ 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 $E(x : \sigma) \vdash M : \tau$.*

Proof: Ad 1. Consider the derivation of $E(x : \sigma) \vdash M : \tau$. For ρ_i we take the type assigned by an axiom instance to i-th occurrence of x:

$$E(x : \sigma) \vdash x : \rho_i, \qquad \vdash \sigma \le \rho_i.$$

It is easy to see that

$$E(x_1 : \rho_1, \ldots, x_k : \rho_k) \vdash M' : \tau.$$

From this, the thesis follows.

Ad 2. Let σ be a lower bound of the set $\{\rho_1, \ldots, \rho_k\}$ (cf. Prop. 1.8). If $E \vdash N : \rho_1 \to \cdots \to \rho_k \to \tau$, then also

$$E(x_1 : \rho_1, \ldots, x_k : \rho_k) \vdash M' : \tau.$$

The axiom instance for each x_i must look like

$$E(x_1 : \rho_1, \ldots, x_k : \rho_k) \vdash x_i : \rho_i$$

But then also

$$E(x : \sigma) \vdash x : \rho_i$$

since $\sigma \le \rho_i$ for all i. Thus

$$E(x : \sigma) \vdash M : \tau.$$

∎

2.3 The canonical form of ML terms

Let us say that a term M is *canonical* if it is of the form:

$$\begin{array}{rcl} M & \equiv & \mathbf{let}\ x_1 = N_1\ \mathbf{in} \\ & & \mathbf{let}\ x_2 = N_2\ \mathbf{in} \\ & & \quad\vdots \\ & & \mathbf{let}\ x_{n-1} = N_{n-1}\ \mathbf{in} \\ & & \mathbf{let}\ x_n = N_n\ \mathbf{in}\ N_{n+1} \end{array}$$

for some $n \geq 0$, and where $N_1, \ldots, N_{n+1}$ do not contain **let**

2.5 Lemma ([KTU89])
Let M be an arbitrary term. We can construct a term M' in work-space logarithmic in $|M|$ such that:

1. *M' is in canonical form,*
2. *For every environment A and open type τ, $E \vdash M : \tau$ iff $E \vdash M' : \tau$.*

2.6 Lemma
Let M be a term in canonical form, τ an open type and E an environment in which all types are closed. If there is a derivation D whose last assertion is $E \vdash M : \tau$, then there is a derivation D' whose last assertion is $E \vdash M : \tau$ and in which every environment type is either open or closed.

Proof: Let $M = \textbf{let } x_1 = N_1 \textbf{ in} \ldots \textbf{let } x_n = N_n \textbf{ in } N_{n+1}$, where N_i contain no **let**'s. We shall proceed by induction on n.

- If $n = 0$, then $M = N_1$ and in derivation occur, apart from the types from E, only open types.
- For the induction step, consider a term of the form

$$M' \equiv \textbf{let } x = N \textbf{ in } M,$$

where M is in canonical form and contains at most n **let**'s, and N contains none. The last rule in D must have been

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

Where $\vec{\alpha} = \mathrm{FV}(\tau)$, since all types in E are closed. By the induction assumption, M has a derivation where every environment type is open or closed, thus so has M'.

∎

2.4 Example

The following example illustrates difference between ML and $\mathrm{ML}_{\leq}$:

Assume we have two type constants i, r, with $i \leq r$, an atomic constant pi of type r and a functional constant g of type $r \to i$. Now consider the following term:

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

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:

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:

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

Figure 3: A type derivation in $ML_{\leq}$

2.7 Proposition
For any terms N_1, N_2

$$E \vdash \textbf{\textit{let}}\ x = N_1\ \textbf{\textit{in}}\ N_2 : \tau$$

iff

$$E(x : \sigma) \vdash N_2 : \tau$$

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 open 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.

3.1 Fact
SUP is undecidable [KTU93].

3.2 Corollary
Uniform 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 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.

3.3 Fact
ASUP is DEXPTIME-complete [KTU89]

4 From $\mathrm{ML}_{\leq}$ to ASSUP

First we have to make some assumption about constants in our language. For purpose of this section we assume there is a finite number of constants $c_1, \ldots, c_p$ and that the type of the constant c_i is $\tau(c_i)$. In fact, as the next section points out, one can make some more modest assumptions about constants.

