BETA-REDUCTION AS UNIFICATION

(Preliminary Draft)

A.J. Kfoury*
Department of Computer Science
Boston University
Boston, MA 02215, U.S.A.

July 8, 1996

Contents

*Partly supported by NSF grant CCR–9417382.

1 Introduction

It is certainly possible to define a unification problem $\mathcal{U}$ with the property that, given a λ-term M, we can derive an instance Δ of $\mathcal{U}$ from M such that if Δ has a solution then M is β-strongly normalizable. For example, $\mathcal{U}$ can be standard first-order unification **1UP**, and Δ the set of first-order constraints $\gamma(M)$ characterizing the typability of M in the simply-typed λ-calculus: If $\gamma(M)$ has a solution, then M is simply-typable and, *a fortiori*, β-strongly normalizable. But, of course, the converse is not true: If M is β-strongly normalizable, $\gamma(M)$ does not necessarily have a solution.

Our first task is this: The definition of a unification problem $\wedge$**UP** with the property that, given a λ-term M, we can derive an instance $\Gamma(M)$ of $\wedge$**UP** from M such that $\Gamma(M)$ has a solution if and only if M is β-strongly normalizable. There is a type discipline for pure λ-terms that characterizes β-strong normalization; this is the system of intersection types (without a "top" type that can be assigned to every λ-term). In this report, we use a lean version of the usual system of intersection types, which we call $\boldsymbol{\lambda}^{\rightarrow,\wedge}$. Hence, $\wedge$**UP** is also an appropriate unification problem to characterize typability of λ-terms in $\boldsymbol{\lambda}^{\rightarrow,\wedge}$. Quite apart from the new light it sheds on β-reduction, such an analysis turns out to have several other benefits.

2 A Unification Problem

We define a unification problem, here called $\wedge$**UP**, which gives an appropriate algebraic characterization of β-strong normalization. The result is proved in Section 3 and again, differently, in Section 7. Formally, $\mathrm{TVar} = \{\alpha^i_{\mathbf{p}} \mid i \in \mathbb{P} \cup \{\varepsilon\},\ \mathbf{p} \in \mathbb{P}^+\}$ is the set of *type variables*, and $\wedge\text{-Var} = \{d_{\mathbf{p}} \mid \mathbf{p} \in \mathbb{P}^+\}$ is the set of $\wedge$*-variables.*

Definition 2.1 (types) The set of types $\mathbb{T}$ is a proper subset of the usual intersection types over one type constant, denoted $\square$, defined inductively:

1. $\square \in \mathbb{T}^{\rightarrow}$.
2. If $\sigma \in \mathbb{T}^{\rightarrow} \cup \mathbb{T}^{\wedge}$ and $\tau \in \mathbb{T}^{\rightarrow}$ then $(\sigma \rightarrow \tau) \in \mathbb{T}^{\rightarrow}$.
3. If $\sigma_1, \ldots, \sigma_n \in \mathbb{T}^{\rightarrow}$ and $n \geqslant 2$ then $(\sigma_1 \wedge \cdots \wedge \sigma_n) \in \mathbb{T}^{\wedge}$.

Let $\mathbb{T} = \mathbb{T}^{\rightarrow} \cup \mathbb{T}^{\wedge}$.

Definition 2.2 (type schemes over TVar and $\wedge$-Var) Let $A \subseteq \wedge\text{-Var}$ and $B \subseteq \mathrm{TVar}$. By simultaneous induction we define three disjoint sets of type schemes over A and B, namely $\mathbb{T}^{\rightarrow}(A,B)$, $\mathbb{T}^{\wedge}(A,B)$ and $\mathbb{T}^{\bullet}(A,B)$:

1. $\{\square\} \cup B \subset \mathbb{T}^{\rightarrow}(A,B)$.

2. If $\sigma \in \mathbb{T}^{\rightarrow}(A,B) \cup \mathbb{T}^{\wedge}(A,B) \cup \mathbb{T}^{\bullet}(A,B)$ and $\tau \in \mathbb{T}^{\rightarrow}(A,B)$ then $(\sigma \rightarrow \tau) \in \mathcal{T}^{\rightarrow}$.

3. If $\sigma_1, \ldots, \sigma_n \in \mathbb{T}^{\rightarrow}(A,B) \cup \mathbb{T}^{\bullet}(A,B)$ and $n \geqslant 2$ then $(\sigma_1 \wedge \cdots \wedge \sigma_n) \in \mathbb{T}^{\wedge}(A,B)$.

4. If $\sigma \in \mathbb{T}^{\rightarrow}(A,B) \cup \mathbb{T}^{\bullet}(A,B)$ and $d \in A$ then $d\sigma \in \mathbb{T}^{\bullet}(A,B)$.

The set of *type schemes* over A and B is $\mathbb{T}(A,B) = \mathbb{T}^{\rightarrow}(A,B) \cup \mathbb{T}^{\wedge}(A,B) \cup \mathbb{T}^{\bullet}(A,B)$. A particular case is $A = \wedge\text{-Var}$ and $B = \text{TVar}$, for which we define:

$$\mathcal{T}^{\rightarrow} = \mathbb{T}^{\rightarrow}(\wedge\text{-Var}, \text{TVar}) \quad \text{and} \quad \mathcal{T}^{\wedge} = \mathbb{T}^{\wedge}(\wedge\text{-Var}, \text{TVar}) \quad \text{and} \quad \mathcal{T}^{\bullet} = \mathbb{T}^{\bullet}(\wedge\text{-Var}, \text{TVar})$$

Let $\mathcal{T} = \mathcal{T}^{\rightarrow} \cup \mathcal{T}^{\wedge} \cup \mathcal{T}^{\bullet}$. For $\sigma \in \mathcal{T}$, we denote the set of type variables occurring in σ by $\text{TVar}(\sigma)$, and the set of $\wedge$-variables by $\wedge\text{-Var}(\sigma)$. We identify appropriate subsets of $\mathcal{T} = \mathbb{T}(\wedge\text{-Var}, \text{TVar})$:

$$\mathbb{T}(\wedge\text{-Var}) = \mathbb{T}(\wedge\text{-Var}, \varnothing) = \{\ \sigma \in \mathcal{T} \mid \text{TVar}(\sigma) = \varnothing\ \}$$

$$\mathbb{T}(\text{TVar}) = \mathbb{T}(\varnothing, \text{TVar}) = \{\ \sigma \in \mathcal{T} \mid \wedge\text{-Var}(\sigma) = \varnothing\ \}$$

Consistent with this notation, $\mathbb{T} = \mathbb{T}(\varnothing, \varnothing)$.

Conventions 2.3

1. Parentheses are omitted in formal type expressions whenever convenient, provided no ambiguity is introduced:
   (a) As usual, $\tau_1 \rightarrow \tau_2 \rightarrow \tau_3$ is shorthand for $(\tau_1 \rightarrow (\tau_2 \rightarrow \tau_3))$.
   (b) $\wedge$ binds more strongly than $\rightarrow$, so that $\tau_1 \wedge \tau_2 \rightarrow \tau_3$ is shorthand for $(\tau_1 \wedge \tau_2) \rightarrow \tau_3$, not $\tau_1 \wedge (\tau_2 \rightarrow \tau_3)$.

2. Viewing $\wedge$ as a binary constructor, other studies take it commutative and associative, and sometimes also idempotent. Here, $\wedge$ is only associative, and neither commutative nor idempotent.

3. $\wedge$-variables distribute over $\wedge$, but not $\rightarrow$, i.e. we take $d(\tau_1 \wedge \tau_2)$ as shorthand for $d\tau_1 \wedge d\tau_2$. With this convention we can restrict $\sigma_1, \ldots, \sigma_n$ in clause 3 (resp. σ in clause 4) of Definition 2.2 to be in the subset $\mathcal{T}^{\rightarrow} \cup \mathcal{T}^{\bullet}$ rather than in the full set $\mathcal{T}$.

4. $\tau_1 \equiv \tau_2$ means "τ_1 and τ_2 are syntactically identical" modulo the preceding conventions.

Convention 2 above becomes important when we later construct a solution for a set of constraints: We will need to keep track of which $\wedge$-component in a type scheme $\sigma \in \mathcal{T}^\wedge$ corresponds to which $\wedge$-component in another type scheme $\tau \in \mathcal{T}^\wedge$. Convention 3 is not essential but allows several definitions to be written more compactly and clearly (see Remark 2.9, Definitions 4.2 and 4.3).

Definitions 2.4 and 2.5 introduce appropriate restrictions satisfied by type schemes derived from λ-terms (Section 3).

Definition 2.4 ($\wedge$-contexts) The set of $\wedge$-contexts is $\mathcal{C} = (\wedge\text{-Var})^*$. Every $\wedge$-context c is therefore a sequence $d_{\mathbf{p}_1} d_{\mathbf{p}_2} \cdots d_{\mathbf{p}_n}$ of $\wedge$-variables, for some $n \geqslant 0$. We define a function $\wedge$-*context* from $(\text{TVar} \cup \wedge\text{-Var}) \times \mathcal{T}$ to finite subsets of $\mathcal{C}$. First define $\wedge\text{-}\mathit{context}(\alpha, \tau)$ for every $\alpha \in \text{TVar}$ and $\tau \in \mathcal{T}$, by induction:

1. $\wedge\text{-}\mathit{context}(\alpha, \square) = \varnothing$ and $\wedge\text{-}\mathit{context}(\alpha, \beta) = \begin{cases} \{\varepsilon\}, & \text{if } \alpha \equiv \beta, \\ \varnothing, & \text{if } \alpha \not\equiv \beta. \end{cases}$
2. $\wedge\text{-}\mathit{context}(\alpha, \sigma \to \tau) = \wedge\text{-}\mathit{context}(\alpha, \sigma) \cup \wedge\text{-}\mathit{context}(\alpha, \tau)$.
3. $\wedge\text{-}\mathit{context}(\alpha, \sigma_1 \wedge \cdots \wedge \sigma_n) = \wedge\text{-}\mathit{context}(\alpha, \sigma_1) \cup \cdots \cup \wedge\text{-}\mathit{context}(\alpha, \sigma_n)$.
4. $\wedge\text{-}\mathit{context}(\alpha, d\sigma) = \{dc \mid c \in \wedge\text{-}\mathit{context}(\alpha, \sigma)\}$.

We next define $\wedge\text{-}\mathit{context}(d, \tau)$ for every $d \in \wedge\text{-Var}$ and $\tau \in \mathcal{T}$, by induction:

1. $\wedge\text{-}\mathit{context}(d, \square) = \varnothing$ and $\wedge\text{-}\mathit{context}(d, \alpha) = \varnothing$.
2. $\wedge\text{-}\mathit{context}(d, \sigma \to \tau) = \wedge\text{-}\mathit{context}(d, \sigma) \cup \wedge\text{-}\mathit{context}(d, \tau)$.
3. $\wedge\text{-}\mathit{context}(d, \sigma_1 \wedge \cdots \wedge \sigma_n) = \wedge\text{-}\mathit{context}(d, \sigma_1) \cup \cdots \cup \wedge\text{-}\mathit{context}(d, \sigma_n)$.
4. $\wedge\text{-}\mathit{context}(d, d'\sigma) = \begin{cases} \{d'c \mid c \in \wedge\text{-}\mathit{context}(d, \sigma)\}, & \text{if } d \not\equiv d', \\ \{\varepsilon\} \cup \{d'c \mid c \in \wedge\text{-}\mathit{context}(d, \sigma)\}, & \text{if } d \equiv d'. \end{cases}$

Definition 2.5 (well-behaved type schemes) A type scheme $\tau \in \mathcal{T}$ is well-behaved if it satisfies 4 conditions:

1. For every $\alpha \in \text{TVar}(\tau)$, $\wedge\text{-}\mathit{context}(\alpha, \tau)$ is a singleton set.
2. For every $d \in \wedge\text{-Var}(\tau)$, $\wedge\text{-}\mathit{context}(d, \tau)$ is a singleton set.
3. For all $\alpha^i_{\mathbf{p}}, \alpha^j_{\mathbf{q}} \in \text{TVar}(\tau)$, if $\mathbf{p} \neq \mathbf{q}$ then neither $\mathbf{p}$ nor $\mathbf{q}$ is a prefix of the other.
4. For all $d_{\mathbf{p}}, d_{\mathbf{q}} \in \wedge\text{-Var}(\tau)$, if $\mathbf{p} \neq \mathbf{q}$ then neither $\mathbf{p}$ nor $\mathbf{q}$ is a prefix of the other.

In words, conditions 1 and 2 require that the $\wedge$-context of every type variable and the $\wedge$-context of every $\wedge$-variable be uniquely defined. None of these 4 conditions is implied by the others. This is clear for conditions 3 and 4. The next example shows the independence of conditions 1 and 2.

