数理解析研究所講究録
1021 巻 1997 年 119-135



# EQUIVALENCES BETWEEN ILLATIVE COMBINATORY LOGICS AND PURE TYPE SYSTEMS

M. W. Bunder, W. J. M. Dekkers and J. H. Geuvers

## ABSTRACT

For each Pure Type System there is a closely related "corresponding" system of illative combinatory logic, which nevertheless differs from the Pure Type System in three important ways. Combinations of changes eliminating these differences lead to a number of intermediate systems. In this paper we give conditions under which pairs of these systems are equivalent. In some cases these equivalences have been proved, in some others these are, as yet, conjectures. We list the known Pure Type Systems that satisfy these conditions.

## 1. INTRODUCTION

Combinatory logic and lambda calculus were first introduced (in Curry [9] and Church [7] respectively) as part of a stronger ("illative") system designed as a foundation for logic and mathematics. Both illative systems however turned out to be inconsistent. Church in [8] then produced a typed version of pure lambda calculus, while illative combinatory logic used restricted versions of some of its rules as shown below.

More recently theories generalising typed lambda calculus were introduced by Martin-Löf [14], Girard [10], Coquand and Huet [6] and others. Barendregt set up a generalised framework for such systems and this was generalised a little more by Berardi [2] and Terlouw [15] to Pure Type Systems (PTSs). (For details see Barendregt [1], where these are referred to as Generalised Type Systems (GTSs).)

Pure Type Systems bear a striking resemblance to illative combinatory logics, such as those of Bunder [3] and [4], but, as we will show, there are major apparent differences. Some of these differences, we will show, are not essential, others will disappear under conditions which are valid for most PTSs.

## 2. PURE TYPE SYSTEMS

All PTSs have a set of **constants** $\mathcal{C}$ and a set of **variables** $V$. These allow the definition of a set of **pseudoterms** $\mathcal{T}$ given by:

$$\mathcal{T} = V \mid \mathcal{C} \mid \mathcal{T}\mathcal{T} \mid \lambda V{:}\mathcal{T}.\mathcal{T} \mid \Pi V{:}\mathcal{T}.\mathcal{T}.$$

Note that if $x \in V$ and $A, B \in \mathcal{T}$, $\Pi x{:}A.B$ is a generalised type expression in the sense of Martin-Löf, Girard, Coquand and Huet etc. In the special case where $x$ is not a free variable of $A$ or $B$, $\Pi x{:}A.B$ is well known to behave exactly as the $A \to B$ of simple type theory (See for example Barendregt [1]).

Expressions of the form $M : A$, where $M, A \in \mathcal{T}$, are called **statements**. $M : A$ can be interpreted informally as: the pseudoterm $M$ has type $A$.

A sequence of statements of the form $\langle x_1 : A_1, \ldots, x_n : A_n \rangle$ where $x_1, \ldots, x_n$ are distinct members of $V$ and $A_1, \ldots, A_n \in \mathcal{T}$ is called a **pseudocontext**. If $\Gamma$ is a pseudocontext and $M, A \in \mathcal{T}$ an expression of the form $\Gamma \vdash M : A$ is called a **judgement**. A pseudocontext $\Gamma$ will be called a **context** if for some $M, A \in \mathcal{T}$, $\Gamma \vdash M : A$ is derivable from the postulates given below.

A judgement $\Gamma \vdash M : A$ can be informally interpreted as: the pseudoterm $M$ has type $A$ in the pseudocontext $\Gamma$.

**Notation** Throughout this paper $c, c_1, c_2 \ldots$ will denote elements of $\mathcal{C}$, $x, y, z$, $x_1, x_2, \ldots$ variables, $A, B, \ldots M, N, \ldots$ elements of $\mathcal{T}$ and $s, s_1, s_2, \ldots$ elements of a set $\mathcal{S}$ of **sorts**, which for each PTS will form a subset of $\mathcal{C}$. We will use $\Gamma, \Gamma', \Gamma_1, \ldots$ for (pseudo) contexts and, later $\Delta, \Delta', \Delta_1 \ldots$ for sets of statements and also sets of pseudoterms. We will use $FV(M)$ for the set of free variables of $M$ and $FV(\Gamma)$ and $FV(\Delta)$ for the set of free variables of $\Gamma$ and $\Delta$.

