12-2008

Church and Curry: Combining Intrinsic and Extrinsic Typing

Frank Pfenning
Carnegie Mellon University

Church and Curry: Combining Intrinsic and Extrinsic Typing

Frank Pfenning

1 Introduction

Church's formulation of the simple theory of types [1] has the pleasing property that every well-formed term has a unique type. The type is thus an intrinsic attribute of a term. Furthermore, we can restrict attention to well-formed terms as the only meaningful ones, because the property of being well-formed is evidently and easily decidable.

Curry's formulation of combinatory logic [3], later adapted to the λ-calculus [9], assigns types to terms extrinsically and terms may have many different types. In other words, types capture properties of terms which have meaning independent of the types we might assign. This formulation very easily supports both finitary and infinitary polymorphism. Especially the former, expressed as *intersection types* [2], seems to be incompatible with uniqueness of types.

In this paper we show that it can be very fruitful to consider a two-layer approach. In the first layer, we have an intrinsically typed λ-calculus in the tradition of Church. A second layer of types, constructed in the tradition of Curry, captures properties of terms, but only those already well-formed according to the first layer. In order to avoid confusion, in the remainder of the paper we call types from the second layer *sorts* and use *types* only to refer to the first layer.

The resulting system combines the strengths of the two approaches. We can restrict attention to well-formed terms, and we exploit this when defining substitution and related operations. Moreover, we can easily define finitary sort polymorphism, and the system is quite precise in the properties it can assign to terms without losing decidability of sort checking.

The outline of the paper is as follows. We first define an intrinsically typed λ-calculus in the tradition of Church and consider canonical forms, which are β-normal and η-long. The theory of canonical forms includes a brief study of hereditary substitution and iterated expansion. We then define a system of sorts, including subsorting and intersection sorts, as a

second layer. Of particular interest for this layer are the preservation of sorts under hereditary substitution, its converse, and iterated expansion. Finally we extend subsorting to higher types and conclude with some remarks on related and future work.

Not much in this paper is new: hereditary substitution and iterated expansion go back to Watkins [13], who devised them to deal with the complexities of a dependent type theory with linearity and a monad. Much of the work on sorts and subsorting at higher types is joint work with Lovas [10] in the context of a dependent type theory or with Dunfield [6] in the context of functional programming. My goal in writing this paper was to distill these ideas to their purest form to make them as accessible as I could. In the process I rediscovered some of the virtues of Church's presentation of type theory. I also came to understand much better that the construction of sorts as a refinement of types represents a synthesis of Church's and Curry's approaches to type theory which are often seen as antithetical.

2 An Intrinsically Typed λ-Calculus

Church's formulation of type theory has two base types: o for truth values and ι for individuals. Since we are interested in the underlying λ-calculus and not the logic, we just consider ι as the only base type. We use a modern notation for function types; Church wrote $(\beta\,\alpha)$ for $\alpha \to \beta$.

DEFINITION 1 (Types). We define the set of types inductively.

1. ι is a type.
2. If α and β are types, then $\alpha \to \beta$ is a type.

We assume that for each type α we have an infinite supply of variables x^α. We also have typed constants c^α. Well-formed terms (henceforth just called *terms*) satisfy the following inductive definition.

DEFINITION 2 (Well-formed terms). We define the set of well-formed terms M^α of type α.

1. Any variable x^α or constant c^α is a term.
2. If x^α is a variable and M^β a term then $(\lambda x.\, M)^{\alpha\to\beta}$ is a term.
3. If $M_1^{\alpha\to\beta}$ and M_2^α are terms, then $(M_1\, M_2)^\beta$ is a term.

It is an easy inductive property that the type of a term is unique, and that we can decide if a term is well-formed according to these rules. We elide the definitions defining the usual notions of free and bound variables

and α-conversion, β-conversion, and η-conversion. We write $\mathrm{fv}(M)$ for the free variables in a term M. As is common practice, we will mostly omit type superscripts, since they can usually be determined from context.

> **Convention.** *In the remainder of this paper we assume that all terms are well-formed according to the above definition. Moreover, we will tacitly apply α-conversion to satisfy conditions on bound variables.*

3 Canonical Forms

In the area of logical frameworks, the notion of *canonical form* is particularly important. The representation methodology of logical frameworks such as LF [8] or hereditary Harrop formulas [11] puts the canonical forms of a given type in one-to-one correspondence with the objects to be modelled. To achieve this kind of bijection, canonical forms should be β-normal and η-long. The following mutually inductive definition of canonical and atomic terms captures this.

DEFINITION 3 (Canonical and atomic terms).

1. If x^α is a variable and N^β is canonical then $(\lambda x.\, N)^{\alpha\to\beta}$ is canonical.
2. If R^ι is atomic then R^ι is canonical.
3. If $R^{\beta\to\alpha}$ is atomic and N^β is canonical then $(R\,N)^\alpha$ is atomic.
4. A variable x^α or constant c^α is atomic.

We have made the intrinisic types of the terms explicit, but except for clause 2 where we must specify the type ι, the types are entirely redundant.

We can see from the definition that a canonical term cannot contain a β-redex, because the left-hand side of an application $R\,N$ must be atomic, which can only be a variable or another application. It is also fully η-expanded because the body of a λ-term is either another λ-term (in which case we do not care about its type) or an atomic term, in which case it must be of type ι.

> **Convention.** *Unless explicitly noted otherwise, in the remainder of this paper M and N stand for canonical terms and R stands for atomic terms.*

4 Hereditary Substitution

Once α-conversion is understood, the crucial operation in the λ-calculus is substitution, written as $[M/x]N$. Church only considered it if the type of M matches the type of the variable x, and if the bound variables of N are distinct from x and the free variables of M so as to avoid variable capture. The condition on the bound variables of N can always be achieved by renaming of bound variables, so substitution is a total operation modulo α-conversion.

Unfortunately, canonical forms are not closed under substitution. A simple example is $[(\lambda x^\iota.\, x)/y](y^{\iota\to\iota}\, z^\iota)$ which yields $(\lambda x^\iota.\, x)\, z$. If we are to develop a complete theory of canonical forms, we need a different operation. The central insight is that instead of creating the redex $(\lambda x^\iota.\, x)\, z$ we can spawn another substitution operation, $[z/x^\iota]x$. This new substitution *is at a smaller type* than the original substitution. In this example, we reduced the type from $\iota \to \iota$ to ι. This observation holds in general, as we show below.

We define *hereditary substitution* using three related operations. The principal operation is the substitution of a canonical term into a canonical term, yielding a canonical term. We write $[M^\alpha/x^\alpha]^n(N^\beta) = N'^\beta$. When substituting for x in an atomic term R we need to distinguish two cases: either the variable at the head of R is x, or else it is different from x. In the former case we may need to substitute hereditarily so as to avoid creating a redex. In the latter case we just proceed compositionally since the head of the result will remain unchanged.

We write $[M/x^\alpha]^{rn}(R^\beta) = N^\beta$ if the head of R is x and $[M^\alpha/x^\alpha]^{rr}(R^\beta) = R'^\beta$ if the head of R is different from x. The superscript indicates if we are mapping an atomic term (r) to another atomic term (r) or to a canonical term (n).

DEFINITION 4 (Head). We define the head $\mathrm{hd}(R)$ of an atomic term R with

$$\begin{array}{lcl} \mathrm{hd}(x) & = & x \\ \mathrm{hd}(c) & = & c \\ \mathrm{hd}(R\,N) & = & \mathrm{hd}(R) \end{array}$$

DEFINITION 5 (Hereditary substitution). We define three forms of hereditary substitution

1. $[M^\alpha/x^\alpha]^n(N^\beta) = N'^\beta$,
2. $[M^\alpha/x^\alpha]^{rr}(R^\beta) = R'^\beta$ for $\mathrm{hd}(R) \neq x$, and
3. $[M^\alpha/x^\alpha]^{rn}(R^\beta) = N^\beta$ for $\mathrm{hd}(R) = x$

by the following equations. They constitute an inductive definition, first on the structure of α and second on the structure of N and R, as confirmed by the subsequent theorem.

$$
\begin{array}{lcll}
[M/x^\alpha]^n(\lambda y.\, N) & = & \lambda y.\, [M/x^\alpha]^n N & \text{provided } y \neq x \text{ and } y \notin \mathrm{fv}(M) \\
[M/x^\alpha]^n(R) & = & [M/x^\alpha]^{rr}(R) & \text{if } \mathrm{hd}(R) \neq x \\
[M/x^\alpha]^n(R) & = & [M/x^\alpha]^{rn}(R) & \text{if } \mathrm{hd}(R) = x \\
\\
[M/x^\alpha]^{rr}(R_1\, N_2) & = & ([M/x^\alpha]^{rr} R_1)\, ([M/x^\alpha]^n N_2) & \\
[M/x^\alpha]^{rr}(y) & = & y & \text{for } x \neq y \\
[M/x^\alpha]^{rr}(c) & = & c & \\
\\
[M/x^\alpha]^{rn}(R_1\, N_2) & = & [N_2'/y_2^{\beta_2}]^n(N_1) & \text{where } [M/x^\alpha]^{rn}(R_1) = \lambda y_2^{\beta_2}.\, N_1^{\beta_1} \\
 & & & \text{and } [M/x^\alpha]^n(N_2) = N_2' \\
[M/x^\alpha]^{rn}(x) & = & M^\alpha &
\end{array}
$$

Hereditary substitution is always defined. The key observation is that if $[M/x^\alpha]^{rn}(R^\beta) = N^\beta$ then β is a subexpression of α. We refer to this property as *type reduction*. We write $\alpha \geq \beta$ if β is a subexpression of α, and $\alpha > \beta$ if β is a strict subexpression of α.[1] In the crucial last case of the hereditary substitution property below, we have $\alpha > \beta_2$ so the hereditary substitution does indeed take place at a strictly smaller type.

THEOREM 6 (Hereditary substitution).

1. $[M^\alpha/x^\alpha]^n(N^\beta) = N'^\beta$ *for a unique canonical* N'.
2. $[M^\alpha/x^\alpha]^{rr}(R^\beta) = R'^\beta$ *for a unique atomic* R' *if* $\mathrm{hd}(R) \neq x$.
3. $[M^\alpha/x^\alpha]^{rn}(R^\beta) = N'^\beta$ *for a unique canonical* N' *if* $\mathrm{hd}(R) = x$. *Furthermore, in this case,* $\alpha \geq \beta$.

Proof. Uniqueness is straightforward, since the clauses in the definition of hereditary substitution do not overlap.

We prove existence by nested induction, first on the type α and second the terms N and R. Furthermore, the type and term may remain the same when (1) appeals to (2) or (3), but must become strictly smaller when (2) or (3) appeal to (1).

Case(1): $N = \lambda y_2.\, N_1$ with $y_2 \neq x$ and $y_2 \notin \mathrm{fv}(M)$. Then

[1] This should not be confused with a subtyping as familiar from programming languages. We introduce a corresponding notion of *subsorting* later in this paper.

$[M/x^\alpha]^n N_1 = N_1'$ for some N_1' by i.h.(1) on α and N_1
$[M/x^\alpha]^n (\lambda y_2.\, N_1) = \lambda y_2.\, N_1'$ by defn. of $[\,]^n$.

Case(1): $N = R^\iota$ with $\mathrm{hd}(R) \neq x$. Then

$[M/x^\alpha]^{rr}(R) = R'$ for some R' by i.h.(2) on α and R
$[M/x^\alpha]^n(R) = R'$ by defn. of $[\,]^n$

Case(1): $N = R^\iota$ with $\mathrm{hd}(R) = x$. Then

$[M/x^\alpha]^{rn}(R) = N'$ for some N' by i.h.(3) on α and R
$[M/x^\alpha]^n(R) = N'$ by defn. of $[\,]^n$

Case(2): $R = x$. Impossible, since we assumed $\mathrm{hd}(R) \neq x$.