Example 2.6 Let $\sigma \equiv dd'\alpha \wedge d'\alpha' \to \beta$. Then $\wedge$-*context*$(\alpha,\sigma) = \{dd'\}$, $\wedge$-*context*$(\alpha',\sigma) = \{d'\}$ and $\wedge$-*context*$(\beta,\sigma) = \{\varepsilon\}$. Thus σ satisfies condition 1 in Definition 2.5, but does not satisfy condition 2, because $\wedge$-*context*$(d',\sigma) = \{\varepsilon, d\}$. Now let $\tau \equiv \alpha \wedge d\alpha \wedge dd'\alpha \to \beta$. Then $\wedge$-*context*$(d,\tau) = \{\varepsilon\}$ and $\wedge$-*context*$(d',\tau) = \{d\}$, which implies that τ satisfies condition 2 in Definition 2.5. But τ does not satisfy condition 1, because $\wedge$-*context*$(\alpha,\tau) = \{\varepsilon, d, dd'\}$.

Definition 2.7 (renaming functions) A *renaming function* f has two disjoint parts: an injection from TVar to TVar, and an injection from $\wedge$-Var to $\wedge$-Var, extended by induction on $\mathcal{T}$ to $f : \mathcal{T} \to \mathcal{T}$. A particular kind of renaming functions is now defined, others are considered later. For every $j \in \mathbb{P}$, define the renaming function $\langle\ \rangle_j : \mathcal{T} \to \mathcal{T}$, by induction:

1. $\langle \Box \rangle_j = \Box$ and $\langle \alpha^i_{\mathbf{p}} \rangle_j = \alpha^i_{\mathbf{p},j}$.
2. $\langle \sigma \to \tau \rangle_j = \langle \sigma \rangle_j \to \langle \tau \rangle_j$.
3. $\langle \sigma_1 \wedge \cdots \wedge \sigma_n \rangle_j = \langle \sigma_1 \rangle_j \wedge \cdots \wedge \langle \sigma_n \rangle_j$.
4. $\langle d_{\mathbf{p}}\ \sigma \rangle_j = d_{\mathbf{p},j}\ \langle \sigma \rangle_j$.

Definition 2.8 (valuations of $\wedge$-Var) A *valuation* φ *of* $\wedge$*-variables* is a map $\varphi : \wedge\text{-Var} \to \mathbb{P}$ such that $\varphi(d) = 1$ for almost all $d \in \wedge$-Var. Such a valuation is extended to $\varphi : \mathbb{T}(\wedge\text{-Var}, \text{TVar}) \to \mathbb{T}(\text{TVar})$ by induction on $\mathcal{T} = \mathbb{T}(\wedge\text{-Var}, \text{TVar})$:

1. $\varphi(\Box) = \Box$ and $\varphi(\alpha) = \alpha$.
2. $\varphi(\sigma \to \tau) = \varphi(\sigma) \to \varphi(\tau)$.
3. $\varphi(\sigma_1 \wedge \cdots \wedge \sigma_n) = \varphi(\sigma_1) \wedge \cdots \wedge \varphi(\sigma_n)$.
4. $\varphi(d\sigma) = \varphi(\langle \sigma \rangle_1) \wedge \cdots \wedge \varphi(\langle \sigma \rangle_{\varphi(d)})$.

Remark 2.9 Keep in mind that $\wedge$-variables distribute over $\wedge$ (3 in Conventions 2.3), e.g. $d(\alpha_1 \wedge \alpha_2)$ is shorthand for $d\alpha_1 \wedge d\alpha_2$. If this were not the case, and say $\varphi(d) = 2$, we would have:

$$\varphi(d(\alpha_1 \wedge \alpha_2)) = \alpha_{1,1} \wedge \alpha_{2,1} \wedge \alpha_{1,2} \wedge \alpha_{2,2} \neq \alpha_{1,1} \wedge \alpha_{1,2} \wedge \alpha_{2,1} \wedge \alpha_{2,2} = \varphi(d\alpha_1 \wedge d\alpha_2)$$

The inequality is a consequence of the non-commutativity of $\wedge$ (2 in Conventions 2.3). It is certainly possible to relax or altogether omit these conventions, but then some of the later definitions become more complicated to formulate.

Definition 2.10 (valuations of TVar) A *valuation ψ of type variables* (a "substitution") is a total function $\psi : \text{TVar} \to \mathbb{T}^{\to}$ such that $\psi(\alpha) = \Box$ for almost all $\alpha \in \text{TVar}$, extended in the usual way to $\psi : \mathbb{T}(\text{TVar}) \to \mathbb{T}$ by induction on $\mathbb{T}(\text{TVar})$:

1. $\psi(\Box) = \Box$.

2. $\psi(\sigma \to \tau) = \psi(\sigma) \to \psi(\tau)$.

3. $\psi(\sigma_1 \wedge \cdots \wedge \sigma_n) = \psi(\sigma_1) \wedge \cdots \wedge \psi(\sigma_n)$.

Note that the range of ψ is restricted to $\mathbb{T}^{\to}$, a proper subset of $\mathbb{T}$. Without this restriction, several things go awry later (see, in particular, Example 4.7).

Lemma 2.11

1. *If $\sigma \in \mathcal{T}$ is well-behaved, then so is $\langle\sigma\rangle_j$, for every $j \in \mathbb{P}$.*

2. *If $\sigma \in \mathcal{T}$ is well-behaved and $\varphi : \wedge\text{-Var} \to \mathbb{P}$, then $\varphi(\sigma)$ is well-behaved.*

Proof: Part 1 is clear from Definitions 2.5 and 2.7. A formal proof is by induction on $\mathcal{T}$. Part 2 is by induction on $\mathcal{T}$, using part 1 also. ∎

Definition 2.12 (unification instances of $\wedge$UP) A *constraint* is an equation of the form $\sigma = \tau$ where $\sigma, \tau \in \mathcal{T}$. The constraint $\sigma = \tau$ is *well-behaved* if the type scheme $\sigma \wedge \tau$ is well-behaved. An *instance* Δ of $\wedge$**UP** is a well-behaved finite set of constraints, i.e.

$$\Delta = \{\sigma_1 = \tau_1,\ \sigma_2 = \tau_2,\ \ldots\ ,\ \sigma_n = \tau_n\}$$

such that the type scheme $\sigma_1 \wedge \tau_1 \wedge \sigma_2 \wedge \tau_2 \wedge \cdots \wedge \sigma_n \wedge \tau_n$ is well-behaved. A *solution* for Δ is a pair (φ, ψ) where $\varphi : \wedge\text{-Var} \to \mathbb{P}$ (a valuation for $\wedge$-Var) and $\psi : \text{TVar} \to \mathbb{T}^{\to}$ (a valuation for TVar), such that $\psi(\varphi(\sigma_i)) \equiv \psi(\varphi(\tau_i))$ for $i = 1, \ldots, n$, in which case we write $(\varphi, \psi) \models \Delta$. A particular case is $\Delta = \varnothing$, the empty set of constraints, which always has a solution.

Sometimes we say $\varphi : \wedge\text{-Var} \to \mathbb{P}$ is a *solution* of Δ, and simply write $\varphi \models \Delta$, if there is a valuation $\psi : \text{TVar} \to \mathbb{T}^{\to}$ (in general not unique) such that $(\varphi, \psi) \models \Delta$.

Notions and functions defined earlier in this section for type schemes are extended to constraint sets in the obvious way. For example, if Δ is the constraint set above, then

$$\text{TVar}(\Delta) = \text{TVar}(\sigma_1 \wedge \tau_1 \wedge \cdots \wedge \sigma_n \wedge \tau_n)$$

The sets $\wedge\text{-Var}(\Delta)$, $\wedge$-*context*(α, Δ), $\wedge$-*context*(d, Δ), etc., are defined similarly.

3 Typability in the System of Intersection Types

The two conditions in the next definition are standard in the literature, whenever constraints are formulated in relation to typability of λ-terms in a type inference system.

Definition 3.1 (well-named $\boldsymbol{\lambda}$-terms) A λ-term $M \in \Lambda$ is *well-named* if it satisfies two conditions:

1. No variable in M has more than one binding occurrence.

2. The bound and free variables in M are disjoint sets.

Lemma 3.2 *For every λ-term $M \in \Lambda$, we can effectively define a well-named λ-term $N \in \Lambda$ such that $M \equiv_\alpha N$.*

Proof: By induction on the definition of M. ∎

Let $M \in \Lambda$. Formally, the set of λ-variables is $\lambda\text{-Var} = \{x_i \mid i \in \mathbb{P}\}$. For the same variable x_i, the occurrences of x_i (free or bound but not binding) in M are uniquely identified by their occurrence numbers, as

$$x_i^{(j_1)}, x_i^{(j_2)}, \ \dots \ , x_i^{(j_n)}$$

for some $n \geqslant 0$ and $j_1, j_2, \dots, j_n \in \mathbb{P}$. Subscripts are part of the variable name, superscripts are not. For simplicity of notation, we often assign occurrence numbers consecutively, starting with 1, as M is scanned from left-to-right, as

$$x_i^{(1)}, x_i^{(2)}, \ \dots \ , x_i^{(n)}$$

but our analysis is independent of this numbering scheme. All that matters is that an occurrence of x_i in M is uniquely identified by an occurrence number.

We define a procedure which, given a well-named λ-term M, generates a finite set $\Gamma(M)$ of constraints. If Δ is a set of constraints and $d \in \wedge\text{-Var}$, we write $d\Delta$ to denote the set of constraints:

$$d\Delta = \ \{d\sigma = d\tau \mid \sigma = \tau \text{ is a constraint in } \Delta\}$$

The constraints in $\Gamma(M)$ do not mention the type constant $\square$, and are written over proper subsets of TVar and $\wedge$-Var, namely the subsets:

$$\text{TVar}_1 = \ \{ \ \alpha_i^j \mid i \in \mathbb{P}, j \in \mathbb{P} \cup \{\varepsilon\} \ \} \quad \text{and} \quad \wedge\text{-Var}_1 = \ \{ \ d_i \mid i \in \mathbb{P} \ \}$$

For a fixed $i \in \mathbb{P}$, all type variables of the form α_i^j correspond to λ-variables x_i. For convenience, distinguish a subset TVar_{aux} of TVar_1 whose members do not correspond to λ-variables in M:

$$\text{TVar}_{aux} = \{ \alpha_i \mid i \in \mathbb{P},\ x_i \text{ does not occur in } N \}$$

We reserve β (possibly decorated) as a metavariable to range over TVar_{aux} ("aux" is for "auxiliary").

For later purposes we need to generate the set of constraints $\Gamma(M)$ with polarities inserted, i.e. with a sign "+" or "−" inserted in front of every type variable. Polarities are bookkeeping markers, which are not part of the syntax of type schemes.

Definition 3.3 (procedure Γ to generate constraints) Simultaneously with $\Gamma(M)$, we define a type scheme $\tau(M)$ and, for the λ-variable occurrence $x_i^{(j)}$ in M, a $\wedge$-context $\theta(i,j,M)$. It will turn out that $\wedge$-*context*$(\alpha_i^j, \Gamma(M)) = \theta(i,j,M)$. By induction on well-named λ-terms:

1. *Variables* $x_i^{(j)}$:
   - $\Gamma(x_i^{(j)}) = \varnothing$.
   - $\tau(x_i^{(j)}) = (+\alpha_i^j)$.
   - $\theta(k,\ell,x_i^{(j)}) = \begin{cases} \varepsilon, & \text{if } k = i \text{ and } \ell = j, \\ \bot, & \text{if } k \neq i \text{ or } \ell \neq j. \end{cases}$
2. *Applications* (NP):
   - $\Gamma(NP) = \Gamma(N) \cup d\Gamma(P) \cup \{\tau(N) = d\tau(P) \to (-\beta)\}$,
     for a fresh $d \in \wedge\text{-Var}_1$ and a fresh $\beta \in \text{TVar}_{aux}$.
   - $\tau(NP) = (+\beta)$.
   - $\theta(k,\ell,(NP)) = \begin{cases} \theta(k,\ell,N), & \text{if } \theta(k,\ell,N) \neq \bot, \\ d\theta(k,\ell,P), & \text{if } \theta(k,\ell,P) \neq \bot, \\ \bot, & \text{if } \theta(k,\ell,N) = \theta(k,\ell,P) = \bot. \end{cases}$
3. *Abstractions* $(\lambda x_i\ N)$, where the free occurrences of x_i in N are $x_i^{(1)}, \ldots, x_i^{(m)}$ as N is scanned from left to right, for some $m \geqslant 0$:
   - $\Gamma(\lambda x_i\ N) = \Gamma(N)$.
   - $\tau(\lambda x_i\ N) = \begin{cases} c_1(-\alpha_i^1) \wedge \cdots \wedge c_m(-\alpha_i^m) \to \tau(N), & \text{if } m \geqslant 1,\ c_j \equiv \theta(i,j,N),\ 1 \leqslant j \leqslant m, \\ (-\alpha_i) \to \tau(N), & \text{if } m = 0. \end{cases}$
   - $\theta(k,\ell,(\lambda x_i\ N)) = \begin{cases} \bot, & \text{if } k = i, \\ \theta(k,\ell,N), & \text{if } k \neq i. \end{cases}$

The process of going from M to $\Gamma(M)$ as defined above is not strictly speaking a function, for two reasons. First, subterm occurrences in M are not specified to be generated in a fixed predetermined order. But this only affects the order in which auxiliary type variables from TVar_{aux} and $\wedge$-variables from $\wedge\text{-Var}_1$ are introduced into $\Gamma(M)$ in part 2 of the procedure, which turns out to be a convenience for us. That is, in some of the proofs later, it is notationally convenient *not* to prescribe a particular order in which members of TVar_{aux} and $\wedge\text{-Var}_1$ are introduced.