Given a closed term M in canonical form we will construct an instance Γ_M of ASSUP such that M is typable iff Γ_M has a sub-solution. We shall assume the following notational conventions about the term M:

1. The **let**-bound variables are taken from the set $\{x_0, x_1, \ldots\}$.
2. The λ-bound variables are taken from the set $\{y_0, y_1, \ldots\}$.
3. Every variable is bound exactly once

The instance Γ_M of ASSUP will be written with variables all in the set:

$$\begin{aligned}
V &= \{\beta_i \mid i \in \omega\} \cup \{\gamma_i \mid i \in \omega\} \cup \{\delta_i \mid i \in \omega\} \cup \\
\cup & \quad \{\beta_i^{(j)} \mid i,j \in \omega\} \cup \{\gamma_i^{(j)} \mid i,j \in \omega\} \cup \{\kappa_i^{(j)} \mid i,j \in \omega\}
\end{aligned}$$

We think of V as a set of type meta-variables, which is why we do not include α's in V, type expressions being written exclusively using type variables named by α's.

Suppose there is a derivation D which assigns a type to M. We think of β_i as standing for the assumed type of x_i before it is discharged in D (by an application of the rule LET), whereas $\beta_i^{(j)}$ stands for the type assigned to the j-th occurrence of x_i in M; γ_i stands for the assumed type of y_i before it is discharged (by an application of rule ABS), while $\gamma_i^{(j)}$ stands for the type assigned to the j-th occurrence of y_i in M. Finally $\kappa_i^(j)$ stands for the j-th occurrence of the constant c_i. We include the δ's in V as auxiliary variables, which will be needed in the construction of Γ_M.

Before setting up Γ_M we construct an system Δ_M of type inequalities. The definition of Δ_M is by induction on M.

Let M be in canonical form, and $M_1, M_2, \ldots, M_n$ be all the subterms of M such that, for $k = 1, \ldots, n$, if M_k is not an object variable, then $M_k = (M_i M_j)$ or $(\lambda y.M_i)$ or $(\mathbf{let}\ x = M_i\ \mathbf{in}\ M_j)$ for some $i \neq j$ and $i, j \in \{1, 2, \ldots, k-1\}$. The set $\{M_1, M_2, \ldots, M_n\}$ mentions all occurrences of the same subterm, i.e., we may have $M_i = M_j$ for $i \neq j$. Observe that $M = M_n$.

Definition of Δ_k for $k = 1, \ldots, n$:

Simultaneously with Δ_k we define a type expression t_k with variables in V, by induction on $k = 1, \ldots, n$:

1. If M_k is the j-th occurrence of x_i in M, then set $\Delta_k = \emptyset$ and $t_k = \beta_i^{(j)}$. (We number the occurrences of x_i in M with $0, 1, 2, \ldots$, starting from the left end of M. The binding occurrence $\cdots$ **let** $x_i = \cdots$ is not counted.)
2. If M_k is the j-th occurrence of y_i in M, then set $\Delta_k = \{\gamma_i \leq \gamma_i^{(j)}\}$ and $t_k = \gamma_i^{(j)}$
3. If M_k is the j-th occurrence of c_i in M, then set $\Delta_k = \{\tau(c_i) \leq \kappa_i^{(j)}\}$ and $t_k = \kappa_i^{(j)}$
4. If $M_k = (M_i M_j)$ then set $\Delta_k = \Delta_i \cup \Delta_j \cup \{t_i \leq t_j \to \delta\}$ and $t_k = \delta$, where δ is a fresh auxiliary variable.
5. If $M_k = (\lambda y_i\ M_j)$ then set $\Delta_k = \Delta_\ell$ and $t_k = \gamma_i \to t_j$.
6. If $M_k = (\mathbf{let}\ x_i = M_j\ \mathbf{in}\ M_\ell)$ then set $\Delta_k = \Delta_j \cup \Delta_\ell \cup \{t_j \leq \beta_i\}$ and $t_k = t_\ell$.

Instead of Δ_n and t_n, we also write Δ_M and t_M. ∎

Let $\mathcal{T}$ be the set of open types over variables named by α's and S be a substitution $S : V \to \mathcal{T}$. We say that S *satisfies* Δ_M ($S \models \Delta_M$) if $S(t) \leq_\tau S(u)$ for every inequality $t \leq u$ in Δ_M.

If $\tau \in \mathcal{T}$ and $\mathrm{FV}(\tau) = \{\alpha_1, \ldots, \alpha_n\}$, then we write $\forall\bar{\varphi}.\tau$ to mean $\forall\vec{\alpha}.\tau$.