Case(2): $R = y$ with $y \neq x$. Then

$[M/x^\alpha]^{rr}(y) = y$ by defn. of $[\,]^{rr}$

Case(2): $R = c$. As in the previous case.

Case(2): $R = R_1\, N_2$. Then

$[M/x^\alpha]^{rr}(R_1) = R_1'$ for some R_1' by i.h.(2) on α and R_1
$[M/x^\alpha]^n(N_2) = N_2'$ for some N_2' by i.h.(1) on α and N_2
$[M/x^\alpha]^{rr}(R_1\, N_2) = R_1'\, N_2'$ by defn. of $[\,]^{rr}$

Case(3): $R = x$. Then

$[M/x^\alpha]^{rn}(x) = M^\alpha$ by defn. of $[\,]^{rn}$
$\alpha \geq \alpha$ by defn. of $\geq$

Case(3): $R = y$ with $y \neq x$ or $R = c$. Impossible, since we assumed $\mathrm{hd}(R) = x$.

Case(3): $R^\beta = R_1^{\beta_2 \to \beta_1}\, N_2^{\beta_2}$ with $\beta = \beta_1$. Then

$[M/x^\alpha]^{rn}(R_1) = N_1'$ for some $N_1'^{\beta_2 \to \beta_1}$ and
$\alpha \geq \beta_2 \to \beta_1$ by i.h.(3) on α and R_1
$N_1'^{\beta_2 \to \beta_1} = \lambda y_2.\, N_1$ for some $y_2^{\beta_2}$ and $N_1^{\beta_1}$ since N_1' is canonical
$[M/x^\alpha]^n(N_2) = N_2'$ by i.h.(1) on α and N_2
$\alpha > \beta_2$ by defn. of $>$
$[N_2'/y_2^{\beta_2}]^n(N_1) = N'$ by i.h.(1) on β_2 and N_1
$[M/x^\alpha]^n(R_1\, N_2) = N'$ by defn. of $[\,]^{rn}$
$\alpha > \beta_1$ by defn. of $>$

■

The hereditary substitution theorem expresses that we can substitute canonical terms for variables (which are atomic) in a canonical term and obtain a canonical term.

Conversely, if we have an atomic term, we can convert it to a canonical term by a process analogous to several η-expansions.

DEFINITION 7 (Iterated expansion). We define $\eta^\alpha(R^\alpha) = N^\alpha$ by induction on α.

$$\begin{array}{lcl} \eta^\iota(R) & = & R \\ \eta^{\alpha\to\beta}(R) & = & \lambda x^\alpha.\, \eta^\beta(R\,\eta^\alpha(x)) \quad \text{choosing } x \notin \mathrm{fv}(R) \end{array}$$

Again, this is easily seen to be well-founded and to return a canonical term.

THEOREM 8 (Iterated expansion). *$\eta^\alpha(R^\alpha) = N^\alpha$ for some canonical N.*

Proof. By induction on the structure of α.

Case: $\alpha = \iota$. Then

$\eta^\iota(R) = R$	by defn. of η
R^ι canonical	by defn. of canonical

Case: $\alpha = \alpha_2 \to \alpha_1$. Then

Let $x_2^{\alpha_2}$ be a variable not in $\mathrm{fv}(R)$	
$\eta^{\alpha_2}(x_2) = N_2$ for some canonical N_2	by i.h. on α_2
$\eta^{\alpha_1}(R\,N_2) = N_1$ for some canonical N_1	by i.h. on α_1
$\eta^\alpha(R) = \lambda x_2.\, N_1$	by defn. of η
$\lambda x_2.\, N_1$ canonical	by defn. of canonical

■

We have been especially detailed in the analysis of the definitions of hereditary substitution and iterated expansion because the inductive patterns of these definitions recur multiple times in our development below.

5 Composition and Identity

With ordinary substitution, we usually need some simple lemmas that show composition and identity properties. For example, $[M_1/x_1][M_2/x_2]N =$

$[[M_1/x_1]M_2/x_2]([M_1/x_1]N)$ and $[x/x]N = N$. The corresponding properties for hereditary substitution are bit more complex to state because we need to obey the discipline of canonical and atomic terms which entails that there are several forms of hereditary substitution, as we have seen in the previous section.

Also, proofs become a bit more tedious. In the cases of ordinary substitution above, they are straightforward by induction on the structure of N. Hereditary substitutions are defined by nested induction on a type and a term, so the proofs of the composition properties employ a corresponding nested induction.

When all is said and done, though, these proofs are really not much more difficult since the structure of the definitions guides every single step of the development.

We need one more preparatory lemma.

THEOREM 9 (Vacuous Substitution).

1. $[M/x]^n(N) = N$ *if* $x \notin \mathrm{fv}(N)$
2. $[M/x]^{rr}(R) = R$ *if* $x \notin \mathrm{fv}(R)$

Proof. By straightforward induction on the structure of N and R. Note that $[M/x]^{rn}(R)$ is never needed when $x \notin \mathrm{fv}(R)$ because $\mathrm{hd}(R)$ can not be x. ■

THEOREM 10 (Composition of hereditary substitutions). *Assume* $x_1 \neq x_2$ *and* $x_2 \notin \mathrm{fv}(M_1)$. *Then*

$$[M_1/x_1]^n([M_2/x_2]^n(N)) = [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x]^n(N))$$

Proof. We generalize to the following statements, assuming $x_1 \neq x_2$ and $x_2 \notin \mathrm{fv}(M_1)$.

1. $[M_1/x_1]^n([M_2/x_2]^n(N)) = [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x]^n(N))$
2. $[M_1/x_1]^{rr}([M_2/x_2]^{rr}(R)) = [[M_1/x_1]^n(M_2)/x_2]^{rr}([M_1/x_1]^{rr}(R))$
   if $\mathrm{hd}(R) \neq x_1$ and $\mathrm{hd}(R) \neq x_2$
3. $[M_1/x_1]^n([M_2/x_2]^{rn}(R)) = [[M_1/x_1]^n(M_2)/x_2]^{rn}([M_1/x_1]^{rr}(R))$
   if $\mathrm{hd}(R) = x_2$
4. $[M_1/x_1]^{rn}([M_2/x_2]^{rr}(R)) = [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x_1]^{rn}(R))$
   if $\mathrm{hd}(R) = x_1$

Let $x_1^{\alpha_1}$ and $x_2^{\alpha_2}$. Then the proof proceeds by nested induction, first on α_1 and α_2, and second on the terms N and R. Also, part (1) may appeal to parts (2), (3), and (4) with the same types and terms, but any other appeal has to strictly decrease the induction measure. For the outer induction, when α_1 decreases then α_2 stays the same and vice versa, and in the first case of part (4) they change roles, so one could take the unordered pair of the two types or the sum of their sizes as the induction measure.

Case(1): $N = \lambda y_2.\, N_1$.

$$
\begin{array}{ll}
& [M_1/x_1]^n([M_2/x_2]^n(\lambda y_2.\, N_1)) \\
= & [M_1/x_1]^n(\lambda y_2.\, [M_2/x_2]^n(N_1)) & \text{by defn. of } []^n \\
= & \lambda y_2.\, [M_1/x_1]^n([M_2/x_2]^n(N_1)) & \text{by defn. of } []^n \\
= & \lambda y_2.\, [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x_1]^n(N_1)) & \text{by i.h.(1) on } \alpha_1,\, \alpha_2,\, N_1 \\
= & [[M_1/x_1](M_2)/x_2]^n([M_1/x_1](\lambda y_2.\, N_1)) & \text{by defn. of } []^n \text{ (twice)}
\end{array}
$$

Case(1): $N = R$ with $\mathrm{hd}(R_1) \neq x_1$ and $\mathrm{hd}(R_1) \neq x_2$.

$$
\begin{array}{ll}
& [M_1/x_1]^n([M_2/x_2]^n(R)) \\
= & [M_1/x_1]^{rr}([M_2/x_2]^{rr}(R)) & \text{by defn. of } []^n \text{ (twice)} \\
= & [[M_1/x_1]^n(M_2)/x_2]^{rr}([M_1/x_1]^{rr}(R)) & \text{by i.h.(2) on } \alpha_1,\, \alpha_2,\, R \\
= & [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x_1]^n(R)) & \text{by defn. of } []^n \text{ (twice)}
\end{array}
$$

Case(1): $N = R$ with $\mathrm{hd}(R) = x_2$.

$$
\begin{array}{ll}
& [M_1/x_1]^n([M_2/x_2]^n(R)) \\
= & [M_1/x_1]^n([M_2/x_2]^{rn}(R)) & \text{by defn. of } []^n \\
= & [[M_1/x_1]^n(M_2)/x_2]^{rn}([M_1/x_1]^{rr}(R)) & \text{by i.h.(3) on } \alpha_1,\, \alpha_2,\, R \\
= & [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x_1]^n(R)) & \text{defn. of } []^n \text{ (twice)}
\end{array}
$$

Case(1): $N = R$ with $\mathrm{hd}(R) = x_1$.

$$
\begin{array}{ll}
& [M_1/x_1]^n([M_2/x_2]^n(R)) \\
= & [M_1/x_1]^{rn}([M_2/x_2]^{rr}(R)) & \text{by defn. of } []^n \text{ (twice)} \\
= & [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x_1]^{rn}(R)) & \text{by i.h.(4) on } \alpha_1,\, \alpha_2,\, R \\
= & [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x_1]^n(R)) & \text{by defn. of } []^n
\end{array}
$$

Case(2): $R = R_1\, N_2$ with $\mathrm{hd}(R) \neq x_1$ and $\mathrm{hd}(R) \neq x_2$.

$$
\begin{array}{l}
[M_1/x_1]^{rr}([M_2/x_2]^{rr}(R_1\, N_2)) \\
= [M_1/x_1]^{rr}([M_2/x_2]^{rr}(R_1))\ [M_1/x_1]^n([M_2/x_2]^n(N_2))
\end{array}
$$

$$
\begin{array}{rl}
 & \text{by defn. of } [\,]^{rr} \text{ (twice)} \\
= [[M_1/x_1]^n(M_2)/x_2]^{rr}([M_1/x_1]^{rr}(R_1))\ [M_1/x_1]^n([M_2/x_2]^n(N_2)) & \\
 & \text{by i.h.(2) on } \alpha_1,\ \alpha_2,\ R_1 \\
= [[M_1/x_1]^n(M_2)/x_2]^{rr}([M_1/x_1]^{rr}(R_1)) & \\
\qquad [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x_1]^n(N_2)) & \text{by i.h.(1) on } \alpha_1,\ \alpha_2,\ N_2 \\
= [[M_1/x_1]^n(M_2)/x_2]^{rr}([M_1/x_1]^{rr}(R_1\ N_2)) & \text{by defn. of } [\,]^{rr} \text{ (twice)}
\end{array}
$$

Case(2): $R = y$ with $y \neq x_1$ and $y \neq x_2$.

$$
\begin{array}{rl}
 & [M_1/x_1]^{rr}([M_2/x_2]^{rr}(y)) \\
= y & \text{by defn. of } [\,]^{rr} \text{ (twice)} \\
= [[M_1/x_1]^{rr}(M_2)/x_2]([M_1/x_1]^{rr}(y)) & \text{by defn. of } [\,]^{rr} \text{ (twice)}
\end{array}
$$

Case(2): $R = c$. Like the previous case.