Second, occurrence numbers for the same λ-variable in M are not assigned in a unique way, e.g. they are not necessarily assigned consecutively from left to right, starting with 1. Again, this turns out to be notationally convenient.

Lemma 3.4 *For every well-named $M \in \Lambda$, the set $\Gamma(M)$ is a well-behaved set of constraints.*

Proof: The proof of Lemma 5.1 establishes this fact, and more. ■

Example 3.5 Let $M \equiv (\lambda f.\lambda x.f^{(1)}\ (f^{(2)}\ (f^{(3)}x)))\ (\lambda g.\lambda y.g^{(1)}\ (g^{(2)}y))$. (Instead of the formal λ-variables x_1, x_2, x_3, and x_4, we use x, y, f, and g for better readability.) $\Gamma(M)$ contains 6 constraints:

$$\begin{array}{lrcl}
(1) & d_3d_2(+\alpha_f^3) & = & d_3d_2(d_1(+\alpha_x) \to (-\beta_1)) \\
(2) & d_3(+\alpha_f^2) & = & d_3(d_2(+\beta_1) \to (-\beta_2)) \\
(3) & (+\alpha_f^1) & = & d_3(+\beta_2) \to (-\beta_3) \\
(4) & d_6d_5(+\alpha_g^2) & = & d_6d_5(d_4(+\alpha_y) \to (-\beta_4)) \\
(5) & d_6(+\alpha_g^1) & = & d_6(d_5(+\beta_4) \to (-\beta_5)) \\
(6) & (-\alpha_f^1) \wedge d_3(-\alpha_f^2) & \wedge & d_3d_2(-\alpha_f^3) \to (d_3d_2d_1(-\alpha_x) \to (+\beta_3)) \\
 & & = & d_6((-\alpha_g^1) \wedge d_5(-\alpha_g^2) \to (d_5d_4(-\alpha_y) \to (+\beta_5))) \to (-\beta_6)
\end{array}$$

A solution (not unique) for $\Gamma(M)$ is given by a pair (φ, ψ) where (for convenience we write d_{111} instead of $d_{1,1,1}$, d_{112} instead of $d_{1,1,2}$, etc.):

- $\varphi(d) = 1$ for all $d \in \wedge\text{-Var} - \{d_{111}, d_{112}, d_{121}, d_{122}, d_{21}, d_{22}, d_3, d_6\}$, and
- $\varphi(d_{111}) = \varphi(d_{112}) = \varphi(d_{121}) = \varphi(d_{122}) = \varphi(d_{21}) = \varphi(d_{22}) = \varphi(d_3) = 2$ and $\varphi(d_6) = 7$.

We omit the details of ψ, easily obtained by inspecting the constraints in $\varphi(\Gamma(M))$.

Example 3.6 Let $N \equiv (\lambda x.x^{(1)}x^{(2)})\ (\lambda y.y^{(1)}y^{(2)})$. $\Gamma(N)$ is the following set of constraints (polarities omitted in this example):

$$\begin{array}{lrcl}
(1) & \alpha^1_x & = & d_1\alpha^2_x \to \beta_1 \\
(2) & d_3\alpha^1_y & = & d_3(d_2\alpha^2_y \to \beta_2) \\
(3) & \alpha^1_x \wedge d_1\alpha^2_x \to \beta_1 & = & d_3(\alpha^1_y \wedge d_2\alpha^2_y \to \beta_2) \to \beta_3
\end{array}$$

$\Gamma(M)$ does not have a solution, corresponding to the fact that M is not typable in $\boldsymbol{\lambda}^{\to,\wedge}$, by the next theorem.

System $\boldsymbol{\lambda}^{\to,\wedge}$ is our lean version of the system of intersection types. (Similar but not quite identical restrictions of the system of intersection types are extensively studied in [8] and [9].) In the definition of $\boldsymbol{\lambda}^{\to,\wedge}$ below, A is a *type assignment*, i.e. a partial function from λ-Var to $\mathbb{T}$ with finite domain of definition, written as a finite list of pairs. If A and B are type assignements, then $A \wedge B$ is a new type assignment given by:

$$(A \wedge B)(x) = \begin{cases} \bot, & \text{if } A(x) = B(x) = \bot \ , \\ A(x), & \text{if } A(x) \neq \bot \text{ and } B(x) = \bot \ , \\ B(x), & \text{if } A(x) = \bot \text{ and } B(x) \neq \bot \ , \\ A(x) \wedge B(x), & \text{if } A(x) \neq \bot \text{ and } B(x) \neq \bot \ . \end{cases}$$

We take $\wedge$ non-commutative and non-idempotent (Conventions 2.3). Suppose there is a proof in $\boldsymbol{\lambda}^{\to,\wedge}$ for the sequent $A \vdash M : \tau$, where M is well-named, and x is a λ-variable occurring free in M. If there are $n \geqslant 1$ invocations of rule VAR in this proof to derive n types for x, then

$$A(x) = \sigma_1 \wedge \sigma_2 \wedge \cdots \wedge \sigma_n$$

where $\sigma_i \in \mathbb{T}^{\to}$ for $i = 1, \ldots, n$. If m is the number of of occurrences of x in M, then $n \geqslant m$.

System $\boldsymbol{\lambda}^{\to,\wedge}$

VAR $\quad x : \tau \ \vdash \ x : \tau \qquad \tau \in \mathbb{T}^{\to}$

ABS-I $$\frac{A,\ x : \sigma_1 \wedge \cdots \wedge \sigma_n \ \vdash \ M : \tau \qquad n \geqslant 1}{A \ \vdash \ (\lambda x.M) : (\sigma_1 \wedge \cdots \wedge \sigma_n \to \tau)}$$

ABS-K $$\frac{A \ \vdash \ M : \tau \qquad \sigma \in \mathbb{T}^{\to}}{A \ \vdash \ (\lambda x.M) : (\sigma \to \tau)}$$

APP $$\frac{A \ \vdash \ M : (\sigma_1 \wedge \cdots \wedge \sigma_n \to \tau) \qquad B_1 \ \vdash \ N : \sigma_1\ ,\ \ldots\ ,\ B_n \ \vdash \ N : \sigma_n \qquad n \geqslant 1}{A \wedge B_1 \wedge \cdots \wedge B_n \ \vdash \ (MN) : \tau}$$

Theorem 3.7 *For every well-named $M \in \Lambda$, M is typable in $\boldsymbol{\lambda}^{\rightarrow,\wedge}$ iff $\Gamma(M)$ has a solution.*

Proof: By induction on M. Details omitted in this preliminary draft. ∎

The next result is our promised characterization of β-SN via the unification problem $\wedge$**UP**. Another proof of this result is given in Section 7 (Corollary 7.14).

Corollary 3.8 *For every well-named $M \in \Lambda$, M is β-SN iff $\Gamma(M)$ has a solution.*

Proof: Immediate from Theorem 3.7, using the fact that M is β-SN iff M is typable in $\boldsymbol{\lambda}^{\rightarrow,\wedge}$, proved in [5].[1] ∎

Corollary 3.9 *There is no algorithm which, given an arbitrary well-named $M \in \Lambda$, can decide whether the set of constraints $\Gamma(M)$ has a solution.*

Proof: Immediate from Corollary 3.8, using the fact that it is undecidable whether an arbitrary λ-term is β-SN. ∎

Theorem 3.10 *There is a semi-decision procedure which, given an arbitrary well-named $M \in \Lambda$, terminates iff the set of constraints $\Gamma(M)$ has a solution. Moreover, if and when the procedure terminates, it returns a solution (φ, ψ) for $\Gamma(M)$.*

Proof: We can effectively generate all valuations $\varphi : \wedge\text{-Var} \rightarrow \mathbb{P}$ such that $\varphi(d) = 1$ for almost all $d \in \wedge\text{-Var}$, and all valuations $\psi : \text{TVar} \rightarrow \mathbb{T}^{\rightarrow}$ such that $\psi(\alpha) = \square$ for almost all $\alpha \in \text{TVar}$. We systematically generate all such pairs (φ, ψ), and we stop the procedure if and when we find one which is a solution for $\Gamma(M)$. ∎

4 Transformation of Constraint Sets

We devise another semi-decision procedure to test whether $\Gamma(M)$ has a solution. The new procedure (Corollary 7.17) does not return a solution (φ, ψ) for $\Gamma(M)$ if and when it terminates, in contrast to the procedure of Theorem 3.10. It is possible to adjust the new procedure in such a way that it returns a solution (φ, ψ) for $\Gamma(M)$ if and when it terminates, but at the cost of introducing unnecessary complications.

Definition 4.1 (special type schemes) The set of *special type schemes* is partitioned into two disjoint sets, $\mathcal{R}$ and $\mathcal{S}$, which are simultaneously defined by induction. We define them here with polarities "+" and "−" inserted:

[1] That M is β-SN iff M is typable in $\boldsymbol{\lambda}^{\rightarrow,\wedge}$ is proved once more, in an altogether different way, in Section 7 (Corollary 7.15).

1. $(+\square) \in \mathcal{R}^{\rightarrow}$ and $(-\square) \in \mathcal{S}^{\rightarrow}$.
   If $\alpha \in \mathrm{TVar}$ then $(+\alpha) \in \mathcal{R}^{\rightarrow}$ and $(-\alpha) \in \mathcal{S}^{\rightarrow}$.

2. If $\sigma \in \mathcal{R}^{\circ}$ and $\tau \in \mathcal{S}^{\rightarrow}$ then $(\sigma \rightarrow \tau) \in \mathcal{S}^{\rightarrow}$.
   If $\sigma \in \mathcal{S}^{\rightarrow} \cup \mathcal{S}^{\wedge} \cup \mathcal{S}^{\bullet}$ and $\tau \in \mathcal{R}^{\rightarrow}$ then $(\sigma \rightarrow \tau) \in \mathcal{R}^{\rightarrow}$.

3. If $\sigma_1, \ldots, \sigma_n \in \mathcal{R}^{\rightarrow} \cup \mathcal{R}^{\bullet}$ and $n \geqslant 2$ then $(\sigma_1 \wedge \cdots \wedge \sigma_n) \in \mathcal{R}^{\wedge}$.
   If $\sigma_1, \ldots, \sigma_n \in \mathcal{S}^{\rightarrow} \cup \mathcal{S}^{\bullet}$ and $n \geqslant 2$ then $(\sigma_1 \wedge \cdots \wedge \sigma_n) \in \mathcal{S}^{\wedge}$.

4. If $\sigma \in \mathcal{R}^{\rightarrow}$ and $d \in \wedge\text{-Var}$ then $d\sigma \in \mathcal{R}^{\circ}$.
   If $\sigma \in \mathcal{R}^{\rightarrow} \cup \mathcal{R}^{\bullet}$ and $d \in \wedge\text{-Var}$ then $d\sigma \in \mathcal{R}^{\bullet}$.
   If $\sigma \in \mathcal{S}^{\rightarrow} \cup \mathcal{S}^{\bullet}$ and $d \in \wedge\text{-Var}$ then $d\sigma \in \mathcal{S}^{\bullet}$.

Note that $\mathcal{R}^{\circ} \subset \mathcal{R}^{\bullet}$. The two sets of special type schemes are:

$$\mathcal{R} = \mathcal{R}^{\rightarrow} \cup \mathcal{R}^{\wedge} \cup \mathcal{R}^{\bullet} \qquad \text{and} \qquad \mathcal{S} = \mathcal{S}^{\rightarrow} \cup \mathcal{S}^{\wedge} \cup \mathcal{S}^{\bullet}$$

We call $\mathcal{R}$ (resp. $\mathcal{S}$) the set of *positive* (resp. *negative*) special type schemes. With polarities omitted, $\mathcal{R} \cup \mathcal{S}$ is a proper subset of $\mathcal{T}$. We say that type variable α occurs *positively* (resp. *negatively*) in $\sigma \in \mathcal{R} \cup \mathcal{S}$ if α occurs as $+\alpha$ (resp. $-\alpha$) in σ. We denote by $\pm\mathrm{TVar}(\sigma)$ the set of type variables occurring in σ with polarities inserted, and denote by $\mathrm{TVar}(\sigma)$ the same set with polarities omitted.

Although $\wedge$-variables are not preceded by a "+" or "−", we identify each with a polarity. We say that $d \in \wedge\text{-Var}$ occurs *positively* (resp. *negatively*) in $\sigma \in \mathcal{R} \cup \mathcal{S}$ if there is $\tau \in \mathcal{R}$ (resp. $\tau \in \mathcal{S}$) such that $d\tau$ occurs in σ.

The new semi-decision procedure is the result of repeatedly applying transformation rules to the set of constraints $\Gamma(M)$, for a given well-named $M \in \Lambda$.

Definition 4.2 (local transformation rules) The local transformation rules are: $\rightarrow$PARSE and $\wedge$PARSE. They are local because they work on one constraint at a time, without affecting other constraints in a simultaneous set of constraints Δ.