4.1 Lemma

Let M be a closed term in canonical form, and τ a type. The judgement $\vdash M : \tau$ is derivable iff there is a substitution S such that:

1. $S \models \Delta_M$,
2. $S(t_M) = \tau$,
3. $\forall\bar{\varphi}.\, S(\beta_i) \leq S(\beta_i^{(j)})$ *for all relevant i and j.*

Proof: Consider a derivation of $\vdash M : \tau$ and assume that all subterms of M are numbered as in the definition of Δ_M. This means that for every $1 \leq k \leq n$ there exists a type τ_k, and an environment E_k such that:

1. $E_k \vdash M_k : \tau_k$
2. $\mathrm{dom}(E_k) = FV(M_k)$ (in particular $E_n = \emptyset$)
3. if $M_k = M_i M_j$ then $E_i = E_j = E_k$
4. if $M_k = \lambda y_i.M_j$ then $E_j = E_k(y_i : \rho_i)$ for some ρ_i

5. if $M_k = (\mathbf{let}\ x_i = M_j\ \mathbf{in}\ M_\ell)$ then $E_j = E_k$ and $E_\ell = E_k(x_i : \forall\bar{\varphi}.\, \tau_j)$

It is easy to check that a substitution S such that $S(\beta_i) = \tau_j$ (where j is the number of the term at the right side of the defining occurrence of x_i), $S(\gamma_i) = \rho_i$ for all relevant i and such that its value for the superscripted variables is determined by types in respective axiom instances, fulfills all conditions of the thesis. The only unobvious case is the **let** expression, so let us have a closer look at it:

$$\frac{E \vdash M_j : \tau_j \quad E(x_i : \forall\bar{\varphi}.\, \tau_j) \vdash M_\ell : \tau_\ell}{E \vdash \mathbf{let}\ x_i = M_j\ \mathbf{in}\ M_l : \tau_\ell}$$

Now, remembering that in this case $\Delta_k = \Delta_j \cup \Delta_\ell \cup \{t_j \leq \beta_i\}$ and $t_k = t_\ell$, observe that $S(\beta_i) = S(t_j)$, thus $S \models \Delta_k$.

For the converse implication, assume that S fulfills all conditions of the lemma and define

1. $E_n = \emptyset$,
2. if $M_k = M_i M_j$ then $E_i = E_j = E_k$
3. if $M_k = \lambda y_i.M_j$ then $E_j = E_k(y_i : S(\gamma_i))$,
4. if $M_k = (\mathbf{let}\ x_i = M_j\ \mathbf{in}\ M_\ell)$ then $E_\ell = E_k(x_i : \forall\bar{\varphi}.\, S(\beta_i))$ and $E_j = E_k$

Now, it is easy to prove by induction on the structure of M that for all $1 \leq k \leq n$

$$E_k \vdash_{\mathrm{ML}_\leq} M_k : S(t_k)$$

Thus, by the Lemma 2.3[2]

$$E_k \vdash M_k : S(t_k)$$

Again, the interesting case is **let**:

$$\frac{\dfrac{E_k \vdash M_j : S(\tau_j) \quad S(\tau_j) \leq S(\beta_i)}{E_k \vdash M_j : S(\beta_i)} \qquad E_k(x_i : \forall\bar{\varphi}.\, S(\beta_i)) \vdash M_\ell : S(\tau_\ell)}{E_k \vdash \mathbf{let}\ x_i = M_j\ \mathbf{in}\ M_l : S(\tau_\ell)}$$

∎

4.2 Lemma
The set of inequalities Δ_M can be decomposed as follows:

$$\begin{aligned}\Delta_M \quad = \quad & \Delta_{N_1} \cup \{t_{N_1} \leq \beta_1\} \cup \\ & \Delta_{N_2} \cup \{t_{N_2} \leq \beta_2\} \cup \\ & \quad\vdots \\ & \Delta_{N_n} \cup \{t_{N_n} \leq \beta_n\} \cup \\ & \Delta_{N_{n+1}}\end{aligned}$$

where for $i = 1, \ldots, n+1$:

[2] remember that in this section $\vdash$ stands for $\vdash_{\mathrm{ML}^\star_\leq}$

1. $\mathrm{FV}(\Delta_{N_i} \cup \{t_{N_i}\}) \ \cap \ \{\beta_k \mid k \in \omega\} \ = \ \emptyset$,
2. $\mathrm{FV}(\Delta_{N_i} \cup \{t_{N_i}\}) \ \cap \ \{\beta_k^{(\ell)} \mid k, \ell \in \omega\} \ \subseteq \ \{\beta_k^{(\ell)} \mid k < i \text{ and } \ell \in \omega\}$,

and for $1 \leq i < j \leq n+1$*:*

3. $\mathrm{FV}(\Delta_{N_i} \cup \{t_{N_i}\}) \ \cap \ \mathrm{FV}(\Delta_{N_j} \cup \{t_{N_j}\}) \ \subseteq \ \{\beta_k^{(\ell)} \mid k < i \text{ and } \ell \in \omega\}$.