Case(3): $R = R_1\ N_2$ with $\mathrm{hd}(R_1) = x_2$.

$$
\begin{array}{rl}
 & [M_1/x_1]^n([M_2/x_2]^{rn}(R_1\ N_2)) \\
= [M_1/x_1]^n([[M_2/x_2]^n(N_2)/y_2]^n(N_1)) & \text{by defn. of } [\,]^{rn} \text{ and (*)} \\
= [[M_1/x_1]^n([M_2/x_2]^n(N_2))/y_2]^n([M_1/x_1]^n(N_1)) & \\
 & \text{by i.h.(1) on } \alpha_1 \text{ and } \beta_2 \\
= [[M_1/x_1]^n(M_2)/x_2]^{rn}([M_1/x_1]^{rr}(R_1)\,[M_1/x_1]^n(N_2)) & \\
 & \text{by defn. of } [\,]^{rn} \text{ and (**) and (***)} \\
= [[M_1/x_1]^n(M_2)/x_2]^{rn}([M_1/x_1]^{rr}(R_1\ N_2)) & \text{by defn. of } [\,]^{rr}
\end{array}
$$

where (*)

$[M_2/x_2]^{rn}(R_1) = \lambda y_2.\, N_1$ for some $y_2^{\beta_2}$ and $N_1^{\beta_1}$ by hered. subst.
and $\alpha_2 \geq \beta_2 \to \beta_1$ by type reduction of $[\,]^{rn}$

and (**)

$$
\begin{array}{rl}
 & [[M_1/x_1]^n(M_2)/x_2]^{rn}([M_1/x_1]^{rr}(R_1)) \\
= [M_1/x_1]^n([M_2/x_2]^{rn}(R_1)) & \text{by i.h.(3) on } \alpha_1,\ \alpha_2,\ R_1 \\
= [M_1/x_1]^n(\lambda y_2.\, N_1) & \text{equality (*)} \\
= \lambda y_2.\, [M_1/x_1]^n(N_1) & \text{by defn. of } [\,]^n
\end{array}
$$

and (***)

$$
\begin{array}{rl}
 & [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x_1]^n(N_2)) \\
= [M_1/x_1]^n([M_2/x_2]^n(N_2)) & \text{by i.h.(1) on } \alpha_1,\ \alpha_2,\ N_2
\end{array}
$$

Case(3): $R = x_2$.

$[M_1/x_1]^n([M_2/x_2]^{rn}(x_2))$
$= [M_1/x_1]^n(M_2)$ by defn. of $[\,]^{rn}$
$= [[M_1/x_1]^n(M_2)/x_2]^{rn}(x_2)$ by defn. of $[\,]^{rn}$
$= [[M_1/x_1]^n(M_2)/x_2]^{rn}([M_1/x_1]^{rr}(x_2))$ by defn. of $[\,]^{rr}$

Case(4): $R = R_1\, N_2$ with $\mathrm{hd}(R_1) = x_1$.

$[M_1/x_1]^{rn}([M_2/x_2]^{rr}(R_1\, N_2))$
$= [M_1/x_1]^{rn}([M_2/x_2]^{rr}(R_1)\, [M_2/x_2]^n(N_2))$ by defn. of $[\,]^{rr}$
$= [[[M_1/x_1]^n(M_2)/x_2]^n([M_1/x_1]^n(N_2))/y_2]^n([[M_1/x_1](M_2)/x_2](N_1))$
by defn. of $[\,]^{rn}$, (**) and (***)
$= [[M_1/x_1]^n(M_2)/x_2]^n([[M_1/x_1]^n(N_2)/y_2]^n(N_1))$
by i.h.(1) on α_2, β_2
$= [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x_1]^{rn}(R_1\, N_2))$ by defn. of $[\,]^{rn}$ and (*)

where (*)

$[M_1/x_1]^{rn}(R_1) = \lambda y_2.\, N_1$ for some $y_2^{\beta_2}$ and $N_1^{\beta_1}$ by hered. subst.
and $\alpha_1 \geq \beta_2 \to \beta_1$ by type reduction of $[\,]^{rn}$

and (**)

$[M_1/x_1]^{rn}([M_2/x_2]^{rr}(R_1))$
$= [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x_1]^{rn}(R_1))$ by i.h.(4) on α_1, α_2 and R_1
$= [[M_1/x_1]^n(M_2)/x_2]^n(\lambda y_2.\, N_1)$ by equality (*)
$= \lambda y_2.\, [[M_1/x_1]^n(M_2)/x_2]^n(N_1)$ by defn. of $[\,]^n$

and (***)

$[M_1/x_1]^n([M_2/x_2]^n(N_2))$
$= [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x_1]^n(N_2))$ by i.h.(1) on α_1, α_2, N_2

Case(4): $R = x_1$.

$[M_1/x_1]^{rn}([M_2/x_2]^{rr}(x_1))$
$= [M_1/x_1]^{rn}(x_1)$ by defn. of $[\,]^{rr}$
$= M_1$ by defn. of $[\,]^{rn}$
$= [[M_1/x_1]^n(M_2)/x_2]^n(M_1)$ by vacuous substitution
$= [[M_1/x_1]^n(M_2)/x_2]^n([M_1/x_1]^{rn}(x_1))$ by defn. of $[\,]^{rn}$

■

The identity property of the ordinary substitution $[x/x]N = N$ is almost trivial, and the reverse $[N/x]x = N$ is part of the definition of substitution.

Here, the identity properties reveal an interplay between iterated expansion and hereditary substitution. We cannot substitute $[x/x]$ because x may not be canonical. Instead, we substitute the iterated expansion of x, so the *left identity* property shows that the iterated expansion of a variable behaves like the variable itself under substitution. Conversely, the *right identity* shows that substituting into the expansion of a variable amounts to substituting into the variable itself.

THEOREM 11 (Identity of iterated expansion).

1. *(Left identity)* $[\eta^\alpha(x)/x]^n(N) = N$
2. *(Right identity)* $[N/x]^n(\eta^\alpha(x)) = N$

Proof. We generalize to the following five properties.

1. $[\eta^\alpha(x)/x]^n(N) = N$
2. $[\eta^\alpha(x)/x]^{rr}(R) = R$ if $\mathrm{hd}(R) \neq x$
3. $[\eta^\alpha(x)/x]^{rn}(R) = \eta^\beta(R^\beta)$ if $\mathrm{hd}(R) = x$
4. $[N/x]^n(\eta^\alpha(R)) = \eta^\alpha([N/x]^{rr}(R))$ if $\mathrm{hd}(R) \neq x$
5. $[N/x]^n(\eta^\alpha(R)) = [N/x]^{rn}(R)$ if $\mathrm{hd}(R) = x$

Right identity follows from part (5) using $R = x$.

The proof is by nested induction, first on α and second on N and R. Also, (1) may appeal to (2) and (3) with unchanged type or term, but appeals from (2) and (3) to any part will strictly decrease the measure of the parameters.

Case(1): $N = \lambda y_2.\, N_1$ for $y_2 \neq x$.

$$
\begin{array}{ll}
& [\eta^\alpha(x)/x]^n(\lambda y_2.\, N_1) \\
= & \lambda y_2.\, [\eta^\alpha(x)/x]^n(N_1) \qquad \text{by defn. of } []^n \\
= & \lambda y_2.\, N_1 \qquad \text{by i.h.(1) on } \alpha \text{ and } N_1
\end{array}
$$

Case(1): $N = R^\iota$ with $\mathrm{hd}(R) \neq x$.

$$
\begin{array}{ll}
& [\eta^\alpha(x)/x]^n(R) \\
= & [\eta^\alpha(x)/x]^{rr}(R) \qquad \text{by defn. of } []^n \\
= & R \qquad \text{by i.h.(2) on } \alpha \text{ and } R
\end{array}
$$

Case(1): $N = R^{\iota}$ with $\mathrm{hd}(R) = x$.

$$
\begin{array}{ll}
[\eta^{\alpha}(x)/x]^{n}(R) & \\
= [\eta^{\alpha}(x)/x]^{rn}(R) & \text{by defn. of } []^{n} \\
= \eta^{\iota}(R) & \text{by i.h.(3) on } \alpha \text{ and } R \\
= R & \text{by defn. of } \eta
\end{array}
$$

Case(2): $R = R_1\, N_2$.

$$
\begin{array}{ll}
[\eta^{\alpha}(x)/x]^{rr}(R_1\, N_2) & \\
= [\eta^{\alpha}(x)/x]^{rr}(R_1)\, [\eta^{\alpha}(x)/x]^{n}(N_2) & \text{by defn. of } []^{rr} \\
= R_1\, [\eta^{\alpha}(x)/x]^{n}(N_2) & \text{by i.h.(2) on } \alpha \text{ and } R_1 \\
= R_1\, N_2 & \text{by i.h.(1) on } \alpha \text{ and } N_2
\end{array}
$$

Case(2): $R = y$ for $y \neq x$.

$$
\begin{array}{ll}
[\eta^{\alpha}(x)/x]^{rr}(y) & \\
= y & \text{by defn. of } []^{rr}
\end{array}
$$

Case(2): $R = c$. Like the previous case.

Case(3): $R = R_1^{\beta_2 \to \beta_1}\, N_2^{\beta_2}$ where $\mathrm{hd}(R_1) = x$.

$$
\begin{array}{ll}
[\eta^{\alpha}(x)/x]^{rn}(R_1\, N_2) & \\
= [N_2/y_2]^{n}(\eta^{\beta_1}(R_1\, \eta^{\beta_2}(y_2))) & \text{by defn. of } []^{rn}, \text{ (*) and (**)} \\
= \eta^{\beta_1}([N_2/y_2]^{rr}(R_1\, \eta^{\beta_2}(y_2))) & \text{by i.h.(4) on } \beta_1 \\
= \eta^{\beta_1}([N_2/y_2]^{rr}(R_1)\, [N_2/y_2]^{n}(\eta^{\beta_2}(y_2))) & \text{by defn. of } []^{rr} \\
= \eta^{\beta_1}(R_1\, [N_2/y_2]^{n}(\eta^{\beta_2}(y_2))) & \text{by vacuous substitution} \\
= \eta^{\beta_1}(R_1\, [N_2/y_2]^{rn}(y_2)) & \text{by i.h.(5) on } \beta_2 \\
= \eta^{\beta_1}(R_1\, N_2) & \text{by defn. of } []^{rn}
\end{array}
$$

where (*)

$$
\begin{array}{ll}
[\eta^{\alpha}(x)/x]^{rn}(R_1) & \\
= \eta^{\beta_2 \to \beta_1}(R_1) & \text{by i.h.(3) on } \alpha \text{ and } R_1 \\
= \lambda y_2.\, \eta^{\beta_1}(R_1\, \eta^{\beta_2}(y_2)) & \text{by defn. of } \eta \\
\alpha \geq \beta_2 \to \beta_1 & \text{by type reduction of } []^{rn}
\end{array}
$$

and (**)

$$
\begin{array}{ll}
[\eta^{\alpha}(x)/x]^{n}(N_2) & \\
= N_2 & \text{by i.h.(1) on } \alpha \text{ and } N_2
\end{array}
$$

Case(3): $R = x$.

$[\eta^\alpha(x)/x]^{rn}(x)$
$= \eta^\alpha(x)$ by defn. of $[\,]^{rn}$

Case(4): $\alpha = \alpha_2 \to \alpha_1$ and $\mathrm{hd}(R) \neq x$.