$\rightarrow$PARSE
$$\frac{\Delta \ \cup \ \{ \ c(\sigma_1 \rightarrow \sigma_2) \ = \ c(\tau_1 \rightarrow \tau_2) \ \}}{\Delta \ \cup \ \{ \ c\tau_1 \ = \ c\sigma_1 \ , \ c\sigma_2 \ = \ c\tau_2 \ \}}$$

where $c \in \mathcal{C}$, $\sigma_1 \in \mathcal{S}$, $\sigma_2 \in \mathcal{R}^{\rightarrow}$, $\tau_1 \in \mathcal{R}^{\circ}$, and $\tau_2 \in \mathcal{S}^{\rightarrow}$.

∧PARSE $$\frac{\Delta \ \cup \ \{\ c_1\sigma_1 \wedge \cdots \wedge c_n\sigma_n \ = \ c_1\tau_1 \wedge \cdots \wedge c_n\tau_n \ \}}{\Delta \ \cup \ \{\ c_1\sigma_1 \ = \ c_1\tau_1 \ , \ \ldots \ , \ c_n\sigma_n \ = \ c_1\tau_n \ \}}$$

where $c_1, \ldots, c_n \in \mathcal{C}$, $n \geqslant 2$, $\sigma_1, \ldots, \sigma_n \in \mathcal{R}^{\rightarrow}$, and $\tau_1, \ldots, \tau_n \in \mathcal{S}^{\rightarrow}$. A totally equivalent but more explicit way of writing →PARSE is this:

→PARSE $$\frac{\Delta \ \cup \ \{\ c(\sigma_{1,1} \wedge \cdots \wedge \sigma_{1,n} \rightarrow \sigma_2) \ = \ c(\tau_1 \rightarrow \tau_2) \ \}}{\Delta \ \cup \ \{\ c\tau_1 \ = \ c\sigma_{1,1} \wedge \cdots \wedge c\sigma_{1,n} \ , \ c\sigma_2 \ = \ c\tau_2 \ \}}$$

where $c \in \mathcal{C}$, $n \geqslant 1$, $\sigma_{1,1}, \ldots, \sigma_{1,n} \in \mathcal{S}^{\rightarrow} \cup \mathcal{S}^{\bullet}$, $\sigma_2 \in \mathcal{R}^{\rightarrow}$, $\tau_1 \in \mathcal{R}^{\circ}$, and $\tau_2 \in \mathcal{S}^{\rightarrow}$. The equivalence between the two different ways of writing →PARSE follows from the fact that ∧-variables distribute over ∧ (see Conventions 2.3).

Definition 4.3 (global transformation rules) The global transformation rules are: SUBST (in two versions), CLEAN, ALPHA and XPAND. They are global because they affect more than one constraint at a time. In contrast to the local rules, they do not increase the number of constraints.

If Δ is a set of constraints, $\alpha \in \text{TVar}$ and $\sigma \in \mathcal{R}^{\rightarrow}$, we write $\Delta[+\alpha := \sigma]$ to denote the set of constraints obtained by replacing every positive occurrence $+\alpha$ in Δ, if any, by σ. Similarly, we define $\Delta[-\alpha := \tau]$ where now $\tau \in \mathcal{S}^{\rightarrow}$.

+SUBST $$\frac{\Delta \ \cup \ \{\ c(+\alpha) \ = \ c\tau \ \}}{\Delta[-\alpha := \tau]}$$

−SUBST $$\frac{\Delta \ \cup \ \{\ c\sigma \ = \ c(-\alpha) \ \}}{\Delta[+\alpha := \sigma]}$$

where $c \in \mathcal{C}$, $\alpha \in \text{TVar}$, $\sigma \in \mathcal{R}^{\rightarrow}$, and $\tau \in \mathcal{S}^{\rightarrow}$. A use of +SUBST or −SUBST decreases the number of constraints by one. A particular case of +SUBST is when α occurs positively but not negatively in Δ; this particular case is identified as +SUBST$_1$.

+SUBST$_1$ $$\frac{\Delta \ \cup \ \{\ c(+\alpha) \ = \ c\tau \ \}}{\Delta} \qquad -\alpha \notin \pm\text{TVar}(\Delta)$$

We can similarly define −SUBST$_1$, the particular case of −SUBST when α occurs negatively but not positively in Δ. If $d \in \wedge$-Var, then $d\Delta$ is the constraint set:

$$d\Delta = \ \{d\sigma = d\tau \mid \sigma = \tau \text{ is a constraint in } \Delta\}$$

Another global transformation rule is:

CLEAN $$\frac{d\Delta \ \cup \ \Delta'}{\Delta \ \cup \ \Delta'} \qquad d \notin \wedge\text{-Var}(\Delta')$$

Rules $+$SUBST$_1$, $-$SUBST$_1$ and CLEAN are used to "clean up" constraint sets.

Recall the notion of renaming function (Definition 2.7), which is simultaneously an injection from TVar to TVar and injection from $\wedge$-Var to $\wedge$-Var. The rule ALPHA renames variables in a constraint set "without changing its meaning".

ALPHA $$\frac{\Delta}{f(\Delta)} \qquad f \text{ is a renaming function}$$

We need ALPHA in order to achieve a good fit between β-reduction and unification. It plays a role on the side of unification equivalent to α-conversion, which is implicit in β-reduction. (More on this at the beginning of Section 7.)

We need one more global transformation rule, for which the notation is a bit more complicated. Let Δ be a set of constraints, $d \in \wedge$-Var and $c_1, \dots, c_n \in \mathcal{C}$, for some $n \geqslant 1$. We write

$$\Delta[d := c_1 \wedge \cdots \wedge c_n]$$

to denote the set of constraints obtained by replacing every type scheme occurrence in Δ of the form $d\rho$ for some $\rho \in \mathcal{R}^{\rightarrow} \cup \mathcal{R}^{\bullet} \cup \mathcal{S}^{\rightarrow} \cup \mathcal{S}^{\bullet}$ by[2]

$$c_1\langle\rho\rangle_1 \wedge \cdots \wedge c_n\langle\rho\rangle_n$$

(ρ is a *type scheme occurrence* in Δ if there is a constraint $\sigma = \tau$ in Δ and ρ occurs in σ or τ.) In words, we create n distinct copies of ρ each with a new private set of type variables and $\wedge$-variables.

XPAND $$\frac{\text{“}cd\sigma \ = \ cc_1\tau_1 \wedge \cdots \wedge cc_n\tau_n\text{” is a constraint in } \Delta}{\Delta[d := c_1 \wedge \cdots \wedge c_n]}$$

where $c, c_1, \dots, c_n \in \mathcal{C}$, $n \geqslant 1$, $d \in \wedge$-Var, and $\sigma \in \mathcal{R}^{\rightarrow}$, $\tau_1, \dots, \tau_n \in \mathcal{S}^{\rightarrow}$.

A particular case of rule XPAND is when $n = 1$, i.e. the right-hand side of the constraint in the premise of XPAND can be just $cc_1\tau_1$. In this case, in order to minimize the renaming of type and $\wedge$-variables, we can take $\Delta[d := c_1]$ to mean: Replace every type scheme occurrence in Δ of the form $d\rho$ by $c_1\rho$ (not by $c_1\langle\rho\rangle_1$).

[2]No type scheme $d\rho$ for some $\rho \in \mathcal{R}^{\wedge} \cup \mathcal{S}^{\wedge}$ will occur in Δ. See 3 in Conventions 2.3.

Remark 4.4 In the case $n \geqslant 2$ it is tempting to redefine the expansion $[d := c_1 \wedge \cdots \wedge c_n]$ to mean: Replace every type scheme occurrence in Δ of the form $d\rho$ by $c_1\rho \wedge c_2\langle\rho\rangle_1 \wedge \cdots \wedge c_n\langle\rho\rangle_{n-1}$. But this redefinition would violate conditions 3 and 4 in Definition 2.5.

In Section 5 we prove the soundness of the transformation rules under appropriate restrictions. In the present context this means the following: If Δ and Δ' are constraint sets such that Δ' is obtained from Δ by one of the transformation rules, then Δ has a solution iff Δ' has a solution. In Section 7 we strengthen this result: If constraint set Δ has a solution, then repeated use of the transformation rules on Δ produces the empty constraint set $\varnothing$, which always has a solution. The examples below illustrate some of the issues we have to deal with.

Remark 4.5 Keep in mind that, in order to use the transformation rules on a constraint set Δ, polarities must be inserted. On the other hand, the question of whether Δ has (or does not have) a solution does not depend on the presence of polarities.

Example 4.6 Consider the term M of Example 3.5 and the corresponding set of constraints $\Delta_0 = \Gamma(M) = \{(1), \ldots, (6)\}$. We can transform constraint (6), using $\to$PARSE, to obtain two new constraints:

$$(7) \qquad d_6((-\alpha_g^1) \wedge d_5(-\alpha_g^2) \;\to\; d_5d_4(-\alpha_y) \to (+\beta_5)) \;=\; (-\alpha_f^1) \wedge d_3(-\alpha_f^2) \wedge d_3d_2(-\alpha_f^3)$$

$$(8) \qquad d_3d_2d_1(-\alpha_x) \;\to\; (+\beta_3) \;=\; (-\beta_6)$$

The resulting constraint set is now $\Delta_1 = \{(1), (2), (3), (4), (5), (7), (8)\}$. Using XPAND relative to constraint (7), we obtain another constraint set:

$$\Delta_2 \;=\; \Delta_1[d_6 := \varepsilon \wedge d_3 \wedge d_3d_2] \;=\; \{(1), (2), (3), (8), (9), (10), (11)\}$$

The constraints containing d_6, namely (4), (5) and (7), are transformed into (9), (10), and (11), respectively (for convenience we write d_{51} instead of $d_{5,1}$, d_{52} instead of $d_{5,2}$, etc.):

$$(9) \qquad d_{51}(+\alpha_{g1}^2) \wedge d_3d_{52}(+\alpha_{g2}^2) \wedge d_3d_2d_{53}(+\alpha_{g3}^2) \;=$$

$$d_{51}(d_{41}(+\alpha_{y1}) \to (-\beta_{41})) \wedge d_3d_{52}(d_{42}(+\alpha_{y2}) \to (-\beta_{42})) \wedge d_3d_2d_{53}(d_{43}(+\alpha_{y3}) \to (-\beta_{43}))$$

$$(10) \qquad (+\alpha_{g1}^1) \wedge d_3(+\alpha_{g2}^1) \wedge d_3d_2(+\alpha_{g3}^1) \;=$$

$$(d_{51}(+\beta_{41}) \to (-\beta_{51})) \wedge\; d_3(d_{52}(+\beta_{42}) \to (-\beta_{52})) \wedge\; d_3d_2(d_{53}(+\beta_{43}) \to (-\beta_{53}))$$

$$(11) \qquad ((-\alpha_{g1}^1) \wedge d_{51}(-\alpha_{g1}^2) \;\to\; d_{51}d_{41}(-\alpha_{y1}) \to (+\beta_{51}))\wedge$$

$$d_3((-\alpha_{g1}^1) \wedge d_{51}(-\alpha_{g1}^2) \;\to\; d_{51}d_{41}(-\alpha_{y1}) \to (+\beta_{51}))\wedge$$

$$d_3d_2((-\alpha_{g1}^1) \wedge d_{51}(-\alpha_{g1}^2) \;\to\; d_{51}d_{41}(-\alpha_{y1}) \to (+\beta_{51})) \;=\; (-\alpha_f^1) \wedge d_3(-\alpha_f^2) \wedge d_3d_2(-\alpha_f^3)$$

At this point we can use $\wedge$PARSE relative to (9), $\wedge$PARSE relative to (10), and $\wedge$PARSE relative to (11). In each case, a single constraint is replaced by three constraints. We stop the transformation process here, as it takes more than 50 steps to terminate. The results in Section 7 show that transforming the initial Δ_0 repeatedly is bound to terminate with the empty constraint set $\varnothing$, which trivially has a solution. On the other hand, using repeatedly the transformation rules on $\Gamma(N)$ in Example 3.6, the process does not terminate, because N is not β-SN, again by the results in Section 7.