Proof: This follows immediately from our conventions for naming variables and the definition of Δ_M. ∎

Definition of Γ_M:

Consider the decomposition of Δ_M as given in Lemma 4.2. For each of the $n+1$ lines in this decomposition we define a single inequality $T_i \leq U_i$, where $i = 1, \ldots, n+1$.

For $i = 1, \ldots, n$, if $\Delta_{N_i} = \{t_1 = u_1, \ldots, t_p = u_p\}$ where the t's and u's are type expressions over V, then:

$$\begin{aligned} T_i &= (\delta_1 \to \delta_1) \to \cdots \to (\delta_p \to \delta_p) \to (\delta_{p+1} \to \delta_{p+1}) \\ U_i &= (t_1 \to u_1) \to \cdots \to (t_p \to u_p) \to (\beta_i \to t_{N_i}) \end{aligned}$$

where the δ's are fresh auxiliary variables.

If $\Delta_{N_{n+1}} = \{t_1 = u_1, \ldots, t_q = u_q\}$ where the t's and u's are type expressions over V, then:

$$\begin{aligned} T_{n+1} &= (\delta_1 \to \delta_1) \to \cdots \to (\delta_q \to \delta_q) \\ U_{n+1} &= (t_1 \to u_1) \to \cdots \to (t_q \to u_q) \end{aligned}$$

where the δ's are fresh auxiliary variables.

Define the instance Γ'_M (not yet Γ_M) as:

$$\begin{aligned} \Gamma'_M \;=\; & \{T_1 \leq U_1\} \cup \\ & \{T_2 \leq U_2\} \cup \{\beta_1 \leq \beta_1^{(j)} \mid \beta_1^{(j)} \in \mathrm{FV}(\Delta_M)\} \cup \\ & \quad \vdots \\ & \{T_{n+1} \leq U_{n+1}\} \cup \{\beta_n \leq \beta_n^{(j)} \mid \beta_n^{(j)} \in \mathrm{FV}(\Delta_M)\} \end{aligned}$$

Γ'_M is "almost" acyclic, but not quite, with each line in this decomposition corresponding to a column. If Γ'_M is not acyclic, it can only happen because there is some i, $1 \leq i \leq n$, and a variable $\beta_i^{(j)}$ such that:

$$\beta_i^{(j)} \notin \mathrm{FV}(U_{i+1}) \qquad \text{and} \qquad \beta_i^{(j)} \in \mathrm{FV}(U_{i+m})$$

for some $m \geq 2$. For each such $\beta_i^{(j)}$, we replace the single inequality

$$\{\beta_i \leq \beta_i^{(j)}\}$$

by the following m inequalities:

$$\{\beta_i \leq \delta_1,\ \delta_1 \leq \delta_2,\ \ldots\ ,\ \delta_{m-1} \leq \beta_i^{(j)}\}$$

where the δ's are fresh auxiliary variables. Note that this irregularity may not happen to γ-variables. The resulting instance of semi-subunification is the desired Γ_M.

4.3 Lemma
Γ'_M is an instance of semi-subunification such that:

1. *If $S : V \to \mathcal{T}$ is a solution of Γ'_M in the sense of semi-subunification, then $\vdash M : \tau$ where $S(t_M) = \tau$.*
2. *If $\vdash M : \tau$ for some open type τ, then there is $S : V \to \mathcal{T}$ which is a solution of Γ'_M in the sense of semi-subunification such that $S(t_M) = \tau$.*

Proof: If $S : V \to \mathcal{T}$ is a substitution, it is also easy to check that $S \models \Delta_{N_i} \cup \{\beta_i = t_{N_i}\}$ iff S is a solution of $\{T_i \leq U_i\}$ in the sense of semi-unification, for $i = 1, \ldots, n$. Likewise, $S \models \Delta_{N_{n+1}}$ iff S is a solution of $\{T_{n+1} \leq U_{n+1}\}$ in the sense of semi-unification. Hence, by Lemma 4.1 and the definition of Γ'_M (not Γ_M), we conclude that both parts of the present lemma are true. ∎