$[N/x]^n(\eta^{\alpha_2\to\alpha_1}(R))$
$= [N/x]^n(\lambda y_2.\, \eta^{\alpha_1}(R\,\eta^{\alpha_2}(y_2)))$ by defn. of η
$= \lambda y_2.\, [N/x]^n(\eta^{\alpha_1}(R\,\eta^{\alpha_2}(y_2)))$ by defn. of $[\,]^n$
$= \lambda y_2.\, \eta^{\alpha_1}([N/x]^{rr}(R\,\eta^{\alpha_2}(y_2)))$ by i.h.(4) on α_1
$= \lambda y_2.\, \eta^{\alpha_1}([N/x]^{rr}(R)\,[N/x]^n(\eta^{\alpha_2}(y_2)))$ by defn. of $[\,]^{rr}$
$= \lambda y_2.\, \eta^{\alpha_1}([N/x]^{rr}(R)\,\eta^{\alpha_2}([N/x]^{rr}(y_2)))$ by i.h.(4) on α_2
$= \lambda y_2.\, \eta^{\alpha_1}([N/x]^{rr}(R)\,\eta^{\alpha_2}(y_2))$ by defn. of $[\,]^{rr}$
$= \eta^{\alpha_2\to\alpha_1}([N/x]^{rr}(R))$ by defn. of η

Case(4): $\alpha = \iota$ and $\mathrm{hd}(R) \neq x$.

$[N/x]^n(\eta^\iota(R))$
$= [N/x]^n(R)$ by defn. of η
$= [N/x]^{rr}(R)$ by defn. of $[\,]^n$
$= \eta^\iota([N/x]^{rr}(R))$ by defn. of η

Case(5): $\alpha = \alpha_2 \to \alpha_1$ and $\mathrm{hd}(R) = x$.

$[N/x]^n(\eta^{\alpha_2\to\alpha_1}(R))$
$= [N/x]^n(\lambda y_2.\, \eta^{\alpha_1}(R\,\eta^{\alpha_2}(y_2)))$, $y_2 \notin \mathrm{fv}(N) \cup \mathrm{fv}(R)$ by defn. of η
$= \lambda y_2.\, [N/x]^n(\eta^{\alpha_1}(R\,\eta^{\alpha_2}(y_2)))$ by defn. of $[\,]^n$
$= \lambda y_2.\, [N/x]^{rn}(R\,\eta^{\alpha_2}(y_2))$ by i.h.(5) on α_1
$= \lambda y_2.\, [\eta^{\alpha_2}(y_2)/y_2]^n(N_1)$ by defn. of $[\,]^{rn}$, (*) and (**)
$= \lambda y_2.\, N_1$ by i.h.(1) on α_2
$= [N/x]^{rn}(R)$ by equality (*)

where (*)

$[N/x]^{rn}(R)$
$= \lambda y_2.\, N_1$ for some $N_1^{\alpha_1}$ by hereditary substitution and renaming, since $y_2 \notin \mathrm{fv}(N) \cup \mathrm{fv}(R)$

and (**)

$[N/x]^n(\eta^{\alpha_2}(y_2))$
$= \eta^{\alpha_2}([N/x]^{rr}(y_2))$ by i.h.(4) on α_2
$= \eta^{\alpha_2}(y_2)$ by defn. of $[\,]^{rr}$

Case(5): $\alpha = \iota$ and $\mathrm{hd}(R) = x$.

$$\begin{array}{ll} [N/x]^n(\eta^\iota(R)) & \\ = [N/x]^n(R) & \text{by defn. of } \eta \\ = [N/x]^{rn}(R) & \text{by defn. of } []^n \end{array}$$

■

6 Simple Sorts

Now that we have defined an intrinsically typed λ-calculus in the tradition of Church, we can define a second layer of *sorts* in the tradition of Curry [3]. In this and the following section we temporarily allow general well-formed terms, and not just canonical forms.

We begin by considering the type ι. For Church's purposes, a single type of individuals was sufficient: other "types" could be explicitly defined using the higher-order features of his type theory, or one could define new constants and represent sorts as predicates. For example, if we wanted to introduce natural numbers, we might declare constants z^ι, $\mathsf{s}^{\iota\to\iota}$, and $\mathsf{nat}^{\iota\to o}$ and the assumptions

$$\begin{array}{l} \mathsf{nat}(\mathsf{z}) \\ \forall x^\iota.\, \mathsf{nat}(x) \supset \mathsf{nat}(\mathsf{s}(x)) \end{array}$$

However, reasoning with natural numbers now requires a good deal of explicit inferences with the nat predicate, while from a modern perspective it seems that static type-checking should be able ascertain when a given term represents a natural number.

In order to explore this, we allow declarations of sorts such as nat. In addition we can declare new constants and give them sorts. For example:

$$\begin{array}{lcl} \mathsf{nat}^\iota \text{ sort} & & \\ \mathsf{z}^\iota & : & \mathsf{nat} \\ \mathsf{s}^{\iota\to\iota} & : & \mathsf{nat} \to \mathsf{nat} \end{array}$$

The first declaration states that nat is a new sort constant that represents a subset of the terms of type ι. We say that nat *refines* ι. We use Q for sort constants, also called base sorts.

From the last declaration we can see that for this new device to be useful, the language of sorts needs to include *function sorts*, and they must match the types in a consistent way. Declarations for base sorts and constants are collected in a signature Σ. We call these simple sorts, since we extend them later to include intersections, and because they match simple types.

DEFINITION 12 (Simple sorts). We define simple sorts inductively with respect to a signature Σ.

1. A base sort Q^ι declared in Σ is a simple sort.

2. If S^α and T^β are simple sorts, then $(S \to T)^{\alpha\to\beta}$ is a simple sort.

The general form of a constant declaration is $c^\alpha : S^\alpha$, that is, the sort assigned to a constant must refine its type. This is essential in this two-layer approach so we do not lose the good properties of the underlying intrinsically typed λ-calculus.

We follow a similar restriction when we assign sorts to variables $x^\alpha : S^\alpha$. Since sorts are extrinsic to variables, just as they are extrinsic to terms, we must now explicitly track the sort we would like to assign to a variable. We collect the sort assignments to variables in a *context* Γ which is a collection of variable declarations as above. We write $\mathrm{dv}(\Gamma)$ for the set of variables declared in a context.

More generally, type assignment is a relation between a term and a sort in a context. We write $\Gamma \vdash M^\alpha : S^\alpha$. In modern parlance we refer to this as a *judgment* and define it by a set of inference rules, but it could just as well be written out as an inductive definition. Interestingly, Curry's original paper also used an axiomatic formulation to establish that a term has a sort by logical reasoning, while Church's original paper gave typing as an explicitly inductive definition. We assume all sort and term constants are declared in a fixed signature Σ.

$$\frac{x{:}S \in \Gamma}{\Gamma \vdash x : S} \qquad \frac{c{:}S \in \Sigma}{\Gamma \vdash c : S}$$

$$\frac{\Gamma, x{:}S \vdash M : T \quad (x \notin \mathrm{dv}(\Gamma))}{\Gamma \vdash \lambda x.\, M : S \to T} \qquad \frac{\Gamma \vdash M : S \to T \quad \Gamma \vdash N : S}{\Gamma \vdash M\, N : T}$$

We have omitted all the types, but it is essential to remember that we consider $M : S$ only when M has type α and S refines α, that is, $M^\alpha : S^\alpha$. If we ignore this restriction, then the rules above define a Curry-Howard style type assignment system for the simply-typed λ-calculus [9] which is essentially isomorphic to Church's system. We do not precisely formulate or prove such a theorem here in order to concentrate on richer sort systems.

7 Subsorts

The first generalization will be to allow subsorts. Continuing the previous example, we can distinguish the sort with just the constant z (zero) and the sort of all positive terms (pos). We also express that they are *subsorts* of

the sort of natural numbers (nat).

$$\begin{array}{lcl} \mathsf{zero} & \leq & \mathsf{nat} \\ \mathsf{pos} & \leq & \mathsf{nat} \\ \mathsf{z} & : & \mathsf{zero} \\ \mathsf{s} & : & \mathsf{nat} \to \mathsf{pos} \end{array}$$

We now also allow multiple sort declarations for constants. Since a sort represents a property of terms, this is perfectly natural. We also need a rule of *subsumption*, allowing us to exploit the knowledge of subsorting. Remarkably, the remainder of the system does not need to be changed.

$$\frac{}{Q \leq Q} \qquad \frac{Q_1 \leq Q_2 \quad Q_2 \leq Q_3}{Q_1 \leq Q_3}$$

$$\frac{\Gamma \vdash M : Q \quad Q \leq Q'}{\Gamma \vdash M : Q'}$$

We can now prove, for example, $\mathsf{s}\,\mathsf{z} : \mathsf{pos}$ or $\lambda x.\,\mathsf{s}\,x : \mathsf{zero} \to \mathsf{pos}$.

However, this system is unfortunately very weak. For example, $\lambda x.\,x$ has the property expressed by the sort $(\mathsf{nat} \to \mathsf{zero}) \to (\mathsf{zero} \to \mathsf{nat})$. In words: any function mapping an arbitrary natural number to 0 will also map 0 to some natural number. But this typing cannot be established in the system above, because subtyping is too weak. Fortunately, this problem can be solved by strengthening the subtyping relation and raising it to higher type, as we will do in Section 12.

8 Sort Checking Canonical Forms

We now restrict our attention again to canonical forms. Careful study of a sorting system for canonical forms can be transferred to a calculus that permits arbitrary well-formed terms, although we do not pursue this transfer in this paper.

The rules for sort assignment in the previous section do not immediately describe an algorithm for performing sort checking. If we try to infer sorts for terms, then the rule for λ-abstraction presents a problem in that, even if we can infer the sort of the body, the sort of the bound variable has to be guessed. Conversely, if we instead try to check a term against a known sort, then the rule for application creates problems because we do not know the sort of the argument. This is in contrast to Church's system where we can systematically construct the unique type of a term (or fail, if it is not well-formed). One avenue now would be to introduce unification into the sort-checking process, but this will be difficult to extend to more expressive

sort systems. Instead, we observe that on *canonical forms* sort-checking can be performed very elegantly.

We distinguish two judgments, $\Gamma \vdash N \Leftarrow S$ and $\Gamma \vdash R \Rightarrow S$. For the first one, we envision Γ, N, and S to be given, and we want to verify if N has sort S. For the second, we assume that Γ and R are given, and we want to synthesize a sort S for R. At the place where checking meets synthesis (or: where an atomic term is viewed as a canonical one) we must have a base sort that refines ι.

$$\frac{x{:}S \in \Gamma}{\Gamma \vdash x \Rightarrow S} \qquad \frac{c{:}S \in \Sigma}{\Gamma \vdash c \Rightarrow S}$$

$$\frac{\Gamma, x{:}S \vdash N \Leftarrow T \quad (x \notin \mathrm{dv}(\Gamma))}{\Gamma \vdash \lambda x.\, N \Leftarrow S \to T} \qquad \frac{\Gamma \vdash R \Rightarrow S \to T \quad \Gamma \vdash N \Leftarrow S}{\Gamma \vdash R\, N \Rightarrow T}$$

$$\frac{\Gamma \vdash R \Rightarrow Q' \quad Q' \leq Q}{\Gamma \vdash R \Leftarrow Q}$$

This system can easily be recognized as a decision procedure for sorting of terms, based on the intuition given above, as long as basic subsorting is also decidable. Since $Q' \leq Q$ is just the reflexive and transitive closure of the subsorting declarations in the signature, this also holds.

We hold off on a formal statement of this property, since we prove a more general statement in the next section. We close this section by reconsidering the example showing incompleteness of the rules from the previous section. We noted that

$$\not\vdash \lambda x.\, x : (\mathsf{nat} \to \mathsf{zero}) \to (\mathsf{zero} \to \mathsf{nat})$$

This term, $\lambda x^{\iota \to \iota}.\, x$, is not canonical. If we expand it to its canonical form we obtain

$$\lambda x^{\iota \to \iota}.\, \eta^{\iota \to \iota}(x) = \lambda x.\, \lambda y.\, \eta^{\iota}(x\, \eta^{\iota}(y)) = \lambda x.\, \lambda y.\, x\, y$$

Now we can indeed verify that

$$\vdash \lambda x.\, \lambda y.\, x\, y \Leftarrow (\mathsf{nat} \to \mathsf{zero}) \to (\mathsf{zero} \to \mathsf{nat})$$

This derivation uses $\mathsf{zero} \leq \mathsf{nat}$ twice: once for y and once for $x\, y$.