Example 4.7 Consider the constraint set Δ_0:

$$\Delta_0 \ = \ \{ \ d(+\alpha_1) = (-\alpha_2) \wedge (-\alpha_3), \ d(+\alpha_4) = (+\alpha_5) \to (-\alpha_6) \ \}$$

Δ_0 does not have a solution. This fact can be discovered by using the transformation rules. (The example is easy enough so that its non-solvability can be also established by inspection.) The only rule we can initially use on Δ_0 is XPAND. If we use it, we obtain:

$$\begin{aligned} \Delta_1 \ &= \ \Delta_0[d := \varepsilon \wedge \varepsilon] \\ &= \ \{ \ (+\alpha_{11}) \wedge (+\alpha_{12}) = (-\alpha_2) \wedge (-\alpha_3), \ (+\alpha_{41}) \wedge (+\alpha_{42}) = (+\alpha_5) \to (-\alpha_6) \ \} \end{aligned}$$

Using $\wedge$PARSE relative to the first constraint in Δ_1, followed by two uses of +SUBST, we obtain:

$$\Delta_2 \ = \ \{ \ (+\alpha_{41}) \wedge (+\alpha_{42}) = (+\alpha_5) \to (-\alpha_6) \ \}$$

which cannot be transformed further and does not have a solution. On the other hand, if we allow the range of ψ to be all of $\mathbb{T}$, then it is easy to see that Δ_0 has a solution (φ, ψ) where $\varphi(d) = 1$ for all $d \in \wedge$-Var. Hence, *if we do not restrict the range of ψ to the proper subset $\mathbb{T}^{\to}$, our transformation rules do not preserve the solvability of constraint sets.*

Example 4.8 Consider the constraint set Δ_0:

$$\Delta_0 \ = \ \{ \ +\alpha_1 = (-\alpha_2) \wedge (-\alpha_3), \ +\alpha_1 = (+\alpha_4) \to (-\alpha_5) \ \}$$

Δ_0 does not have a solution. Nevertheless, using +SUBST twice on Δ_0, we obtain the empty constraint set $\varnothing$, which has a solution. The anomaly illustrated by this example results from the fact that the initial Δ_0 is not the constraint set of any $M \in \Lambda$, i.e. $\Delta_0 \neq \Gamma(M)$ for every $M \in \Lambda$. Hence, +SUBST *does not preserve the non-solvability of constraint sets that do not correspond to λ-terms.*

5 Invariant Properties of Transformation Rules

The notion of *well-behaved* type scheme (Definition 2.5) is still meaningful in the presence of polarities: If $\sigma \in \mathcal{R} \cup \mathcal{S}$, a type scheme with polarities inserted, we say that σ is well-behaved in case σ is well-behaved after all the polarities are omitted. More generally, every notion defined independently of polarities is still meaningful in their presence. We now list several properties that a constraint set Δ can satisfy.

(A) *Every type variable α occurs at most twice in Δ. And if α occurs twice, it occurs once positively as $+\alpha$ and once negatively as $-\alpha$.*

(B) *Every constraint in Δ is one of two forms:*

(B.1) $c_1\sigma_1 \wedge \cdots \wedge c_n\sigma_n \;=\; c_1\tau_1 \wedge \cdots \wedge c_n\tau_n$

(B.2) $cd\sigma \;=\; cc_1\tau_1 \wedge \cdots \wedge cc_n\tau_n$

where $n \geqslant 1$, $c, c_1, \ldots, c_n \in \mathcal{C}$, $d \in \wedge$-Var, d does not occur in $c_1 \wedge \cdots \wedge c_n$, $\sigma, \sigma_1, \ldots, \sigma_n \in \mathcal{R}^{\rightarrow}$, and $\tau_1, \ldots, \tau_n \in \mathcal{S}^{\rightarrow}$.

The constraints in the premises of rules $\rightarrow$PARSE, $\wedge$PARSE, $+$SUBST, and $-$SUBST, are all cases of (B.1). The constraint in the premise of rule XPAND is a case of (B.2).

Consider a constraint of the form (B.1), resp. (B.2). Let $d' \in \wedge$-Var. We say that d' has an *inner occurrence* in the constraint if d' occurs in $\{\sigma_1, \ldots, \sigma_n, \tau_1, \ldots, \tau_n\}$, resp. if d' occurs in $\{d\sigma, c_1\tau_1, \ldots, c_n\tau_n\}$. We say that d' has an *outer occurrence* in the constraint if d' occurs in $\{c_1, \ldots, c_n\}$, resp. if d' occurs in $\{c\}$. In words, an "outer occurrence" must appear symmetrically on both sides of the constraint and at the top level.

We say that $d' \in \wedge$-Var has an *inner*, resp. *outer*, occurrence in the set of constraints Δ if d' has an inner, resp. outer, occurrence in one of the constraints of Δ.

(C) *If $d \in \wedge$-Var occurs at all in Δ, then there is at most one occurrence of d in Δ which is both positive and inner.*

(D) *If "$\sigma = \tau$" is a constraint in Δ, then:*

(D.1) *No type variable $\alpha \in$ TVar has occurrences in both σ and τ.*

(D.2) *No $\wedge$-variable $d \in \wedge$-Var has inner occurrences in both σ and τ.*

Lemma 5.1 *If M is a well-named λ-term, then $\Gamma(M)$ is a well-behaved constraint set satisfying properties* { (A), (B), (C), (D) }.

Proof: Straightforward, if somewhat tedious induction on M. Details omitted in this preliminary draft. ∎

Let Δ and Δ' be constraint sets, and X one of the transformation rules:

$$X \in \{\rightarrow\text{PARSE}, \wedge\text{PARSE}, +\text{SUBST}, -\text{SUBST}, \text{CLEAN}, \text{ALPHA}, \text{XPAND}\}$$

We write $\Delta\ X\ \Delta'$ in case Δ' is obtained from Δ by using rule X. Note that if $\Delta\ X\ \Delta'$, then Δ' is not necessarily uniquely defined, as X may be used relative to different constraints in Δ. We generalize this notation to (finite) sequences of transformation rules. Let $\mathcal{X} = X_1 \cdots X_n$ where

$$X_1, \ldots, X_n \in \{\rightarrow\text{PARSE}, \wedge\text{PARSE}, +\text{SUBST}, -\text{SUBST}, \text{CLEAN}, \text{ALPHA}, \text{XPAND}\}$$

and $n \geqslant 0$. We define $\Delta\ \mathcal{X}\ \Delta'$ by:

$$\Delta\ \mathcal{X}\ \Delta' \quad \text{iff} \quad \Delta \equiv \Delta_1\ X_1\ \Delta_2\ X_2\ \Delta_3 \quad \cdots \quad X_n\ \Delta_{n+1} \equiv \Delta'$$

We generalize the notation further. Let $\{\mathcal{X}_1, \mathcal{X}_2, \ldots\}$ be a set of sequences of transformation rules:

$$\{\mathcal{X}_1, \mathcal{X}_2, \ldots\} \subseteq \{\rightarrow\text{PARSE}, \wedge\text{PARSE}, +\text{SUBST}, -\text{SUBST}, \text{CLEAN}, \text{ALPHA}, \text{XPAND}\}^*$$

We define $\Delta\ \{\mathcal{X}_1, \mathcal{X}_2, \ldots\}\ \Delta'$ by:

$$\Delta\ \{\mathcal{X}_1, \mathcal{X}_2, \ldots\}\ \Delta' \quad \text{iff} \quad \Delta\ \mathcal{X}_i\ \Delta' \quad \text{for some } i\ .$$

Lemma 5.2 *Let Δ and Δ' be constraint sets, and X a transformation rule in*

$$\{\rightarrow\text{PARSE}, \wedge\text{PARSE}, \text{CLEAN}, \text{ALPHA}\}$$

such that $\Delta\ X\ \Delta'$. If Δ is a well-behaved constraint set satisfying properties {(A), (B), (C), (D)}, *then so is Δ'.*

Proof: For each of the 4 rules under consideration, there are 5 parts to prove, namely that Δ' satisfies the 4 properties listed and that Δ' is well-behaved. This adds up to 20 separate cases. This is a straightforward (and tedious) case analysis. The only non-trivial case (perhaps) is to show that Δ' satisfies (B) when $X = \rightarrow$PARSE, in particular that "d does not occur in $c_1 \wedge \cdots \wedge c_n$" (see the formulation of (B)): For this, use the hypothesis that Δ satisfies not only (B) but also (D) (in fact (D.2) suffices). ∎

Lemma 5.3 *Let Δ and Δ' be constraint sets, and $X = \pm$SUBST such that $\Delta\ X\ \Delta'$. If Δ is a well-behaved constraint set satisfying properties* { (A), (B), (C) }, *then so is Δ'.*

Proof: It suffices to consider $X = +\text{SUBST}$, as the proof for $X = -\text{SUBST}$ is totally symmetric. If Δ is well-behaved, it is easy to check that Δ' is well-behaved, reviewing the 4 conditions in Definition 2.5. It is just as easy to check that if Δ satisfies property (A), then so does Δ', and likewise for properties (B) and (C). ∎

Lemma 5.4 *Let Δ and Δ' be constraint sets such that Δ XPAND Δ'. If Δ is a well-behaved constraint set satisfying properties* { (A), (B), (C) }, *then so is Δ'.*

Proof: Suppose $\Delta' = \Delta[d := c_1 \wedge \cdots \wedge c_n]$ and $n \geqslant 2$ throughout the proof. The case $n = 1$ is immediate, as no renaming of variables takes place, and is therefore omitted.

If Δ is well-behaved, then is Δ', by the 4 conditions in Definition 2.5 (straightforward details omitted). If Δ satisfies (A) and is well-behaved (only condition 3 of 2.5 matters here), then it is easy to see that Δ' satisfies (A) too. ∎

Lemma 5.5 *Let Δ and Δ' be well-behaved constraint sets, X one of the transformation rules, and $\Delta\ X\ \Delta'$.*

Let Δ be a constraint set and $\varphi : \wedge\text{-Var} \to \mathbb{P}$ such that $\varphi \models \Delta$. We write $|\text{TVar}(\varphi(\Delta))|$ to denote the number of type variables occurring in $\varphi(\Delta)$. We say φ is a *minimal solution* of Δ if

- $\varphi \models \Delta$, and
- for every $\varphi' \models \Delta$, we have $|\text{TVar}(\varphi(\Delta))| \leqslant |\text{TVar}(\varphi'(\Delta))|$.

Lemma 5.6 *Let Δ and Δ' be well-behaved constraint sets, X one of the transformation rules, and $\Delta\ X\ \Delta'$. If $\varphi, \varphi' : \wedge\text{-Var} \to \mathbb{P}$ are minimal solutions of Δ and Δ' respectively, then $|\text{TVar}(\varphi(\Delta))| \geqslant |\text{TVar}(\varphi'(\Delta'))|$. If in addition $X = \pm\text{SUBST}$ and Δ satisfies property* (A), *then $|\text{TVar}(\varphi(\Delta))| > |\text{TVar}(\varphi'(\Delta'))|$.*

For later reference, we state two easy facts about transformation rules.

Lemma 5.7 *Let Δ and Δ' be well-behaved constraint sets, and X one of the transformation rules.*

1. *If Δ CLEAN X Δ' then Δ X CLEAN Δ'.*
2. *If Δ ALPHA X Δ' then Δ X ALPHA Δ'.*

In words, we can delay uses of CLEAN *and* ALPHA *past other uses of transformation rules.*[3]

[3]The converse of part 1 is not true in general; there are easy counterexamples. The converse of part 2 is true, but we do not need it.

Proof: Straightforward, if somewhat tedious, case analysis. ■

Lemma 5.8 *Let* Δ *and* Δ' *be well-behaved constraint sets. If* Δ ALPHA ALPHA Δ'*, then* Δ ALPHA Δ'*. In words, consecutive uses of* ALPHA *can be combined into a single use of* ALPHA.

Proof: The composition of two renaming functions is a renaming function. ■

We conclude this section with another conjecture.

Conjecture 5.9 *Let* M *be a well-named* λ*-term. The following are equivalent conditions:*

1. $\Gamma(M)$ *has a solution.*

2. *Using the rules in* $\{\to\text{PARSE}, \wedge\text{PARSE}, +\text{SUBST}, -\text{SUBST}, \text{XPAND}\}$ *repeatedly, in any order, the constraint set* $\Gamma(M)$ *is always transformed into the empty constraint set* $\varnothing$.

In Section 7 we prove a result (Corollary 7.17) which is weaker than the preceding conjecture, but sufficient for our purposes, namely: "$\Gamma(M)$ has a solution" is equivalent to the transformation of $\Gamma(M)$ into $\varnothing$ using the rules repeatedly *in a particular order.* This is the particular order specified by the $\sharp$-transformation of $\Gamma(M)$ (Definition 7.16).

6 A Useful Generalization of Beta-Reduction

K-redexes are the source of many interesting complications in the λ-calculus. The particular complication concerning us here is the difference they introduce between β*-weak-normalization* (β-WN) and β*-strong-normalization* (β-SN). In the absence of K-redexes the two notions coincide. There is a long trail of results on how to reduce β-SN to β-WN without excluding K-redexes since the late 1960's, by Nederpelt, by Klop, and by many others in the 1980's and 1990's (see the references in [4] and [7] for example). We tackle this question once more, not to prove a result (Theorem 6.5) which is likely to be found in some form or other in the extensive literature, but to adapt it to our later needs (Section 7).

Every λ-term M which is not in β-nf contains a *leftmost* β-redex occurrence $R \equiv ((\lambda x.P)Q)$. R is uniquely identified by its λ-binding "λx" which occurs to the left of the λ-binding of every other, if any, β-redex occurrence in M.

Lemma 6.1 *Let* $R \equiv ((\lambda x.P)Q)$ *be a leftmost* β*-redex occurrence in* M*, and let* $M \xrightarrow[\beta]{R} N$.