4.4 Corollary
Γ_M is an acyclic instance such that:

1. *If $S : V \to \mathcal{T}$ is a solution of Γ_M in the sense of semi-subunification, then $\vdash M : \tau$ where $S(t_M) = \tau$.*
2. *If $\vdash M : \tau$ for some open type τ, then there is $S : V \to \mathcal{T}$ which is a solution of Γ_M in the sense of semi-subunification such that $S(t_M) = \tau$.*

Proof: It is easily checked that Γ_M is acyclic. On the other hand from the way Γ_M is obtained from Γ'_M it is obvious that every solution of Γ_M is a solution of Γ'_M and that every solution of Γ'_M can be naturally extended to a solution of Γ_M. ∎

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

Proof: This is an immediate consequence of the preceding lemma. It is not difficult to check that all the intermediary steps in the transformation from M to Γ_M are carried out in constant or logarithmic work-space. ∎

5 From ASSUP to $\mathbf{ML}_{\leq}$

5.1 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. $M_{\alpha_i} = v_i w_i$
   $C_{\alpha_i}[\ ] = v_i[\ ]$

3. $M_{\tau_1 \to \tau_2} = \lambda x.KM_{\tau_2}C_{\tau_1}[x]$
   $C_{\tau_1 \to \tau_2}[\,] = C_{\tau_2}[[\,]M_{\tau_1}]$

Intuitively, M_τ van 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

5.1 Lemma
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[3] *there exists $\tau' \leq S(\tau)$ such that*
$$E \vdash N : \tau'$$

Proof: By induction on τ:

- If $\tau = \kappa$ then $S(\tau) = \kappa$, $M_\tau = c_\kappa$ and $C_\tau[\,] = c_{(\kappa\to\kappa)}[\,]$. Obviously $E \vdash c_\kappa : \tau''$ implies $\kappa \leq \tau''$.
  For the proof of the second part, consider a generic derivation of $C_\kappa[N]$:
$$\frac{E \vdash c_{(\kappa\to\kappa)} : \tau' \to \tau'' \quad E \vdash N : \tau'}{E \vdash c_{(\kappa\to\kappa)}N : \tau''}$$
  If $E \vdash c_{(\kappa\to\kappa)} : \tau' \to \tau''$ then $(\kappa \to \kappa) \leq \tau' \to \tau''$, i.e. $\tau' \leq \kappa \leq \tau''$.
- If $\tau = \alpha_i$ then we have $S(\tau) = \rho_i$, $M_\tau = v_i w_i$, and $C_\tau[\,] = v_i[\,]$. Consider a generic type derivation M_τ:
$$\frac{E \vdash v_i : \tau' \to \tau'' \quad E \vdash w_i : \tau'}{E \vdash v_i w_i : \tau''}$$
  In every such derivation, $\rho_i^1 \to \rho_i^2 \leq \tau' \to \tau''$ and $\rho_i \leq \tau'$. From this in turn follows $\tau' \leq \rho_i^1 \leq S(\tau) \leq \rho_i^2 \leq \tau''$ and $S(\tau) \leq \tau''$.

[3]An alternative formulation: [...] then $E \vdash N : S(\tau)$

Now consider a generic type derivation for $C\tau[N]$:

$$\frac{E \vdash v_i : \tau' \to \tau'' \quad E \vdash N : \tau'}{E \vdash v_i N : \tau''}$$

In every such derivation, $\tau' \leq \rho_i^1 \leq S(\tau) \leq \rho_i^2 \leq \tau''$, hence $\tau' \leq S(\tau) \leq \tau''$.

- If $\tau = \tau_1 \to \tau_2$ then we have $S(\tau) = S(\tau_1) \to S(\tau_2)$, $C_\tau[\ ] = C_{\tau_2}[[\ ]M_{\tau_1}]$, and $M_\tau = \lambda x.KM_{\tau_2}C_{\tau_1}[x]$.

  Consider a generic type derivation for M_τ (where E_x stands for $E(x : \tau_x)$:

$$\frac{\dfrac{E_x \vdash K : \tau_2'' \to \tau_1'' \to \tau' \quad E_x \vdash M_{\tau_2} : \tau_2'' \quad E_x \vdash C_{\tau_1}[x] : \tau_1''}{E_x \vdash KM_{\tau_2}C_{\tau_1}[x] : \tau'}}{E \vdash \lambda x.KM_{\tau_2}C_{\tau_1}[x] : \tau_x \to \tau'}$$

  For every such derivation, all of the following conditions hold:

  1. $\tau_2'' \leq \tau'$ (a property of K)
  2. $S(\tau_2) \leq \tau_2''$ (by the induction assumption for M_{τ_2})
  3. $\tau_x \leq S(\tau_1)$ (since by the induction assumption for $C_{\tau_1}[\,]$, there exists τ_1' such that $\tau_1' \leq S(\tau_1)$ and $E(x : \tau_x) \vdash x : \tau_1'$)

  It is easy to check that this conditions imply $S(\tau) \leq \tau_x \to \tau'$.