9 Intersection Sorts

The system so far, on canonical forms, has a number of desirable properties, but it lacks the basic ability to ascribe more than one property to a term.

So we can prove, for example,

$$\begin{array}{l} \vdash \lambda x^\iota.\, x \Leftarrow \mathsf{nat} \to \mathsf{nat} \\ \vdash \lambda x^\iota.\, x \Leftarrow \mathsf{zero} \to \mathsf{zero} \\ \vdash \lambda x^\iota.\, x \Leftarrow \mathsf{pos} \to \mathsf{pos} \end{array}$$

but we cannot combine these pieces of information into a single judgment. At first one might think of a form of parametric polymorphism to solve this problem, for example, $\forall q^\iota.\, q \to q$ as a common sort. To see that this would not be sufficient, consider the additional sorts even and odd.

$$\begin{array}{lcl} \mathsf{even} & \leq & \mathsf{nat} \\ \mathsf{odd} & \leq & \mathsf{nat} \\ \mathsf{z} & : & \mathsf{even} \\ \mathsf{s} & : & \mathsf{even} \to \mathsf{odd} \\ \mathsf{s} & : & \mathsf{odd} \to \mathsf{even} \end{array}$$

Now we have, for example,

$$\begin{array}{l} \vdash \lambda x^\iota.\, \mathsf{s}(\mathsf{s}\, x)) \Leftarrow \mathsf{even} \to \mathsf{odd} \\ \vdash \lambda x^\iota.\, \mathsf{s}(\mathsf{s}\, x)) \Leftarrow \mathsf{odd} \to \mathsf{even} \end{array}$$

but we cannot combine them using universal quantification.

To increase the expressive power we introduce finitary polymorphism in the form of *intersection sorts*, $S_1 \wedge S_2$. Because we construct a Curry-style sort assignment system on top of Church-style typing, a single term M can only be assigned two sorts S_1 and S_2 if $M^\alpha : S_1^\alpha$ and $M^\alpha : S_2^\alpha$. Therefore, to form the intersection, both sorts must refine the same type $S_1^\alpha \wedge S_2^\alpha$, and the intersection will again refine α.

The rules to introduce and eliminate intersection are quite straightforward: a term M, which need not be canonical, has sort $S_1 \wedge S_2$ if and only if it has both sorts.

$$\frac{\Gamma \vdash M : S_1 \quad \Gamma \vdash M : S_2}{\Gamma \vdash M : S_1 \wedge S_2} \qquad \frac{\Gamma \vdash M : S_1 \wedge S_2}{\Gamma \vdash M : S_1} \qquad \frac{\Gamma \vdash M : S_1 \wedge S_2}{\Gamma \vdash M : S_2}$$

To integrate them into the canonical forms system, we have to think about the direction they should apply.

The first consideration is the availability of information. In the checking rules, we know the sort we are checking against, and we must make sure we also know the sort in the premises. This means that the intersection introduction rule on the left should be a checking rule. In the synthesis rules we may assume that the premise yields a type, but we must make

sure we can synthesize the type in the conclusion. This means that the two elimination rules on the right should be synthesis rules.

$$\frac{\Gamma \vdash N \Leftarrow S_1 \quad \Gamma \vdash N \Leftarrow S_2}{\Gamma \vdash N \Leftarrow S_1 \wedge S_2} \qquad \frac{\Gamma \vdash R \Rightarrow S_1 \wedge S_2}{\Gamma \vdash R \Rightarrow S_1} \qquad \frac{\Gamma \vdash R \Rightarrow S_1 \wedge S_2}{\Gamma \vdash R \Rightarrow S_2}$$

We can also think about it from the logical perspective: an introduction rule, read bottom-up, will decompose the formula (here: the sort $S_1 \wedge S_2$). It should therefore become a checking rule, which also decomposes the sort bottom-up. Conversely, an elimination rule, read top-down, will decompose a formula (here: the sort $S_1 \wedge S_2$). It should therefore become a synthesis rule which also decomposes the sort top-down.

We introduce one more sort, $\top$, which is the unit of intersection and can be thought of as a conjunction of zero conjuncts. The introduction rule (checking), therefore, has zero premises which must be satisfied. And instead of two eliminations rules (synthesis) we have none.

$$\frac{}{\Gamma \vdash N \Leftarrow \top} \qquad \text{No } \top \text{ elimination rule}$$

This rule may at first appear troublesome, but due to our two-layer construction it does not mean that every term is well-sorted, just those that are well-formed according to the typing discipline. So $\top$ is just an uninformative property of well-formed terms.

We consolidate our language of sorts into a definition, extending the previous definition of simple sorts.

DEFINITION 13 (Sorts). We define sorts inductively with respect to a signature Σ.

1. A base sort Q^ι declared in Σ is a sort.
2. If S^α and T^β are sorts, then $(S \to T)^{\alpha \to \beta}$ is a sort.
3. If S^α and T^α are sorts then $(S \wedge T)^\alpha$ is a sort.
4. $\top^\alpha$ is a sort for each type α.

The complete rules for sort assignment can be found in Figure 1. Now we can verify that, indeed, at least for canonical forms sort checking is decidable.

THEOREM 14 (Decidability of sort checking). *Assume a signature Σ with sort declarations is fixed.*

Subsorting $Q' \leq Q$.

$$\frac{}{Q \leq Q} \qquad \frac{Q_1 \leq Q_2 \quad Q_2 \leq Q_3}{Q_1 \leq Q_3} \qquad \frac{Q' \leq Q \in \Sigma}{Q' \leq Q}$$

Sort checking $\Gamma \vdash N^\alpha \Leftarrow S^\alpha$ and
sort synthesis $\Gamma \vdash R^\alpha \Rightarrow T^\alpha$.

$$\frac{x{:}S \in \Gamma}{\Gamma \vdash x \Rightarrow S} \qquad \frac{c{:}S \in \Sigma}{\Gamma \vdash c \Rightarrow S}$$

$$\frac{\Gamma, x{:}S \vdash N \Leftarrow T \quad (x \notin \mathrm{dv}(\Gamma))}{\Gamma \vdash \lambda x.\, N \Leftarrow S \to T} \qquad \frac{\Gamma \vdash R \Rightarrow S \to T \quad \Gamma \vdash N \Leftarrow S}{\Gamma \vdash R\, N \Rightarrow T}$$

$$\frac{\Gamma \vdash R \Rightarrow Q' \quad Q' \leq Q}{\Gamma \vdash R \Leftarrow Q}$$

$$\frac{\Gamma \vdash N \Leftarrow S_1 \quad \Gamma \vdash N \Leftarrow S_2}{\Gamma \vdash N \Leftarrow S_1 \wedge S_2} \qquad \frac{\Gamma \vdash R \Rightarrow S_1 \wedge S_2}{\Gamma \vdash R \Rightarrow S_1} \qquad \frac{\Gamma \vdash R \Rightarrow S_1 \wedge S_2}{\Gamma \vdash R \Rightarrow S_2}$$

$$\frac{}{\Gamma \vdash N \Leftarrow \top} \qquad \text{No } \top \text{ elimination rule}$$

Figure 1. Sort assignment rules for canonical and atomic terms

1. *It is decidable whether* $Q' \leq Q$.
2. *It is decidable whether* $\Gamma \vdash N \Leftarrow S$.
3. *There is a finite number of sorts* T *such that* $\Gamma \vdash R \Rightarrow T$.

We assume that all judgments are well-formed in the manner explained before. For example N *must be of some type* α *and* S *must refine* α.

Proof. Decidability of subsorting follows since it is just the reflexive and transitive closure of the given subsort declarations.

Parts (2) and (3) follow by nested induction, first on the terms N and R. Second, (2) may appeal to (3) with the same term, but (3) may appeal to (2) only on a strictly smaller term. Finally within each part we may break down the structure of S and T if the term stays the same.

Case(2) $S = \top$. Then $\Gamma \vdash N \Leftarrow \top$ by rule.

Case(2) $S = S_1 \wedge S_2$. Then $\Gamma \vdash N \Leftarrow S_1 \wedge S_2$ iff $\Gamma \vdash N \Leftarrow S_1$ and $\Gamma \vdash N \Leftarrow S_2$ by inversion. Both of these are decidable by induction hypothesis (2) on N and S_1 and S_2, respectively, and so $\Gamma \vdash N \Leftarrow S_1 \wedge S_2$ is decidable.

Case(2) $S = S_1 \rightarrow S_2$. Then $\Gamma \vdash N \Leftarrow S_1 \rightarrow S_2$ iff $N = \lambda x_1.\, N_2$ and $\Gamma, x_1{:}S_1 \vdash N_2 \Leftarrow S_2$. This is decidable by induction hypothesis on N_2 and S_2.

Case(2) $S = Q$. Then $\Gamma \vdash N \Leftarrow Q$ iff $N = R$ and there exists a Q' such that $\Gamma \vdash R \Rightarrow Q'$ and $Q' \leq Q$. By induction hypothesis (3) on R, there is a finite set of Q' such that $\Gamma \vdash R \Rightarrow Q'$. By part (1) we can test each one in turn to see if at least one of them is a subsort of Q or not.

Case(3) $R = x$. Then there is a unique T such that $x{:}T \in \Gamma$, so we start with $\Gamma \vdash x \Rightarrow T$. We can saturate the set of judgments $\Gamma \vdash x \Rightarrow T'$ by applying the two intersection elimination rules, the only ones that do not change x. Since there are only finitely many conjuncts in T, this will terminate with a finite set.

Case(3) $R = c$. As in the case for variables, except we may already start with a finite set of T with $\Gamma \vdash c \Rightarrow T_i$ since there may be finitely many declarations for c.

Case(3) $R = R_1\, N_2$. By induction hypothesis on R_1, there are finitely many T' such that $\Gamma \vdash R_1 \Rightarrow T'$. For each T' of the form $T_2 \rightarrow T_1$

for some T_2 and T_1 we can test, by induction hypothesis, whether $\Gamma \vdash N_2 \Leftarrow T_2$. Then for each matching T_1 we get $\Gamma \vdash R_1\, N_2 \Rightarrow T_1$. We complete the set of all sorts synthesized by $R_1\, N_2$ by saturating with respect to the two intersection elimination rules.

■

10 Sort Preservation

The decidability of sort checking is a useful property for an implementation, but we should verify that the system is well-constructed. On canonical forms, this means the preservation of sorts under hereditary substitution and iterated expansions, which are generalizations of the same properties for types. We constructed the rules systematically, which makes the proof of these preservation properties quite straightforward.

We need some properties of typing derivations, specifically that the order of the typing assumptions does not matter (*order irrelevance*), and that we can adjoin additional unused assumptions (*weakening*) without changing the structure of the typing derivation. We elide the straightforward formal statement and proof of these properties. Recall that we assume that all judgments are well-formed with respect to the underlying layer of types.

THEOREM 15 (Sort preservation under hereditary substitution).
Assume $\Gamma \vdash M \Leftarrow S$.

1. *If* $\Gamma, x{:}S \vdash N \Leftarrow T$ *then* $\Gamma \vdash [M/x]^n(N) \Leftarrow T$.
2. *If* $\Gamma, x{:}S \vdash R \Rightarrow T$ *and* $\mathrm{hd}(R) \neq x$ *then* $\Gamma \vdash [M/x]^{rr}(R) \Rightarrow T$.
3. *If* $\Gamma, x{:}S \vdash R \Rightarrow T$ *and* $\mathrm{hd}(R) = x$ *then* $\Gamma \vdash [M/x]^{rn}(R) \Leftarrow T$.

Proof.