1. *If* R *is a I-redex and* N *is* β*-SN, then* M *is* β*-SN.*

2. *If* R *is a K-redex and both* N *and* Q *are* β*-SN, then* M *is* β*-SN.*

Example 6.2 Part 2 of the preceding lemma is not true without the restriction "leftmost". Consider the term

$$M \equiv \ (\ \underbrace{(\lambda x.\ (\lambda v.\lambda w.\ vw))\ \mathbf{I}}_{R_1}\ )\ (\lambda y.\ \underbrace{(\lambda x.\mathbf{I})(y\boldsymbol{\omega}\boldsymbol{\omega})}_{R_2}\ )\ (\lambda v.\lambda w.\ vw)$$

where $\mathbf{I} \equiv (\lambda z.z)$ and $\boldsymbol{\omega} \equiv (\lambda z.zz)$. M contains two β-redex occurrences: R_1 and R_2. R_1 is leftmost-outermost, R_2 is only outermost, and both are K-redexes. (A β-redex occurrence R in M is *outermost* if R does not occur as a proper subterm in another β-redex occurrence in M. Leftmost is a special case of outermost.) β-reducing R_2, we get

$$N \equiv \ (\ (\lambda x.\ (\lambda v.\lambda w.\ vw))\ \mathbf{I}\ )\ (\lambda y.\ \mathbf{I})\ (\lambda v.\lambda w.\ vw)$$

It is not the case that M is β-SN (it is not) if N and $(y\boldsymbol{\omega}\boldsymbol{\omega})$ are β-SN (they both are). This example also shows that relaxing the "leftmost" restriction to "outermost" is not strong enough to get part 2 of Lemma 6.1.

$G_\beta(M)$ is the β-reduction graph of λ-term M (Section 3.1 in [1]). The set of vertices in $G_\beta(M)$ is $\{N \mid M \underset{\beta}{\twoheadrightarrow} N\}$ modulo α-equivalence, i.e. if $M \underset{\beta}{\twoheadrightarrow} N_1$ and $M \underset{\beta}{\twoheadrightarrow} N_2$, and $N_1 \equiv_\alpha N_2$, then N_1 and N_2 refer to the same vertex. There is an edge from vertex N_1 to vertex N_2 in $G_\beta(M)$ iff $N_1 \underset{\beta}{\longrightarrow} N_2$. $G_\beta(M)$ is a connected graph, because every vertex N is accessible from vertex M. Define

$$\mathit{degree}(M) \ = \ \text{“number of edges in } G_\beta(M)\text{”}$$

The relevant fact for us is: M is β-SN iff $G_\beta(M)$ is a finite dag (directed acyclic graph). In particular, if M is β-SN then *degree*(M) is finite (the converse is not true).

Lemma 6.3 *Let $R \equiv ((\lambda x.P)Q)$ be a leftmost β-redex occurrence in M, and let $M \xrightarrow[\beta]{R} N$.*

1. *If R is a I-redex and M is β-SN, then* $\mathrm{degree}(M) > \mathrm{degree}(N)$.
2. *If R is a K-redex and M is β-SN, then* $\mathrm{degree}(M) > \mathrm{degree}(N) + \mathrm{degree}(Q)$.[4]

Definition 6.4 ([β]-reduction) Let $\mathfrak{M}$ be the multiterm $[M_1, \ldots, M_\ell]$, i.e. a finite sequence of λ-terms (repetitions allowed), and $R \equiv ((\lambda x.P)Q)$. We write $\mathfrak{M} \xrightarrow[[\beta]]{R} \mathfrak{N}$ to mean two conditions are satisfied:

1. R is a leftmost β-redex occurrence in $\mathfrak{M}$, i.e. there is $k \in \{1, \ldots, \ell\}$ such that R is leftmost in M_k and $M_1, \ldots, M_{k-1}$ are all in β-nf.

[4]Lemma 6.3 is probably true without the restriction "leftmost" on R, but we do not need such a result.

2. If $M_k \xrightarrow[\beta]{R} N$, then $\mathfrak{N} = \begin{cases} [M_1, \ldots, M_{k-1}, N, M_{k+1}, \ldots, M_\ell], & \text{if } R \text{ is a I-redex,} \\ [M_1, \ldots, M_{k-1}, N, Q, M_{k+1}, \ldots, M_\ell], & \text{if } R \text{ is a K-redex.} \end{cases}$

We write $\mathfrak{M} \xrightarrow[{[\beta]}]{} \mathfrak{N}$, pronounced "multiterm $\mathfrak{M}$ beta-reduces to multiterm $\mathfrak{N}$", if there is a leftmost β-redex occurrence R in $\mathfrak{M}$ such that $\mathfrak{M} \xrightarrow[{[\beta]}]{R} \mathfrak{N}$.

Strictly speaking, the relation $[\beta]$ is not a "notion of reduction" in the sense of Section 3.1 in [1], because it relates two multiterms (rather than two terms). Nevertheless, $[\beta]$-reduction generalizes β-reduction not only in the sense that (1) it relates two multiterms rather than two terms, but also in the sense that (2) it does not discard arguments of K-redexes after their reduction.

Theorem 6.5 *For every $M \in \Lambda$, M is β-SN iff $[M]$ is $[\beta]$-normalizing.*

Proof: There are two inductions in this proof, and to push them through, prove a more general result, namely, for every multiterm $\mathfrak{M}$, the following are equivalent:

(a) Every $M \in \mathfrak{M}$ is β-SN.

(b) $\mathfrak{M}$ is $[\beta]$-SN.

(c) $\mathfrak{M}$ is $[\beta]$-normalizing.

First prove (a) implies (b). Generalize the notion of β-reduction graph to every multiterm $\mathfrak{M}$, by defining $G_\beta(\mathfrak{M})$ as $\biguplus\{G_\beta(M) \mid M \in \mathfrak{M}\}$ (this is multiset union). Unless $\mathfrak{M}$ contains only one term (or no terms at all), $G_\beta(\mathfrak{M})$ is a disconnected graph, with one component for every member of $\mathfrak{M}$ and with the same multiplicity. Define

$$\mathit{degree}(\mathfrak{M}) \;=\; \sum \{\ \mathit{degree}(M) \mid M \in \mathfrak{M}\ \}$$

(this counts $\mathit{degree}(M)$ as many times as there are copies of M in $\mathfrak{M}$). Now, every $M \in \mathfrak{M}$ is β-SN iff $G_\beta(\mathfrak{M})$ is a finite dag.

The proof that (a) implies (b) is by induction on $\mathit{degree}(\mathfrak{M}) \geqslant 0$. If $\mathit{degree}(\mathfrak{M}) = 0$ then every $M \in \mathfrak{M}$ is in β-nf, so that $\mathfrak{M}$ is also $[\beta]$-SN. Assume the result true for every multiterm $\mathfrak{M}$ such that every $M \in \mathfrak{M}$ is β-SN and such that $\mathit{degree}(\mathfrak{M}) \leqslant n$. Consider a fixed, but otherwise arbitrary $\mathfrak{M}$, such that every $M \in \mathfrak{M}$ is β-SN and such that $\mathit{degree}(\mathfrak{M}) = n+1$. We want to show that every $[\beta]$-reduction sequence σ starting from $\mathfrak{M}$ terminates. Consider the first step of such a sequence σ, say $\mathfrak{M} \xrightarrow[{[\beta]}]{} \mathfrak{N}$. Reviewing Definition 6.4, it is easy to see that if every $M \in \mathfrak{M}$ is β-SN then so is every $N \in \mathfrak{N}$ and, by Lemma 6.3, that $\mathit{degree}(\mathfrak{N}) \leqslant n$. Hence, by the induction hypothesis, $\mathfrak{N}$ is $[\beta]$-SN, which in turn implies the sequence σ terminates.

The proof that (b) implies (c) is immediate.

The proof that (c) implies (a) is by induction on the length of $[\beta]$-normalizing sequences. Consider a $[\beta]$-normalizing sequence from a multiterm $\mathfrak{M}$:

$$\mathfrak{M}_0 = \mathfrak{M} \underset{[\beta]}{\longrightarrow} \mathfrak{M}_1 \underset{[\beta]}{\longrightarrow} \mathfrak{M}_2 \underset{[\beta]}{\longrightarrow} \cdots \underset{[\beta]}{\longrightarrow} \mathfrak{M}_n$$

where $\mathfrak{M}_n$ is in $[\beta]$-nf, so that every $M \in \mathfrak{M}_n$ is in β-nf. If $n = 0$, then $\mathfrak{M}_0 = \mathfrak{M}_n$ and the desired conclusion is immediate. Assume the result true for every $[\beta]$-normalizing sequence of length $n \in \mathbb{N}$, and prove it for an an arbitrary $[\beta]$-normalizing sequence of length $n+1$, using Lemma 6.1. ∎

7 Beta-Reduction as Unification

Let $M \in \Lambda$ and $R \equiv ((\lambda x.P)Q)$ a β-redex occurrence in M, i.e. $M \equiv C[R]$ where $C[\;]$ is a context with a single hole. In general, we need to α-convert R before β-reducing it, in order to avoid capture of free variable occurrences in Q by λ-bindings in P. The necessary α-conversion can in fact be restricted to P, so that if $M \xrightarrow[\beta]{R} N$ we can write:

$$N \;\equiv\; C[P'[x := Q]] \quad \text{where} \quad P' \;\equiv_\alpha\; P \;.$$

Neither C nor Q are α-converted. For the correspondence to be established in this section, we need to spell out exactly where α-conversion takes place.

If M is well-named to start with, no α-conversion is necessary at all to avoid capture of free variables. In this case, if we take $N \equiv C[P[x := Q]]$ (no α-conversion in C, P, or Q) then $M \xrightarrow[\beta]{R} N$ is a valid β-reduction. However, the resulting N is not necessarily well-named, and the problem of free-variable capture may be encountered later, if we β-reduce N again. Consider for example the well-named λ-term $M \equiv (\lambda x.xx)(\lambda y.\lambda z.yz)$. Without α-conversion:

$$M \underset{\beta}{\longrightarrow} N \equiv (\lambda y.\lambda z.yz)(\lambda y.\lambda z.yz) \underset{\beta}{\longrightarrow} N_1 \equiv \lambda z.(\lambda y.\lambda z.yz)z \underset{\beta}{\longrightarrow} N_2 \equiv \lambda z.\lambda z.zz$$

None of N, N_1 and N_2 is well-named and the reduction from N_1 to N_2 is not valid. The usual practice is to α-convert whenever necessary only, which we can call *lazy α-conversion.* We do not need to α-convert N (and therefore do not, according to lazy α-conversion), and only α-convert N_1 to, say, $N_1' \equiv \lambda z.(\lambda y.\lambda z'.yz')z$ before β-reducing it. The resulting reduction sequence is now valid:

$$M \underset{\beta}{\longrightarrow} N \equiv (\lambda y.\lambda z.yz)(\lambda y.\lambda z.yz) \underset{\beta}{\longrightarrow} N_1' \equiv \lambda z.(\lambda y.\lambda z'.yz')z \underset{\beta}{\longrightarrow} N_2' \equiv \lambda z'.\lambda z.zz'$$

Our practice here will be different, which we can call *eager α-conversion.* We prevent the problem at an earlier stage: Every β-reduct is α-converted to a well-named λ-term before it is β-reduced again, whether or not capture of free variables occurs. For the preceding example, we have the following reduction sequence according to this convention:

$$M \underset{\beta}{\longrightarrow} \widetilde{N} \equiv (\lambda y.\lambda z.yz)(\lambda y'.\lambda z'.y'z') \underset{\beta}{\longrightarrow} \widetilde{N}_1 \equiv \lambda z.(\lambda y'.\lambda z'.y'z')z \underset{\beta}{\longrightarrow} \widetilde{N}_2 \equiv \lambda z.\lambda z'.zz'$$

Each of $\widetilde{N}$, $\widetilde{N}_1$, and $\widetilde{N}_2$, is well-named. For our purposes, we need to restrict this convention further.

Convention 7.1 Let M be a well-named λ-term, $R \equiv ((\lambda x.P)Q)$ a β-redex occurrence in M, and $M \equiv C[R]$ where $C[\ ]$ is a context with a single hole. Write R as

$$R \equiv \ ((\lambda x.P[x^{(1)}, x^{(2)}, \ldots, x^{(n)}])Q)$$

explicitly listing the $n \geqslant 0$ bound occurrences of x. In this notation, P is a context with exactly n holes, which does not mention x anywhere else. We spell out conditions under which the β-reduction of R in M produces another well-named λ-term N. We write $M \xrightarrow[\beta]{R} N$ provided:[5]

a. Either $n \leqslant 1$ and $N \equiv C[P[Q]]$, with no α-conversion allowed in C, P, or Q.

b. Or $n \geqslant 2$ and $N \equiv C[P[Q_1, \ldots, Q_n]]$ where:

  - $Q \equiv_\alpha Q_1 \equiv_\alpha \cdots \equiv_\alpha Q_n$,
  - $Q_1 Q_2 \cdots Q_n$ is a well-named λ-term,
  - the bound λ-variables in $Q_1 Q_2 \cdots Q_n$ are fresh, i.e. do not occur in $C[P[\ , \ldots, \ ]]$.