  Now consider a generic derivation for NM_{τ_1}

$$\frac{E \vdash N : \tau_1' \to \tau' \quad E \vdash M_{\tau_1} : \tau_1'}{E \vdash NM_{\tau_1} : \tau'}$$

  By the induction assumption for M_{τ_1}, in every such derivation $S(\tau_1) \leq \tau_1'$. Now, by the induction assumption for C_{τ_2}, If $E \vdash C_{\tau_2}[NM_{\tau_1}] : \tau''$ then there exists τ' such that on top of the previous condition, also $\tau_1' \leq S(\tau_2)$. The conjunction of these two conditions implies in turn

$$\tau_1' \to \tau' \leq S(\tau_1 \to \tau_2)$$

which completes the proof. ∎

5.2 Lemma
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 $E \vdash C_\tau[N] : \tau''$*

Proof: By induction on τ:

- If $\tau = \kappa$ then $S(\tau) = \kappa$, $M_\tau = c_\kappa$ and $C_\tau[\ ] = c_{(\kappa\to\kappa)}[\ ]$. Obviously $E \vdash c_\kappa : \kappa$.

  For the proof of the second part, note that the following derivation is correct for all $\kappa'' \geq \kappa$:

  $$\frac{E \vdash c_{(\kappa\to\kappa)} : \kappa \to \kappa'' \quad E \vdash N : \kappa}{E \vdash c_{(\kappa\to\kappa)} N : \kappa''}$$

- If $\tau = \alpha_i$ then we have $S(\tau) = \rho_i$, $M_\tau = v_i w_i$, and $C_\tau[\ ] = v_i[\ ]$. The following type derivation shows that in fact $E \vdash M_{\alpha_i} : S(\alpha_i)$:

  $$\frac{E \vdash v_i : \rho_i \to \rho_i \quad E \vdash w_i : \rho_i}{E \vdash v_i w_i : \rho_i}$$

  Now consider the following derivation for $C_\tau[N]$:

  $$\frac{E \vdash v_i : \rho_i \to \rho_i \quad E \vdash N : \rho_i}{E \vdash v_i N : \rho_i}$$

- If $\tau = \tau_1 \to \tau_2$ then we have $S(\tau) = S(\tau_1) \to S(\tau_2)$, $C_\tau[\ ] = C_{\tau_2}[[\ ]M_{\tau_1}]$, and $M_\tau = \lambda x.KM_{\tau_2}C_{\tau_1}[x]$.

  Let $E_x = E(x : S(\tau_1))$. By the induction assumption for τ_1, there exists τ_1'' such that $E \vdash C_{\tau_1}[x] : \tau_1''$. It is easy to check that the following type derivation for M_τ is correct:

  $$\frac{\dfrac{E_x \vdash K : S(\tau_2) \to \tau_1'' \to S(\tau_2) \quad E_x \vdash M_{\tau_2} : S(\tau_2) \quad E_x \vdash C_{\tau_1}[x] : \tau_1''}{E_x \vdash KM_{\tau_2}C_{\tau_1}[x] : S(\tau_2)}}{E \vdash \lambda x.KM_{\tau_2}C_{\tau_1}[x] : S(\tau_1) \to S(\tau_2)}$$

  By the induction assumption, $E \vdash M_{\tau_1} : S(\tau_1)$, hence the following derivation for NM_{τ_1} is correct:

  $$\frac{E \vdash N : S(\tau_1) \to S(\tau_2) \quad E \vdash M_{\tau_1} : S(\tau_1)}{E \vdash NM_{\tau_1} : S(\tau_2)}$$

  Now, by the induction assumption for τ_2, there exists τ_2'' such that $E \vdash C_{\tau_2}[NM_{\tau_1}] : \tau_2''$. This completes the proof.