There are two forms of induction we can use for this property. One is a nested induction on the structure of the type α of M^α first, and the structure of the given sort derivations second. This is very similar to the proof given for composition of substitutions earlier.

An alternative is to use a nested induction, first over the structure of the computation of $[M/x]^n(N)$, $[M/x]^{rr}(R)$, and $[M/x]^{rn}(R)$, and second on the sort derivation $\mathcal{D}$ for N and R. For this, we consider the equations defining hereditary substitutions as rules of computation, reading them left to right, where we can apply the induction hypothesis to any subcomputation. Since hereditary substitution is well-founded, this is a well-founded form of induction.

We use this alternative method to illustrate it. When we write "*by i.h. on* $[M/x]^n(N)$" we mean the computation starting at this term, which should be a subcomputation of the given one.

Case(1): $[M/x]^n(N)$ is arbitrary and

$$\mathcal{D} = \frac{\begin{array}{cc}\mathcal{D}_1 & \mathcal{D}_2 \\ \Gamma, x{:}S \vdash N \Leftarrow T_1 & \Gamma, x{:}S \vdash N \Leftarrow T_2\end{array}}{\Gamma, x{:}S \vdash N \Leftarrow T_1 \wedge T_2}$$

$\Gamma \vdash [M/x]^n(N) \Leftarrow T_1$ by i.h.(1) on $[M/x]^n(N)$ and $\mathcal{D}_1$
$\Gamma \vdash [M/x]^n(N) \Leftarrow T_2$ by i.h.(2) on $[M/x]^n(N)$ and $\mathcal{D}_2$
$\Gamma \vdash [M/x]^n(N) \Leftarrow T_1 \wedge T_2$ by rule

Case(1): $[M/x]^n(N)$ is arbitrary and

$$\mathcal{D} = \frac{}{\Gamma, x{:}S \vdash N \Leftarrow \top}$$

$\Gamma \vdash [M/x]^n(N) \Leftarrow \top$ by rule

Case(1): $[M/x]^n(\lambda y_2.\, N_1) = \lambda y_2.\, [M/x]^n(N_1)$ where $y_2 \neq x$ and $y_2 \notin \mathrm{fv}(M)$ and

$$\mathcal{D} = \frac{\begin{array}{c}\mathcal{D}_1 \\ \Gamma, x{:}S, y_2{:}T_2 \vdash N_1 \Leftarrow T_1\end{array}}{\Gamma, x{:}S \vdash \lambda y_2.\, N_1 \Leftarrow T_2 \to T_1}$$

$\Gamma, y_2{:}T_2, x{:}S \vdash N_1 \Leftarrow T_1$ by order irrelevance
$\Gamma, y_2{:}T_2 \vdash M \Leftarrow S$ by weakening
$\Gamma, y_2{:}T_2 \vdash [M/x]^n(N_1) \Leftarrow T_1$ by i.h.(1) on $[M/x]^n(N_1)$ and $\mathcal{D}_1$
$\Gamma \vdash \lambda y_2.\, [M/x]^n(N_1) \Leftarrow T_2 \to T_1$ by rule

Case(1): $[M/x]^n(R) = [M/x]^{rr}(R)$ with $\mathrm{hd}(R) \neq x$ and

$$\mathcal{D} = \frac{\begin{array}{cc}\mathcal{D}' & \mathcal{Q} \\ \Gamma, x{:}S \vdash R \Rightarrow Q' & Q' \leq Q\end{array}}{\Gamma, x{:}S \vdash R \Leftarrow Q}$$

$\Gamma \vdash [M/x]^{rr}(R) \Rightarrow Q'$ by i.h.(2) on $[M/x]^{rr}(R)$ and $\mathcal{D}'$
$\Gamma \vdash [M/x]^n(R) \Leftarrow Q$ by rule

Case(1): $[M/x]^n(R) = [M/x]^{rn}(R)$ with $\mathrm{hd}(R) = x$ and

$$\mathcal{D} = \dfrac{\begin{array}{cc}\mathcal{D}' & \mathcal{Q} \\ \Gamma, x{:}S \vdash R \Rightarrow Q' & Q' \leq Q\end{array}}{\Gamma, x{:}S \vdash R \Leftarrow Q}$$

$\Gamma \vdash [M/x]^{rn}(R) \Leftarrow Q'$	by i.h.(3) on $\mathcal{D}'$
$\Gamma \vdash [M/x]^{rn}(R) \Rightarrow Q''$ and $Q'' \leq Q'$	by inversion
$Q'' \leq Q$	by transitivity
$\Gamma \vdash [M/x]^{rn}(R) \Leftarrow Q$	by rule

Case(2): $[M/x]^{rr}(R)$ arbitrary and

$$\mathcal{D} = \dfrac{\begin{array}{c}\mathcal{D}_{12} \\ \Gamma, x{:}S \vdash R \Rightarrow T_1 \wedge T_2\end{array}}{\Gamma, x{:}S \vdash R \Rightarrow T_1}$$

$\Gamma \vdash [M/x]^{rr}(R) \Rightarrow T_1 \wedge T_2$	by i.h.(2) on $[M/x]^{rr}(R)$ and $\mathcal{D}_{12}$
$\Gamma \vdash [M/x]^{rr}(R) \Rightarrow T_1$	by rule

Case(2): $[M/x]^{rr}(R)$ is arbitrary and $\mathcal{D}$ ends in the second intersection elimination rule. Symmetric to the previous case.

Case(2): $[M/x]^{rr}(R_1\, N_2) = ([M/x]^{rr} R_1)\, ([M/x]^n N_2)$ and

$$\mathcal{D} = \dfrac{\begin{array}{cc}\mathcal{D}_1 & \mathcal{D}_2 \\ \Gamma, x{:}S \vdash R_1 \Rightarrow T_2 \to T_1 & \Gamma, x{:}S \vdash N_2 \Leftarrow T_2\end{array}}{\Gamma, x{:}S \vdash R_1\, N_2 \Rightarrow T_1}$$

$\Gamma \vdash [M/x]^{rr} R_1 \Rightarrow T_2 \to T_1$	By i.h.(2) on $[M/x]^{rr} R_1$ and $\mathcal{D}_1$
$\Gamma \vdash [M/x]^n N_2 \Leftarrow T_2$	By i.h.(1) on $[M/x]^n N_2$ and $\mathcal{D}_2$
$\Gamma \vdash ([M/x]^{rr} R_1)\, ([M/x]^n N_2) \Rightarrow T_1$	by rule

Case(2): $[M/x]^{rr}(y) = y$ with $y \neq x$ and

$$\mathcal{D} = \dfrac{y{:}T \in \Gamma, x{:}S}{\Gamma, x{:}S \vdash y \Rightarrow T}$$

$\Gamma \vdash y \Rightarrow T$	by rule

Case(2): $[M/x]^{rr}(c) = c$ and $\mathcal{D}$ ends in the sorting rules for constants. As in the previous case.

Case(3): $[M/x]^{rn}(R)$ is arbitrary and

$$\mathcal{D} = \frac{\begin{array}{c}\mathcal{D}_{12}\\ \Gamma, x{:}S \vdash R \Rightarrow T_1 \wedge T_2\end{array}}{\Gamma, x{:}S \vdash R \Rightarrow T_1}$$

$\Gamma \vdash [M/x]^{rn}(R) \Leftarrow T_1 \wedge T_2$ by i.h.(3) in $[M/x]^{rn}(R)$ and $\mathcal{D}_{12}$
$\Gamma \vdash [M/x]^{rn}(R) \Leftarrow T_1$ by inversion

Case(3): $[M/x]^{rn}(R)$ is arbitrary and $\mathcal{D}$ ends in the second intersection elimination rule. Symmetric to the previous case.

Case(3): $[M/x]^{rn}(R_1\, N_2) = [N_2'/y_2]^n(N_1)$ where $[M/x]^{rn}(R_1) = \lambda y_2.\, N_1$ and $[M/x]^n(N_2) = N_2'$ and

$$\mathcal{D} = \frac{\begin{array}{c}\mathcal{D}_1\\ \Gamma, x{:}S \vdash R_1 \Rightarrow T_2 \to T_1\end{array} \quad \begin{array}{c}\mathcal{D}_2\\ \Gamma, x{:}S \vdash N_2 \Leftarrow T_2\end{array}}{\Gamma, x{:}S \vdash R_1\, N_2 \Rightarrow T_1}$$

$\Gamma \vdash [M/x]^{rn}(R_1) \Leftarrow T_2 \to T_1$ by i.h.(3) on $[M/x]^{rn}(R_1)$ and $\mathcal{D}_1$
$\Gamma \vdash \lambda y_2.\, N_1 \Leftarrow T_2 \to T_1$ by equality
$\Gamma, y_2{:}T_2 \vdash N_1 \Leftarrow T_1$ by inversion
$\Gamma \vdash [M/x]^n(N_2) \Leftarrow T_2$ by i.h.(1) on $[M/x]^n(N_2)$ and $\mathcal{D}_2$
$\Gamma \vdash N_2' \Leftarrow T_2$ by equality
$\Gamma \vdash [N_2'/y_2]^n(N_1) \Leftarrow T_1$ by i.h.(1) on $[N_2'/y_2]^n(N_1)$

Case(3): $[M/x]^{rn}(x) = M$ and

$$\mathcal{D} = \frac{x{:}S \in (\Gamma, x{:}S)}{\Gamma, x{:}S \vdash x \Rightarrow S}$$

$\Gamma \vdash M \Leftarrow S$ by assumption

■

A second property is sort preservation under iterated expansion. In some sense this is dual to the previous property: hereditary substitution allows us to replace a hypothesis $x{:}S$ which should be read as $x{\Rightarrow}S$ with a term $M \Leftarrow S$ and thus provides a way to go from checking to synthesis. Conversely, iterated expansion takes us from $x \Rightarrow S$ to $\eta^\alpha(x) \Leftarrow S$, that is, from synthesis to checking.

THEOREM 16 (Sort preservation under iterated expansion).
If $\Gamma \vdash R^\alpha \Rightarrow T$ *then* $\Gamma \vdash \eta^\alpha(R) \Leftarrow T$.

Proof. By induction on the structure of T.

Case: $T = T_1 \wedge T_2$. Then

$\Gamma \vdash R \Rightarrow T_1 \wedge T_2$	assumption
$\Gamma \vdash R \Rightarrow T_1$	by rule
$\Gamma \vdash \eta^{\alpha}(R) \Leftarrow T_1$	by i.h. on T_1
$\Gamma \vdash R \Rightarrow T_2$	by rule
$\Gamma \vdash \eta^{\alpha}(R) \Leftarrow T_2$	by i.h. on T_2
$\Gamma \vdash \eta^{\alpha}(R) \Leftarrow T_1 \wedge T_2$	by rule

Case: $T = \top$. Then

$\Gamma \vdash \eta^{\alpha}(R) \Leftarrow \top$	by rule

Case: $T = T_2 \to T_1$ and $\alpha = \alpha_2 \to \alpha_1$.

Let x_2 be a variable not in $\mathrm{fv}(R)$.	
$\Gamma, x_2{:}T_2 \vdash x_2 \Rightarrow T_2$	by rule
$\Gamma, x_2{:}T_2 \vdash \eta^{\alpha_2}(x_2) \Leftarrow T_2$	by i.h. on T_2
$\Gamma, x_2{:}T_2 \vdash R \Rightarrow T_2 \to T_1$	by weakening
$\Gamma, x_2{:}T_2 \vdash R\,\eta^{\alpha_2}(x_2) \Rightarrow T_1$	by rule
$\Gamma, x_2{:}T_2 \vdash \eta^{\alpha_1}(R\,\eta^{\alpha_2}(x_2)) \Leftarrow T_1$	by i.h. on T_1
$\Gamma \vdash \lambda x_2.\, \eta^{\alpha_1}(R\,\eta^{\alpha_2}(x_2)) \Leftarrow T_2 \to T_2$	by rule

Case: $T = Q$ and $\alpha = \iota$.