These conditions guarantee both that $M \xrightarrow[\beta]{R} N$ is a valid β-reduction and that N is well-named. If $n \geqslant 2$, note that we restrict α-conversion to the argument Q: Neither C nor P are α-converted.[6]

Definition 7.2 ([β]-reduction revisited) We adjust [β]-reduction according to the preceding convention. The multiterm $\mathfrak{M} = [M_1, \ldots, M_\ell]$ is *well-named* if the single λ-term $xM_1 \cdots M_\ell$ is well-named, where x appears nowhere in $\mathfrak{M}$.[7] Let $R \equiv ((\lambda x.P)Q)$ be a β-redex occurrence in $\mathfrak{M}$, i.e. $M_k \equiv C[R]$ for some $k \in \{1, \ldots, \ell\}$ where $C[\ ]$ is a context with a single hole. Adopting the notation of Definition 6.4 and Convention 7.1, we write $\mathfrak{M} \xrightarrow[{[\beta]}]{R} \mathfrak{N}$ if the two conditions of 6.4 are satisfied in addition to:

[5] It is possible to merge the two cases, $n \leqslant 1$ and $n \geqslant 2$, by requiring that $N \equiv C[P[Q, Q_1, \ldots, Q_{n-1}]]$. But the resulting indexing would complicate some of the bookkeeping later (e.g. in the proof of Lemma 7.7), partly because it would conflict with the conventions for the XPAND rule (see Remark 4.4).

[6] So, in eager α-conversion, C and P are not α-converted; only if $n \geqslant 2$ do we α-convert Q, whether or not capture of free variables occurs. By contrast, in lazy α-conversion, C and Q are not α-converted; and only if capture of free variables occurs do we α-convert P.

[7] We can equivalently require that the single λ-term $M_1 \cdots M_\ell$ be well-named.

3. Conditions a and b of 7.1 are satisfied.

Moreover, if occurrence numbers are inserted (which we need when we use procedure Γ, Definition 7.3 below) we further stipulate:

4. The reduction $\mathfrak{M} \xrightarrow[{[\beta]}]{R} \mathfrak{N}$ does not change occurrence numbers in:
   - $M_1, \ldots, M_{k-1}, M_{k+1}, \ldots, M_\ell$.
   - $C[(\lambda x.P[\ ,\ldots,\ ])[\ ]]$, i.e. they remain the same in $C[P[\ ,\ldots,\ ]]$ after the reduction.
   - Q, if $n \leqslant 1$.

   $\mathfrak{M} \xrightarrow[{[\beta]}]{R} \mathfrak{N}$ introduces fresh occurrence numbers only in $Q_1, \ldots, Q_n$ and only if $n \geqslant 2$.

Names of type variables in the constraint set $\Gamma(\mathfrak{M})$ depend on names of λ-variables and their occurrence numbers in $\mathfrak{M}$. Conditions 3 and 4 above are imposed on the $[\beta]$-reduction $\mathfrak{M} \xrightarrow[{[\beta]}]{R} \mathfrak{N}$ in order to regulate and minimize the process of variable-renaming in going from $\Gamma(\mathfrak{M})$ to $\Gamma(\mathfrak{N})$.

Definition 7.3 (Γ revisited) We extend the procedure Γ (Definition 3.3) to well-named multiterms. If $\mathfrak{M} = [M_1, \ldots, M_\ell]$ is a well-named multiterm, we first assign a unique occurrence number to every λ-variable occurrence in $\mathfrak{M}$, free or bound (but not binding), and then define $\Gamma(\mathfrak{M})$ by:

$$\Gamma(\mathfrak{M}) \ = \ \Gamma(M_1) \ \cup \ \cdots \ \cup \ \Gamma(M_\ell)$$

$\Gamma(\mathfrak{M})$ induces 2 other constraint sets: $\Gamma_a(\mathfrak{M})$ and $\Gamma_b(\mathfrak{M})$. For the definition of $\Gamma_a(\mathfrak{M})$, recall the special case $+\text{SUBST}_1$ of $+\text{SUBST}$ (Definition 4.3): It deletes constraints of the form $c(+\alpha) = c\tau$. $\Gamma_a(\mathfrak{M})$ is the constraint set Δ_2 such that:

1. $\Delta_1 = \Gamma(\mathfrak{M})$.

2. $\Delta_1\ \{+\text{SUBST}_1, \text{CLEAN}\}^*\ \Delta_2$.

3. Δ_2 cannot be transformed further using $+\text{SUBST}_1$ or CLEAN,
   i.e. Δ_2 is in $\{+\text{SUBST}_1, \text{CLEAN}\}$-normal form.

In clause 2, we can write $\Delta_1\ \{+\text{SUBST}_1\}^*\{\text{CLEAN}\}^*\ \Delta_2$ instead, by Lemma 5.7 part 1.[8] It is easy to see that $\Gamma_a(\mathfrak{M})$ is uniquely defined, i.e. it is independent of the order in which we delete constraints in $\Gamma(\mathfrak{M})$ using $+\text{SUBST}_1$ or cross out outermost $\wedge$-variables using CLEAN. $\Gamma_b(\mathfrak{M})$ is a subset of $\Gamma_a(\mathfrak{M})$:

$$\Gamma_b(\mathfrak{M}) \ = \ \Gamma_a(\mathfrak{M}) \ \cap \ \{\ c(\sigma_1 \to \sigma_2) = c(\tau_1 \to \tau_2) \mid c \in \mathcal{C}, \sigma_1 \in \mathcal{S}, \sigma_2 \in \mathcal{R}^{\to}, \tau_1 \in \mathcal{R}^{\circ}, \tau_2 \in \mathcal{S}^{\to}\ \}$$

[8] $+\text{SUBST}_1$ and CLEAN do not commute in general.

In words, $\Gamma_b(\mathfrak{M})$ is the subset of $\Gamma_a(\mathfrak{M})$ consisting of all the constraints to which we can apply the transformation rule $\rightarrow$PARSE.

Lemma 7.4 *Let $\mathfrak{M}$ be a well-named multiterm. The number of β-redex occurrences in $\mathfrak{M}$ is precisely the number of constraints in $\Gamma_b(\mathfrak{M})$. In particular, $\mathfrak{M}$ is in $[\beta]$-nf iff $\Gamma_b(\mathfrak{M}) = \varnothing$.*

Proof: It suffices to prove it for the case of a single term M, i.e. when $\mathfrak{M} = [M]$. The proof is a straightforward induction on M. Note that because of the naming convention for well-named terms and well-named multiterms, every β-redex in $\mathfrak{M}$ occurs exactly once. ∎

Lemma 7.5 *If $\mathfrak{M}$ is a well-named multiterm, then $\mathfrak{M}$ is in $[\beta]$-nf if and only if $\Gamma_a(\mathfrak{M}) = \varnothing$.*

Definition 7.6 (sharp-reduction) This reduction relation is only defined between constraint sets corresponding to well-named multiterms. Given constraint sets Δ_1 and Δ_2, we write $\Delta_1 \underset{\sharp}{\longrightarrow} \Delta_2$ ("Δ_1 sharp-reduces to Δ_2") iff there are well-named multiterms $\mathfrak{M}_1$ and $\mathfrak{M}_2$ such that:

1. $\Delta_1 = \Gamma_a(\mathfrak{M}_1)$ and $\Delta_2 = \Gamma_a(\mathfrak{M}_2)$.
2. There is a β-redex occurrence R in $\mathfrak{M}_1$ such that $\mathfrak{M}_1 \xrightarrow[{[\beta]}]{R} \mathfrak{M}_2$. (By the definition of $[\beta]$-reduction, Definitions 6.4 and 7.2, R is necessarily leftmost in $\mathfrak{M}_1$.)

We say that Δ is in $\sharp$-nf iff there is a well-named multiterm $\mathfrak{M}$ such that $\Delta = \Gamma_a(\mathfrak{M})$ and $\mathfrak{M}$ is in $[\beta]$-nf. By Lemma 7.5, this means Δ is in $\sharp$-nf iff $\Delta = \varnothing$.

Lemma 7.7 *Let Δ_1 and Δ_2 be constraint sets such that $\Delta_1 \underset{\sharp}{\longrightarrow} \Delta_2$. This single $\sharp$-reduction step can be decomposed into a finite sequence of the transformation rules — in this order:*

- *one use of* $\rightarrow$PARSE,
- *one use of* XPAND,
- *zero or more uses of* $\wedge$PARSE,
- *one or more uses of* $-$SUBST,
- *one use of* ALPHA.

Lemma 7.7 gives a precise meaning to the title of this section and the entire report:

BETA-REDUCTION AS UNIFICATION

A single $[\beta]$-reduction step from $\mathfrak{M}_1$ to $\mathfrak{M}_2$ corresponds to a finite sequence of unification steps in

$$\{\to\text{PARSE}\}\{\text{XPAND}\}\{\wedge\text{PARSE}\}^*\{-\text{SUBST}\}^+$$

from Δ_1 to Δ_2. (We ignore the last use of ALPHA in Lemma 7.7, as it is only a renaming of variables.) This correspondence works just fine if $\Delta_1 = \Gamma_a(\mathfrak{M}_1)$ and $\Delta_2 = \Gamma_a(\mathfrak{M}_2)$, but not if $\Delta_1 = \Gamma(\mathfrak{M}_1)$ and $\Delta_2 = \Gamma(\mathfrak{M}_2)$. The next example illustrates the complication, had we taken instead $\Delta_1 = \Gamma(\mathfrak{M}_1)$ and $\Delta_2 = \Gamma(\mathfrak{M}_2)$. The source of the problem are the K-redexes, which were also the reason in Section 6 for the generalization of β-reduction to a relation between multiterms (rather than between terms only).

Example 7.8 Consider the λ-term $M \equiv v(\lambda w.(\lambda x.w^{(1)}y)w^{(2)})$. The leftmost (and only) β-redex in M is $R \equiv (\lambda x.w^{(1)}y)w^{(2)}$. If we set $\mathfrak{M}_1 \equiv [M]$ and $[\beta]$-reduce R, we obtain:

$$\mathfrak{M}_1 \equiv [M] \quad \xrightarrow[{[\beta]}]{R} \quad \mathfrak{M}_2 \equiv [\ v(\lambda w.w^{(1)}y)\ ,\ w^{(2)}\ ]$$

Using Γ instead of Γ_a, the corresponding constraint sets are:[9]

$$\begin{aligned}
\Delta_1 &= \Gamma(\mathfrak{M}_1)\\
&= \{\ d_3(+\alpha_w^1) = d_3(d_1(+\alpha_y) \to (-\beta_1)),\ d_3((-\alpha_x) \to (+\beta_1)) = d_3(d_2(+\alpha_w^2) \to (-\beta_2)),\\
&\qquad\qquad (+\alpha_v) = d_3((-\alpha_w^1) \wedge d_2(-\alpha_w^2) \to (+\beta_2)) \to (-\beta_3)\ \}\\
\Delta_2 &= \Gamma(\mathfrak{M}_2)\\
&= \{\ d_3(+\alpha_w^1) = d_3(d_1(+\alpha_y) \to (-\beta_1)),\ (+\alpha_v) = d_3((-\alpha_w^1) \to (+\beta_1)) \to (-\beta_3)\ \}
\end{aligned}$$

No matter how we use the transformation rules, it is not possible to transform Δ_1 into Δ_2. The reason is this: Δ_2 does not mention α_w^2, whereas Δ_1 does, in its second and third constraints. To reach Δ_2 from Δ_1, using the transformation rules, we have to eliminate α_w^2 somehow. We can eliminate α_w^2 by using $\to$PARSE relative to the second constraint in Δ_1, followed by XPAND to carry out the expansion $[d_2 := \varepsilon]$, followed by $-$SUBST twice to eliminate α_w^2 (and β_2) — but then there will remain an α_x which we cannot eliminate. This happens only because R is a K-redex. On the other hand, using Γ_a instead of Γ:

$$\widetilde{\Delta_1} \ = \ \Gamma_a(\mathfrak{M}_1) \ = \ \{\ (-\alpha_x) \to (+\beta_1) = d_2(+\alpha_w^2) \to (-\beta_2)\ \}$$

and $\widetilde{\Delta_2} = \Gamma_a(\mathfrak{M}_2) = \varnothing$. Now, $\widetilde{\Delta_1}$ can be transformed into $\widetilde{\Delta_2} = \varnothing$, by using $\to$PARSE first, followed by XPAND, followed by $-$SUBST twice.

[9]According to Definition 3.3 we have freedom in choosing the order in which $\wedge$-variables from $\wedge$-Var$_1$ and type variables from TVar$_{aux}$ are introduced by Γ.

If we replace y by x in M, we obtain $M' \equiv v(\lambda w.(\lambda x.w^{(1)}x)w^{(2)})$, where $R' \equiv (\lambda x.w^{(1)}x)w^{(2)}$ is now an I-redex. We then have

$$\mathfrak{M}'_1 \equiv [M'] \;\underset{[\beta]}{\xrightarrow{R'}}\; \mathfrak{M}'_2 \equiv [\; v(\lambda w.w^{(1)}w^{(2)}) \;]$$

and the corresponding constraint sets are — again using Γ not Γ_a:

$$\begin{aligned}
\Delta'_1 &= \Gamma(\mathfrak{M}'_1) \\
&= \{\; d_3(+\alpha^1_w) = d_3(d_1(+\alpha_x) \to (-\beta_1)),\; d_3(d_1(-\alpha_x) \to (+\beta_1)) = d_3(d_2(+\alpha^2_w) \to (-\beta_2)), \\
&\qquad\qquad (+\alpha_v) = d_3((-\alpha^1_w) \wedge d_2(-\alpha^2_w) \to (+\beta_2)) \to (-\beta_3) \;\} \\
\Delta'_2 &= \Gamma(\mathfrak{M}'_2) \\
&= \{\; d_3(+\alpha^1_w) = d_3(d_1(+\alpha^2_w) \to (-\beta_1)),\; (+\alpha_v) = d_3((-\alpha^1_w) \wedge d_1(-\alpha^2_w) \to (+\beta_1)) \to (-\beta_3) \;\}
\end{aligned}$$

If we use $\to$PARSE, followed by XPAND, followed by $-$SUBST twice, it is easy to check that Δ'_1 can be transformed into Δ'_2. Note that this sequence of transformation rules is one given by the conclusion of Lemma 7.7 (we can always append a dummy use of ALPHA at the end of the transformation sequence, to perform the identity renaming). This is an example of a more general situation, in the remark below.