∎

5.2 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

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

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

$$P_{1,j} = \lambda p_1\ldots\lambda p_l.x_ip_1\ldots p_l(\lambda y_1\ldots\lambda y_r.C_{u^{1,j}}[y_j])$$

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

$$P_{i,j} = \lambda p_1\ldots\lambda p_l.x_ip_1\ldots p_l(\lambda z_1\ldots\lambda z_r\lambda y_1\ldots\lambda y_r.C_{u^{i,j}}[y_j])$$

and

$$N_i' \ = \ \lambda z.zP_{i-1,1}\ldots P_{i-1,r}M_{t^{i,1}}\ldots M_{t^{i,r}} \tag{4}$$
$$N_i \ = \ C_i^\ell[N_i'] \tag{5}$$

Finally we define the term M_Γ as follows:

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

5.3 Lemma

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 5.2:

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

and for every $i = 2, \ldots, n+1$ and $j = 1, \ldots, r$ there exists an open 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:

$$\dfrac{\dfrac{\dfrac{E_1(z:\chi_1) \vdash z : \chi_1 \quad E_1(z:\chi_1) \vdash M_{t^{1,1}} : S(t^{1,1}) \quad \cdots \quad E_1(z:\chi_1) \vdash M_{t^{1,r}} : S(t^{1,r})}{E_1(z:\chi_1) \vdash zM_{t^{1,1}} \ldots M_{t^{1,r}} : \beta_1}}{E_1 \vdash N_1' : \chi_1 \to \beta_1}}{\begin{array}{c}\vdots\\ \hline \vdash C_1^\ell[N_1'] : \xi_1\end{array}}$$

Let the type $\xi_1^{(j)}$ be defined as follows:

$$\xi_1^{(j)} = R_{1,j}(\xi_1)$$

note that thus defined $\xi_1^{(j)}$ is an instance of $\forall\bar{\varphi}.\,\xi_1$.

Now we shall prove that for every j there exists $\psi_{i,j}$ such that

$$E_2(x_1 : \xi_1) \vdash P_{i,j} : R_{1,j}(\rho_{1,1}) \to \cdots R_{1,j}(\rho_{1,\ell}) \to \psi_{i,j}$$

In the sequel we shall call the type of $P_{i,j}$ thus inferred $\theta_{i,j}$.

Observe that there exists $\psi_{1,j}$ such that

$$\begin{aligned} E_2 \vdash \lambda y_1 \ldots \lambda y_r . C_{u^{1,j}}[y_j] \quad : \quad & R_{1,j}(S(t^{1,1})) \to \cdots \to R_{1,j}(S(t^{1,j-1})) \to \\ & S(u^{1,j}) \to \\ & R_{1,j}(S(t^{1,j+1})) \to \cdots \to R_{1,j}(S(t^{1,r})) \to \psi_{1,j} \end{aligned}$$

but, since $R_{1,j}(S(t^{1,j}) \le S(u^{1,j})$, also

$$\begin{aligned} E_2 \vdash \lambda y_1 \ldots \lambda y_r . C_{u^{1,j}}[y_j] \quad : \quad & R_{1,j}(S(t^{1,1})) \to \cdots \to R_{1,j}(S(t^{1,j-1})) \to \\ & R(S(t^{1,j})) \to \\ & R_{1,j}(S(t^{1,j+1})) \to \cdots \to R_{1,j}(S(t^{1,r})) \to \psi_{1,j} \end{aligned}$$

thus in fact

$$E_2(x_1 : \forall\bar{\varphi}.\, \xi_1) \vdash P_{1,j} : \theta_{1,j}$$

Basing on this, by an argument similar to the one about N_1, it is not difficult to prove that

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

we shall call this type ξ_2.

Similarly, for $i = 3, \ldots, n+1$ and $j = 1, \ldots, r$, using the fact that $R_{i-1,j}(S(t^{i-1,j})) \le S(u^{i-1,j})$, it is not difficult to also check that:

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

where

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

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

for some type $\psi_{i,j}$. Indeed, for all relevant i, j there exists $\psi_{i,j}$ such that

$$\begin{aligned} E_{i+1} \vdash \lambda z_1 \ldots \lambda z_r \lambda y_1 \ldots \lambda y_r . C_{u^{1,j}}[y_j] \quad : \quad & \theta_{i-1,1} \to \cdots \to \theta_{i-1,r} \to \\ & R_{i,j}(S(t^{i,1})) \to \cdots \to R_{i,j}(S(t^{i,j-1})) \to \\ & S(u^{i,j}) \to \\ & R_{i,j}(S(t^{i,j+1})) \to \cdots \to R_{i,j}(S(t^{i,r})) \to \psi_{i,j} \end{aligned}$$