$\Gamma \vdash R \Rightarrow Q$	assumption
$Q \leq Q$	by reflexivity
$\Gamma \vdash R \Leftarrow Q$	by rule

■

11 Sort Preservation under Converse Substitution

One of the surprising properties of intersection types that was realized early on in their development [2] is that the converse of β-reduction preserves types. This means if we can type a normal form, we can type every term that reduces to it. Here, in the setting of canonical forms, we don't have redexes, but we can ask a corresponding question about the converse of hereditary substitution. We will show in this section that the converse of hereditary substitution preserves sorts.

We first prove a few useful lemmas.

LEMMA 17 (Strengthening).

1. *If* $\Gamma, x{:}S \vdash N \Leftarrow T$ *and* $x \notin \mathrm{fv}(N)$ *then* $\Gamma \vdash N \Leftarrow T$.
2. *If* $\Gamma, x{:}S \vdash R \Rightarrow T$ *and* $x \notin \mathrm{fv}(R)$ *then* $\Gamma \vdash R \Rightarrow T$.

Proof. By straightforward mutual induction on N and R. ■

We write $[R^\alpha/x^\alpha]N$ and $[R^\alpha/x^\alpha]R'$ for the *direct substitution* of R for x in N and R', which should be capture-avoiding as usual. This substitution is *not* hereditary. Intuitively, this is because we are replacing a variable, which is atomic, with an atomic term of the same sort.

An easy induction verifies that direct substitution preserves types and, once we know this, also sorts.

LEMMA 18 (Sort preservation under direct substitution).
Assume $\Gamma \vdash R \Rightarrow S$.

1. *If* $\Gamma, x{:}S \vdash N \Leftarrow T$ *then* $\Gamma \vdash [R/x]N \Leftarrow T$.
2. *If* $\Gamma, x{:}S \vdash R' \Rightarrow T$ *then* $\Gamma \vdash [R/x]R' \Rightarrow T$

Proof. By straightforward mutual induction on the typing derivations for N and R'. ■

LEMMA 19 (Hypothesis intersection).

1. *If* $\Gamma, x{:}S_1 \vdash N \Leftarrow T$ *then* $\Gamma, x{:}S_1 \wedge S_2 \vdash N \Leftarrow T$.
2. *If* $\Gamma, x{:}S_2 \vdash N \Leftarrow T$ *then* $\Gamma, x{:}S_1 \wedge S_2 \vdash N \Leftarrow T$.
3. *If* $\Gamma, x{:}S_1 \vdash R \Rightarrow T$ *then* $\Gamma, x{:}S_1 \wedge S_2 \vdash R \Rightarrow T$.
4. *If* $\Gamma, x{:}S_2 \vdash R \Rightarrow T$ *then* $\Gamma, x{:}S_1 \wedge S_2 \vdash R \Rightarrow T$.

Proof. We show the proof of part (1). The others are analogous.

$\Gamma, x{:}S_1 \vdash N \Leftarrow T$	Given
$\Gamma, y{:}S_1 \wedge S_2, x{:}S_1 \vdash N \Leftarrow T$	by weakening
$\Gamma, y{:}S_1 \wedge S_2 \vdash y \Rightarrow S_1 \wedge S_2$	by rule
$\Gamma, y{:}S_1 \wedge S_2 \vdash y \Rightarrow S_1$	by rule
$\Gamma, y{:}S_1 \wedge S_2 \vdash [y/x]N \Leftarrow T$	by direct substitution
$\Gamma, x{:}S_1 \wedge S_2 \vdash N \Leftarrow T$	by renaming

■

THEOREM 20 (Sort preservation under converse hereditary substitution).

1. *If* $\Gamma \vdash [M/x]^n(N) \Leftarrow T$ *then there exists an* S *such that* $\Gamma \vdash M \Leftarrow S$ *and* $\Gamma, x{:}S \vdash N \Leftarrow T$.

2. *If* $\Gamma \vdash [M/x]^{rr}(R) \Rightarrow T$ *then there exists an* S *such that* $\Gamma \vdash M \Leftarrow S$ *and* $\Gamma, x{:}S \vdash R \Rightarrow T$.

3. *If* $\Gamma \vdash [M/x]^{rn}(R) \Leftarrow T$ *then there exists an* S *such that* $\Gamma \vdash M \Leftarrow S$ *and* $\Gamma, x{:}S \vdash R \Rightarrow T$.

Proof. Again, we have a choice: we can prove this by a nested induction on a type and a sorting derivation, or we can use computation induction on the definition of hereditary substitution. Choosing the latter, the proof is by nested induction, first on the computation of $[M/x]^n(N)$, $[M/x]^{rr}(R)$ or $[M/x]^{rn}(R)$ and second on the given typing derivation $\mathcal{D}$.

Case(1): $[M/x]^n(N)$ is arbitrary and

$$\mathcal{D} = \frac{\begin{array}{cc}\mathcal{D}_1 & \mathcal{D}_2 \\ \Gamma \vdash [M/x]^n(N) \Leftarrow T_1 & \Gamma \vdash [M/x]^n(N) \Leftarrow T_2\end{array}}{\Gamma \vdash [M/x]^n(N) \Leftarrow T_1 \wedge T_2}$$

There exists S_1 such that	
$\Gamma \vdash M \Leftarrow S_1$ and	
$\Gamma, x{:}S_1 \vdash N \Leftarrow T_1$	by i.h.(1) on $[M/x]^n(N)$ and $\mathcal{D}_1$
There exists S_2 such that	
$\Gamma \vdash M \Leftarrow S_2$ and	
$\Gamma, x{:}S_2 \vdash N \Leftarrow T_2$	by i.h.(1) on $[M/x]^n(N)$ and $\mathcal{D}_2$
$\Gamma, x{:}S_1 \wedge S_2 \vdash N \Leftarrow T_1$	by hypothesis intersection
$\Gamma, x{:}S_1 \wedge S_2 \vdash N \Leftarrow T_2$	by hypothesis intersection
$\Gamma, x{:}S_1 \wedge S_1 \vdash N \Leftarrow T_1 \wedge T_2$	by rule
$\Gamma \vdash M \Leftarrow S_1 \wedge S_2$	by rule

Case(1): $[M/x]^n(N)$ is arbitrary and

$$\mathcal{D} = \frac{}{\Gamma \vdash [M/x]^n(N) \Leftarrow \top}$$

$\Gamma \vdash M \Leftarrow \top$	by rule
$\Gamma, x{:}\top \vdash N \Leftarrow \top$	by rule

Case(1): $[M/x]^n(\lambda y_2.\, N_1) = \lambda y_2.\, [M/x]^n(N_1)$ where $y_2 \neq x$ and $y_2 \notin \mathrm{fv}(M)$ and

$$\mathcal{D} = \frac{\begin{array}{c}\mathcal{D}_1 \\ \Gamma, y_2{:}T_2 \vdash [M/x]^n(N_1) \Leftarrow T_1\end{array}}{\Gamma \vdash \lambda y_2.\, [M/x]^n(N_1) \Leftarrow T_2 \to T_1}$$

There exists S such that
$\Gamma, y_2{:}T_2 \vdash M \Leftarrow S$ and
$\Gamma, y_2{:}T_2, x{:}S \vdash N_1 \Leftarrow T_1$ by i.h.(1) on $[M/x]^n(N_1)$ and $\mathcal{D}_1$
$\Gamma, x{:}S, y_2{:}T_2 \vdash N_1 \Leftarrow T_1$ by order irrelevance
$\Gamma, x{:}S \vdash \lambda y_2{:}T_2.\, N_1 \Leftarrow T_2 \to T_1$ by rule
$\Gamma \vdash M \Leftarrow S$ by strengthening since $y_2 \notin \mathrm{fv}(M)$

Case(1): $[M/x]^n(R) = [M/x]^{rr}(R)$ where $\mathrm{hd}(R) \neq x$ and

$$\mathcal{D} = \frac{\begin{array}{c}\mathcal{D}' \\ \Gamma \vdash [M/x]^{rr} R \Rightarrow Q' \quad Q' \leq Q\end{array}}{\Gamma \vdash [M/x]^{rr} R \Leftarrow Q}$$

There exists S such that
$\Gamma \vdash M \Leftarrow S$ and
$\Gamma, x{:}S \vdash R \Rightarrow Q'$ by i.h.(2) on $[M/x]^{rr}(R)$ and $\mathcal{D}'$
$\Gamma, x{:}S \vdash R \Leftarrow Q$ by rule

Case(1): $[M/x]^n(R) = [M/x]^{rn}(R)$ where $\mathrm{hd}(R) = x$ and

$$\mathcal{D} = \frac{\begin{array}{c}\mathcal{D}' \\ \Gamma \vdash [M/x]^{rn}(R) \Rightarrow Q' \quad Q' \leq Q\end{array}}{\Gamma \vdash [M/x]^{rn}(R) \Leftarrow Q}$$

There exists S such that
$\Gamma \vdash M \Leftarrow S$ and
$\Gamma, x{:}S \vdash R \Rightarrow Q'$ by i.h.(3) on $[M/x]^{rn}(R)$ and $\mathcal{D}'$
$\Gamma, x{:}S \vdash R \Leftarrow Q$ by rule

Case(2): $[M/x]^{rr}(R)$ is arbitrary and

$$\mathcal{D} = \frac{\begin{array}{c}\mathcal{D}_{12} \\ \Gamma \vdash [M/x]^{rr}(R) \Rightarrow T_1 \wedge T_2\end{array}}{\Gamma \vdash [M/x]^{rr}(R) \Rightarrow T_1}$$

There exists S such that
$\Gamma \vdash M \Leftarrow S$ and
$\Gamma, x{:}S \vdash R \Rightarrow T_1 \wedge T_2$ by i.h.(2) on $\mathcal{D}_{12}$
$\Gamma, x{:}S \vdash R \Rightarrow T_1$ by rule

Case(2): $[M/x]^{rr}(R)$ is arbitrary $\mathcal{D}$ ends in the second intersection elimination. Symmetric to the previous case.

Case(2): $[M/x]^{rr}(R_1\, N_2) = [M/x]^{rr}(R_1)\, [M/x]^{n}(N_2)$ and

$$\mathcal{D} = \dfrac{\begin{array}{cc}\mathcal{D}_1 & \mathcal{D}_2 \\ \Gamma \vdash [M/x]^{rr}(R_1) \Rightarrow T_2 \to T_1 & \Gamma \vdash [M/x]^{n}(N_2) \Leftarrow T_2\end{array}}{\Gamma \vdash [M/x]^{rr}(R_1)\, [M/x]^{n}(N_2) \Rightarrow T_1}$$

There exists S_1 such that
$\Gamma \vdash M \Leftarrow S_1$ and
$\Gamma, x{:}S_1 \vdash R_1 \Rightarrow T_2 \to T_1$ by i.h.(2) on $[M/x]^{rr}(R_1)$ and $\mathcal{D}_1$
There exists S_2 such that
$\Gamma \vdash M \Leftarrow S_2$ and
$\Gamma, x{:}S_2 \vdash N_2 \Leftarrow T_2$ by i.h.(1) on $[M/x]^{n}(N_2)$ and $\mathcal{D}_2$
$\Gamma \vdash M \Leftarrow S_1 \wedge S_2$ by rule
$\Gamma, x{:}S_1 \wedge S_2 \vdash R_1 \Rightarrow T_2 \to T_1$ by hypothesis intersection
$\Gamma, x{:}S_1 \wedge S_2 \vdash N_2 \Leftarrow T_2$ by hypothesis intersection
$\Gamma, x{:}S_1 \wedge S_2 \vdash R_1\, N_2 \Rightarrow T_1$ by rule

Case(2): $[M/x]^{rr}(y) = y$ for $y \neq x$ and

$$\mathcal{D} = \dfrac{y{:}T \in \Gamma}{\Gamma \vdash y \Rightarrow T}$$

$\Gamma \vdash M \Leftarrow \top$ by rule
$\Gamma, x{:}\top \vdash y \Rightarrow T$ by rule

Case(2): $[M/x]^{rr}(c) = c$ and $\mathcal{D}$ ends in the rule for constants. Proceed as in the previous case.