Remark 7.9 Let M_1 and M_2 be well-named λI-terms. (There is no need to consider multiterms when we restrict our attention to λI-terms — see Definition 6.4.) Let $\Delta_1 = \Gamma(M_1)$ and $\Delta_2 = \Gamma(M_2)$. Write $\Delta_1 \underset{\flat}{\longrightarrow} \Delta_2$ if there is a β-redex occurrence R in M_1 such that $M_1 \underset{\beta}{\xrightarrow{R}} M_2$ (R not necessarily leftmost) . We say that Δ is in $\flat$-nf iff there is a well-named λI-term M such that $\Delta = \Gamma(M)$ and M is in β-nf, but now, in constrast to a constraint set in $\sharp$-nf, it is not necessarily the case that $\Delta = \varnothing$. The interesting fact is that Lemma 7.7, with "$\sharp$" replaced by "$\flat$" throughout, still holds. We do not pursue this line of investigation further, because it would limit the final results to λI-terms, even though it would also simplify our entire analysis and make it more perspicuous.

Lemma 7.10 *If Δ_1 and Δ_2 are constraint sets such that $\Delta_1 \underset{\sharp}{\longrightarrow} \Delta_2$, then Δ_1 has a solution iff Δ_2 has a solution.*

Proof: Immediate consequence of Lemmas 5.5 and 7.7. ■

Lemma 7.11 *If Δ_1 and Δ_2 are constraint sets such that $\Delta_1 \underset{\sharp}{\longrightarrow} \Delta_2$, and φ_i is a minimal solution of Δ_i for $i = 1, 2$, then $|\mathrm{TVar}(\varphi_1(\Delta_1))| > |\mathrm{TVar}(\varphi_2(\Delta_2))|$.*

Proof: Immediate consequence of Lemmas 5.6 and 7.7. ■

Lemma 7.12 *Let $\mathfrak{M}$ be a well-named multiterm and $\Delta = \Gamma_a(\mathfrak{M})$. Then Δ is $\natural$-normalizing iff Δ has a solution.*

Proof: The left-to-right implication is an immediate consequence of (the right-to-left implication in) Lemma 7.10, what it means for Δ to be in $\natural$-nf, and by Lemma 7.5 (which implies that Δ is in $\natural$-nf iff $\Delta = \varnothing$). The empty constraint set $\varnothing$ always has a solution.

For the converse, assume $\Delta = \Delta_1$ has a solution and Δ_1 is not in $\natural$-nf. Consider a $\natural$-reduction sequence from Δ_1:

$$\Delta_1 \underset{\natural}{\longrightarrow} \Delta_2 \underset{\natural}{\longrightarrow} \Delta_3 \underset{\natural}{\longrightarrow} \cdots$$

By Lemma 7.10 (the left-to-right implication), Δ_k has a solution, for every $k \geqslant 1$. Let φ_k be a minimal solution of Δ_k. By Lemma 7.11,

$$|\mathrm{TVar}(\varphi_1(\Delta_1))| \;>\; |\mathrm{TVar}(\varphi_2(\Delta_2))| \;>\; |\mathrm{TVar}(\varphi_3(\Delta_3))| \;>\; \cdots$$

Hence, for some $k \geqslant 1$, it must be that $|\mathrm{TVar}(\varphi_k(\Delta_k))| = 0$. Because the type constant $\square$ occurs nowhere in Δ_1, and therefore nowhere in Δ_k, this implies $\Delta_k = \varnothing$ and Δ_1 is $\natural$-normalizing. ∎

Theorem 7.13 *Let $\mathfrak{M}$ be a well-named multiterm. Then $\mathfrak{M}$ is $[\beta]$-normalizing if and only if $\Gamma(\mathfrak{M})$ has a solution.*

Proof: We prove the following are equivalent conditions:

1. $\mathfrak{M}$ is $[\beta]$-normalizing.
2. $\Gamma_a(\mathfrak{M})$ is $\natural$-normalizing.
3. $\Gamma_a(\mathfrak{M})$ has a solution.
4. $\Gamma(\mathfrak{M})$ has a solution.

By Definition 7.6, we have (1) iff (2). By Lemma 7.12, we have (2) iff (3). By Lemma 5.5, we have (3) iff (4). ∎

Corollary 7.14 is, once again, our characterization of β-SN via the unification problem $\wedge$**UP**. (Corollary 3.8 gives a different proof of this characterization.)

Corollary 7.14 *For every well-named $M \in \Lambda$, M is β-SN iff $\Gamma(M)$ has a solution.*

Proof: By Theorem 6.5, M is β-SN iff $[M]$ is $[\beta]$-normalizing iff, by Theorem 7.13, $\Gamma(M)$ has a solution. ∎

Corollary 7.15 *For every well-named $M \in \Lambda$, M is β-SN iff M is typable in the system $\boldsymbol{\lambda}^{\rightarrow,\wedge}$.*

Proof: Immediate from Theorem 3.7 and Corollary 7.14. ■

Variations of the equivalence in Corollary 7.15 are well-known in the literature (see [2], [3], and the references cited therein); these are "variations" because they use formulations of the system of intersection types that are somewhat different from our $\boldsymbol{\lambda}^{\rightarrow,\wedge}$. One particular feature of the proof of 7.15 here is that it does not use an argument based on the method of "candidats de réductibilité" (or a weaker variant, such as the "realizability" method). This is not the only such proof: Several recent reports prove that typability in $\boldsymbol{\lambda}^{\rightarrow,\wedge}$ (or in minor variations of it) characterizes the class of λ-terms that are β-SN, by methods totally unrelated to "candidats de réductibilité", e.g. [3], [5] and [6].

Definition 7.16 ($\sharp$-transformation) The $\sharp$-transformation of a constraint set Δ is defined only if $\Delta = \Gamma(M)$ for some well-named λ-term M. It is a particular sequence of transformation rules in

$$\{+\text{SUBST}_1\}^* \ (\{\rightarrow\text{PARSE}\}\{\text{XPAND}\}\{\wedge\text{PARSE}\}^*\{-\text{SUBST}\}^+)^\infty$$

induced by the $\sharp$-reduction sequence that starts from $\Gamma(M)$. We make this notion precise. Let $\mathfrak{M}$ be a well-named multiterm and consider a $[\beta]$-reduction sequence starting from $\mathfrak{M}$:

$$\mathfrak{M} \equiv \mathfrak{M}_1 \ \xrightarrow[{[\beta]}]{R_1} \ \mathfrak{M}_2 \ \xrightarrow[{[\beta]}]{R_2} \ \mathfrak{M}_3 \ \xrightarrow[{[\beta]}]{R_3} \ \cdots$$

This $[\beta]$-reduction sequence is uniquely defined, because R_i is the unique leftmost β-redex occurrence in $\mathfrak{M}_i$, for $i = 1, 2, 3, \dots$. Hence, if $\Delta_i = \Gamma_a(\mathfrak{M}_i)$, the $\sharp$-reduction sequence

$$\Delta_1 \ \xrightarrow[\sharp]{} \ \Delta_2 \ \xrightarrow[\sharp]{} \ \Delta_3 \ \xrightarrow[\sharp]{} \ \cdots$$

is also uniquely defined. Starting from $\mathfrak{M} = [M]$ where M is a well-named λ-term, we therefore have a uniquely defined transformation sequence:

$$\Gamma(M) \quad \mathcal{X} \quad \Delta_1 \quad \mathcal{Y}_1 \quad \Delta_2 \quad \mathcal{Y}_2 \quad \Delta_3 \quad \mathcal{Y}_3 \qquad \cdots$$

where $\mathcal{X} \in \{+\text{SUBST}_1, \text{CLEAN}\}^*$ by Definition 7.3, and

$$\mathcal{Y}_i \ \in \ \{\rightarrow\text{PARSE}\}\{\text{XPAND}\}\{\wedge\text{PARSE}\}^*\{-\text{SUBST}\}^+\{\text{ALPHA}\}$$

by Lemma 7.7, for $i = 1, 2, 3, \dots$. Let $\widehat{\mathcal{X}}$ be the subsequence of $\mathcal{X}$ where all CLEAN's are omitted, and $\widehat{\mathcal{Y}}_i$ the prefix of $\mathcal{Y}_i$ where the last ALPHA is omitted, resulting in the following commutative diagram of transformations:

$$\begin{tikzcd}
\Gamma(M) & \widehat{\mathcal{X}} & \widehat{\Delta}_1 \arrow[d] & \widehat{\mathcal{Y}}_1 & \widehat{\Delta}_2 \arrow[d] & \widehat{\mathcal{Y}}_2 & \widehat{\Delta}_3 \arrow[d] & \widehat{\mathcal{Y}}_3 & \cdots \\
 & & \Delta_1 & \mathcal{Y}_1 & \Delta_2 & \mathcal{Y}_2 & \Delta_3 & \mathcal{Y}_3 & \cdots
\end{tikzcd}$$

where every downward arrow is a sequence in $\{\text{CLEAN}\}^*\{\text{ALPHA}\}$. The correctness of this diagram follows from Lemmas 5.7 and 5.8, according to which:

1. Every use of CLEAN can be displaced after a use of another transformation rule.

2. Every use of ALPHA can be displaced after a use of another transformation rule.

3. Consecutive uses of ALPHA can be replaced by a single use of ALPHA.

We call the $\sharp$-*transformation* of $\Gamma(M)$ the uniquely defined sequence $\mathcal{Z} = \widehat{\mathcal{X}}\ \widehat{\mathcal{Y}}_1\ \widehat{\mathcal{Y}}_2\ \widehat{\mathcal{Y}}_3\ \cdots$ which makes the preceding diagram commute. If the $[\beta]$-reduction sequence from $[M]$ (or, equivalently, the $\sharp$-reduction sequence from $\Gamma_a(M)$) terminates, then $\mathcal{Z}$ is a sequence in

$$\{+\text{SUBST}_1\}^*\ (\{\rightarrow\text{PARSE}\}\{\text{XPAND}\}\{\wedge\text{PARSE}\}^*\{-\text{SUBST}\}^+)^*$$

and if it does not terminate, then $\mathcal{Z}$ is a sequence in

$$\{+\text{SUBST}_1\}^*\ (\{\rightarrow\text{PARSE}\}\{\text{XPAND}\}\{\wedge\text{PARSE}\}^*\{-\text{SUBST}\}^+)^\omega$$

Another consequence of Theorem 7.13 is the following corollary, a somewhat weaker result than Conjecture 5.9.

Corollary 7.17 *Let M be a well-named λ-term. The following are equivalent conditions:*

1. *$\Gamma(M)$ has a solution.*

2. *The $\sharp$-transformation of $\Gamma(M)$ terminates with the empty constraint set $\varnothing$.*

Proof: Immediate from Theorem 7.13 and the definition of $\sharp$-transformation. ■

References

[1] Barendregt, H.P., *The Lambda Calculus, Its Syntax and Semantics*, revised edition, North-Holland, Amsterdam, 1984.

[2] Ghilezan, S., “Strong Normalization and Typability with Intersection Types”, *Notre Dame J. Formal Logic*, Vol 37, no. 1, Winter 1996.

[3] Kfoury, A.J. and Wells, J.B., “New Notions of Reduction and Non-Semantic Proofs of Beta Strong Normalization in Typed Lambda Calculi”, Proceedings of *Logic in Computer Science*, June 1995.

[4] Kfoury, A.J. and Wells, J.B., Addendum to “New Notions of Reduction and Non-Semantic Proofs of Beta Strong Normalization in Typed Lambda Calculi”, BUCS Tech Report 95-007, March 1995.

[5] Kfoury, A.J., “A Linearization of the λ-Calculus and an Application”, under preparation.

[6] Retoré, C., “A Note on Intersection Types”, INRIA report RR-2431, January 1995, postscript available at: `ftp://ftp.inria.fr/INRIA/publication/publi-ps-gz/RR/RR-2431.ps.gz`

[7] Sørensen, M.H., “Strong Normalization from Weak Normalization in Typed Lambda Calculi”, Report from CS Department, University of Copenhagen, 1996, available at URL: `http://www.diku.dk/research-groups/topps/personal/rambo.html`

[8] van Bakel, S., “Complete Restrictions of the Intersection Type Discipline”, *Theo. Comp. Sc.*, Vol 102, pp 135-163, 1992.

[9] van Bakel, S., *Intersection Type Disciplines in Lambda Calculus and Applicative Term Rewriting Systems*, Doctoral dissertation, Catholic University of Nijmegen, also issued by the Mathematisch Centrum, Amsterdam, 1993.