but, since $R_{i,j}(S(t^{i,j}) \le S(u^{i,j})$, also

$$\begin{aligned} E_{i+1} \vdash \lambda z_1 \ldots \lambda z_r \lambda y_1 \ldots \lambda y_r . C_{u^{1,j}}[y_j] \quad : \quad & \theta_{i-1,1} \to \cdots \to \theta_{i-1,r} \to \\ & R_{i,j}(S(t^{i,1})) \to \cdots \to R_{i,j}(S(t^{i,j-1})) \to \\ & R(S(t^{i,j})) \to \\ & R_{i,j}(S(t^{i,j+1})) \to \cdots \to R_{i,j}(S(t^{i,r})) \to \psi_{i,j} \end{aligned}$$

thus in fact

$$E_{i+1}(x_i : \forall\bar{\varphi}.\, \xi_i) \vdash P_{i,j} : \theta_{i,j}$$

and

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

∎

5.4 Lemma

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

$$FV(N_i = \begin{cases} V_0 & \text{if } i = 1 \\ V_{i-1} \cup \{x_{i-1}\} & \text{otherwise} \end{cases} \tag{6}$$

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

$$\rho_{i,j}^1 \leq \rho_{i,j}^2 \tag{7}$$
$$E_i^v(v_{i,j}) = \rho_{i,j}^1 \to \rho_{i,j}^2 \tag{8}$$
$$E_i^v \vdash N_i : \xi_i \tag{9}$$

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 (6). Note that S and E_i^v satisfy assumptions of Lemma 5.1.

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

$$S(t^{1,j}) \leq \tau_{1,j} \text{ for } 1 \leq j \leq r \tag{10}$$
$$\varphi_1 \leq \psi_1 \tag{11}$$

Now consider the term $P_{1,j}$. There exist an environment $E_{1,j}^y \supseteq E_1^v$ and a type $\psi_{1,j}$ such that

$$E_{1,j}^y \vdash C_{u^{i,j}} : \psi_{1,j}$$

$$E_{1,j}^y \vdash y : S(u_{1,j})$$

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_1^j 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

$$R_{1,j}(\xi_1) \leq \xi_1^j. \tag{12}$$

From this, it follows that ξ_1^j must be of the form

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

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 5.1 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 (12). From this inequalities, we conclude that

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

By the same token, for $2 \leq i \leq n+1$,[4] the type ξ_i must be of the form

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

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

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

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

$$R_{i,j}(\xi_i) \leq \xi^j_i. \tag{13}$$

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. ∎

5.5 Lemma
ASSUP is log-*space reducible to* $\mathbf{ML}_\leq$ *typability.*

Proof: It is easy to see, that the construction of M_Γ shown above can be indeed done using logarithmic workspace. ∎

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

[4] 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.

[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.

[Dam84] Luis Manuel Martins Damas. *Type assignment in programming languages.* PhD thesis, University of Edinburgh, 1984.

[FM89] Y. Fuh and P. 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.

[Hen88] F. Henglein. Type Inference and Semi-Unification. In *LISP*, pages 184–197, ACM Press, July 1988.

[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.

[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.

[LMS95] G. Longo, K. Milsted, and S. Soloviev. A logic of subtyping. In *Proc. IEEE Symp. on Logic in Computer Science*, pages 292–299, 1995.

[Mai90] H.G. Mairson. Deciding ML typability is complete for deterministic exponential time. In *Proc. 17th ACM Symp. Principles of Programming Languages*, pages 382–401, January 1990.

[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.

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

[Tiu95] J. Tiuryn. Equational axiomatization of bicoercibility for polymorphic types. In Ed. P.S. Thiagarajan, editor, *Proc. 15th Conference Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *Lecture Notes in Computer Science*, pages 166–179. Springer Verlag, 1995.

[Tiu96] Jerzy Tiuryn. A sequent calculus for subtyping polymorphic types. Technical Report TR 96-01(222), Institute of Informatics, Warsaw University, April 1996.

[TU96] J. Tiuryn and P. Urzyczyn. The subtyping problem for second-order types is undecidable. In *Proc. LICS 96, to appear*, 1996.

[Wel94] J. Wells. Typability and type checking in the second-order λ-calculus are equivalent and undecidable. In *Proc. 9th Ann. IEEE Symp. Logic in Comput. Sci.*, pages 176–185, 1994.

[Wel95] J. Wells. The undecidability of Mitchell's subtyping relation. Technical report, Computer Sci. Dept., Boston University, December 1995.