Case(3): $[M/x]^{rn}(R_1\, N_2) = [N_2'/y_2]^{n}(N_1)$ where $[M/x]^{rn}(R_1) = \lambda y_2.\, N_1$ and $[M/x]^{n}(N_2) = N_2'$ and

$$\begin{array}{c}\mathcal{D} \\ \Gamma \vdash [N_2'/y_2]^{n}(N_1) \Leftarrow T\end{array}$$

is arbitrary.

There exists S_2 such that
$\Gamma \vdash N_2' \Leftarrow S_2$ and
$\Gamma, y_2{:}S_2 \vdash N_1 \Leftarrow T$ by i.h.(1) on $[N_2'/y_2]^n(N_1)$ and $\mathcal{D}$
$\Gamma \vdash [M/x]^n(N_2) \Leftarrow S_2$ by equality
There exists S such that
$\Gamma \vdash M \Leftarrow S$ and
$\Gamma, x{:}S \vdash N_2 \Leftarrow S_2$ by i.h.(1) on $[M/x]^n(N_2)$
$\Gamma \vdash \lambda y_2.\, N_1 \Leftarrow S_2 \to T$ by rule
$\Gamma \vdash [M/x]^{rn}(R_1) \Leftarrow S_2 \to T$ by equality
There exists S' such that
$\Gamma \vdash M \Leftarrow S'$ and
$\Gamma, x{:}S' \vdash R_1 \Rightarrow S_2 \to T$ by i.h.(3) on $[M/x]^{rn}(R_1)$
$\Gamma \vdash M \Leftarrow S \wedge S'$ by rule
$\Gamma, x{:}S \wedge S' \vdash R_1 \Rightarrow S_2 \to T$ by hypothesis intersection
$\Gamma, x{:}S \wedge S' \vdash N_2 \Leftarrow S_2$ by hypothesis intersection
$\Gamma, x{:}S \wedge S' \vdash R_1\, N_2 \Rightarrow T$ by rule

Case(3): $[M/x]^{rn}(x) = M$ and

$$\begin{array}{c}\mathcal{D}\\ \Gamma \vdash M \Leftarrow T\end{array}$$

is arbitrary.

$\Gamma \vdash M \Leftarrow T$ by $\mathcal{D}$
$\Gamma, x{:}T \vdash x \Rightarrow T$ by rule

■

12 Subsorting at Higher Types

Recall that iterated expansion preserves sorts: if $R^\alpha \Rightarrow S$ then $\eta^\alpha(R) \Leftarrow S$. Since the substitution property has a (perhaps surprising) converse, we might conjecture that the identity property also has a converse.

Alas, it is not the case that $\eta^\alpha(R) \Leftarrow S$ implies $R \Rightarrow S$. A simple counterexample is $x{:}\mathsf{even} \vdash x \Leftarrow \mathsf{nat}$, but $x{:}\mathsf{even} \not\vdash x \Rightarrow \mathsf{nat}$. Various slightly extended conjectures also fail: synthesis is inherently very weak when compared to checking.

Nevertheless, there are still other interesting developments to consider regarding iterated expansion and the identity theorem. So far, subsorting has been confined to base sorts and this has been sufficient, essentially because we only consider η-long terms. But we can introduce a derived notion of subsorting beyond base sorts and show that we have a system

which is both sound and complete with respect to other characterizations of what subtyping might be. Finally, although beyond the scope of the present paper, we can use it to complete the general system of sort assignment sketched earlier which does not rely on canonical forms, but has a general subsumption rule.

One desirable property would be that S is a subsort of T if and only if every term of sort S is also a term of sort T. This means subsorting is essentially inclusion among sets of terms. However, this quantifies over all canonical terms, and is thus not an a priori decidable condition and of limited immediate utility.

The basic idea on how to reduce subsorting to a property of sorting and iterated expansion is to check whether $x{:}S \vdash \eta^\alpha(x) \Leftarrow T$. If so, S should be a subsort of T because $\eta^\alpha(x)$ is the identity. Because sort-checking is decidable, this is an effective criterion.

DEFINITION 21 (Subsorting at higher types). For two sorts S^α and T^α we write $S \leq T$ iff $x{:}S \vdash \eta^\alpha(x) \Leftarrow T$.

We are justified in reusing the symbol $\leq$, because on base sorts the prior judgment $Q' \leq Q$ coincides with the extended one: the derivation

$$\dfrac{\dfrac{}{x{:}Q' \vdash x \Rightarrow Q'} \quad Q' \leq Q}{x{:}Q' \vdash \eta^\iota(x) \Leftarrow Q}$$

is entirely forced, noting that $\eta^\iota(x) = x$.

We can now show that subsorting via sort checking of identities is sound and complete with respect to inclusion among sets of canonical forms.

THEOREM 22 (Alternative characterization of subtyping).
$S \leq T$ if and only if forall all Γ and N, $\Gamma \vdash N \Leftarrow S$ implies $\Gamma \vdash N \Leftarrow T$.

Proof. Direct in each direction.

($\Longrightarrow$)

$S \leq T$	assumption
$\Gamma \vdash N \Leftarrow S$	assumption
$x{:}S \vdash \eta^\alpha(x) \Leftarrow T$	by defn. of $\leq$
$\Gamma, x{:}S \vdash \eta^\alpha(x) \Leftarrow T$	by weakening
$\Gamma \vdash [N/x]^n(\eta^\alpha(x)) \Leftarrow T$	by sort preservation under substitution
$\Gamma \vdash N \Leftarrow T$	by right identity

($\Longleftarrow$)

$x{:}S \vdash x \Rightarrow S$	by rule
$x{:}S \vdash \eta^\alpha(x) \Leftarrow S$	by sort preservation under expansion
$x{:}S \vdash \eta^\alpha(x) \Leftarrow T$	by assumption
$S \leq T$	by defn. of $\leq$

■

13 Conclusion

We have developed a calculus of canonical forms for a λ-calculus formulated in the style of Church [1] where every well-formed term intrinsically has a unique simple type. The critical ingredient is the operation of *hereditary substitution* which returns a canonical form when substituting a canonical form for a variable in a canonical form. Its counterpart is *iterated expansion* which returns a canonical form when given an atomic one. By an argument relying crucially on the intrinsic types, we could see that these operations are always properly defined. Moreover, hereditary substitutions can be composed with expected results and iterated expansion of variables returns both a left and right identity for hereditary substitutions.

As a second layer we defined a system in the style of Curry [3] where sorts are assigned to terms already known to be well-typed. Sorts thus stand for properties of terms, which is the essence of Curry's approach. An interesting possibility we explored was to introduce finitary polymorphism through which we can explicitly state multiple properties of a single term. We presented an elegant formulation of subsorting and intersection sorts, which can immediately be seen to be decidable. In addition, sorts are preserved under hereditary substitution and iterated expansion, which means that Curry-style sort assignment harmoniously coexists with Church-style typing. Interestingly, the presence of finitary intersection allows us to validate the converse of substitution. In practice this means that we can sort-check a term not in canonical form by converting it to one and sort-checking the result, although the corresponding theorem is beyond the scope of this paper. We cannot directly validate the converse of iterated expansion because sort synthesis of atomic terms is too weak. But we introduce a notion of subsorting based on iterated expansion which characterizes, precisely, inclusion among sets of canonical forms and is directly decidable.

A natural next step is to give a sort assignment system for well-formed terms that are not necessarily in canonical form and prove (a) normalization via hereditary substitution and iterated expansion, (b) that sorting can be decided by conversion to canonical form and application of the algorithms in this paper. We conjecture that this should be possible and relatively straightforward based on the presented results.

We have carried out a related analysis of canonical forms and sorts for a dependently typed λ-calculus [10] and for types for a modal λ-calculus with metavariables [12]. These results illustrate the robustness of the ideas presented here. *Refinement types* [7, 4, 5] for functional programs represent another line of development that shares many ideas with this paper, although there are significant technical differences between a call-by-value functional language with data types, recursion, and effects, and Church's simply-typed λ-calculus. A promising idea toward a synthesis and unification of these threads is to generalize canonical forms via focusing in polarized logic [14] which is closely related to the concurrent logical framework [13]. Perhaps it is no accident that the latter originated the idea of hereditary substitutions.

Acknowledgments. I would like to dedicate this paper to my advisor, Peter Andrews, who introduced me to Church's type theory and taught me the beauty, precision, and utility of mathematical logic. I am greatly indebted to him. I would also like to thank Kevin Watkins, who first discovered hereditary substitutions and patiently explained them to me, and William Lovas and Noam Zeilberger for many insightful discussions regarding canonical forms, subtyping, and intersection types. Chad Brown and William Lovas deserve special thanks for their careful proof-reading and corrections to an earlier draft of this paper. This work has been supported by NSF grants NSF-0702381 *Integrating Types and Verification* and CCR-0306313 *Formal Digital Library*.

BIBLIOGRAPHY

[1] Alonzo Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[2] Mario Coppo, Maria Dezani-Ciancaglini, and Betti Venneri. Functional Character of Solvable Terms. *Zeitschrift für mathematische Logic und Grundlagen der Mathematik*, 27:45–58, 1981.

[3] H. B. Curry. Functionality in Combinatory Logic. *Proceedings of the National Academy of Sciences, U.S.A.*, 20:584–590, 1934.

[4] Rowan Davies. *Practical Refinement-Type Checking*. PhD thesis, Carnegie Mellon University, May 2005. Available as Technical Report CMU-CS-05-110.

[5] Joshua Dunfield. *A Unified System of Type Refinements*. PhD thesis, Carnegie Mellon University, August 2007. Available as Technical Report CMU-CS-07-129.

[6] Joshua Dunfield and Frank Pfenning. Type Assignment for Intersections and Unions in Call-by-Value Languages. In A.D. Gordon, editor, *Proceedings of the 6th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'03)*, pages 250–266, Warsaw, Poland, April 2003. Springer-Verlag LNCS 2620.

[7] Tim Freeman and Frank Pfenning. Refinement Types for ML. In *Proceedings of the Symposium on Programming Language Design and Implementation (PLDI'91)*, pages 268–277, Toronto, Ontario, June 1991. ACM Press.

[8] Robert Harper, Furio Honsell, and Gordon Plotkin. A Framework for Defining Logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, January 1993.

[9] W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980. Hitherto unpublished note of 1969, rearranged, corrected, and annotated by Howard.

[10] William Lovas and Frank Pfenning. A Bidirectional Refinement Type System for LF. In B. Pientka and C. Schürmann, editors, *Proceedings of the Second International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice*, pages 113–128, Bremen, Germany, July 2007. Electronic Notes in Theoretical Computer Science (ENTCS), vol 196.

[11] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform Proofs as a Foundation for Logic Programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[12] Aleksandar Nanevski, Frank Pfenning, and Brigitte Pientka. Contextual Modal Type Theory. *Transactions on Computational Logic*, 2008. To appear.

[13] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A Concurrent Logical Framework I: Judgments and Properties. Technical Report CMU-CS-02-101, Department of Computer Science, Carnegie Mellon University, 2002. Revised May 2003.

[14] Noam Zeilberger. On the Unity of Duality. *Annals of Pure and Applied Logic*, 153(1–3):66–96, April 2008. Special issue on *Classical Logic and Computation*.