### 2.1 Definition

A **Pure Type System** (PTS) is determined by a **specification** $\lambda S = (\mathcal{S}, \mathcal{A}, \mathcal{R})$ where $\mathcal{S}$ is the set of **sorts**, $\mathcal{A}$ a set of **axioms** i.e. statements of the form $c : s$ where $c \in \mathcal{C}$ and $s \in \mathcal{S}$, and $\mathcal{R}$ is a subset of $\mathcal{S} \times \mathcal{S} \times \mathcal{S}$.

Type derivation in the PTS $\lambda S$, written $\Gamma \vdash_{\lambda S} M : A$ (or just $\Gamma \vdash M : A$) is defined by the following axioms and rules:

(axioms) $\quad \langle\ \rangle \vdash c : s \qquad$ if $c : s \in \mathcal{A}$

(start rule) $\quad \dfrac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \qquad x \notin FV(\Gamma)$

(weakening rule) $\quad \dfrac{\Gamma \vdash M : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash M : B} \qquad x \notin FV(\Gamma)$

(product rule) $\quad \dfrac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (\Pi x{:}\, A.B) : s_3} \qquad$ if $(s_1, s_2, s_3) \in R$

(application rule) $\quad \dfrac{\Gamma \vdash F : (\Pi x{:}\, A \cdot B) \qquad \Gamma \vdash M : A}{\Gamma \vdash FM : B[x := M]}$

(abstraction rule) $\quad \dfrac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash (\Pi x{:}\, A \cdot B) : s}{\Gamma \vdash (\lambda x{:}\, A.M) : (\Pi x{:}\, A.B)}$

(conversion rule) $\quad \dfrac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s \qquad B =_\beta B'}{\Gamma \vdash A : B'}$

**Examples** The original type theory $\lambda \rightarrow$ of Church [8] has $\mathcal{C} = \mathcal{S} = \{*\}$, $\mathcal{A} = \{* : *\}$ and $\mathcal{R} = \{(*, *, *)\}$. The calculus of constructions of Coquand and Huet [7] has $\mathcal{C} = \mathcal{S} = \{*, \Box\}$, $\mathcal{A} = \{* : \Box\}$ and $\mathcal{R} = \mathcal{S} \times \mathcal{S} \times \mathcal{S}$.

## 3. ILLATIVE COMBINATORY LOGIC AND LAMBDA CALCULUS

Illative combinatory logic is normally based on the combinators **K** and **S** and the illative constant $\Xi$, using which, implication and quantification can be defined. Equivalently (see Hindley and Seldin [12] Chapters 15-17) we can use the language of lambda calculus, which is more convenient, as PTSs also use this language. So while we will talk about **ICLs** (systems of illative combinatory logic) we will be using equivalent **ILCs** (systems of illative lambda calculus).

For each ICL we will assume a set of constants $\mathcal{C}$ (including $\Xi$) and a set of variables $V$. Terms are then defined by:

$$Tm = V \mid \mathcal{C} \mid \lambda V.Tm \mid TmTm.$$

If $\Delta$ is a set of terms and $X$ is a term, an expression of the form $\Delta \vdash X$ is called a **judgement**. This is interpreted informally as: from $\Delta, X$ can be derived. In particular $\vdash XY$ is interpreted informally as $\vdash Y \in X$ as well as $\vdash X(Y)$, so that $X$ can be seen as both a class and a unary predicate. $\vdash \Xi XY$ is interpreted informally as $\vdash X \subseteq Y$ and also as $\vdash (\forall x \in X)Y(x)$, ie $\Xi$ is a subset relation or a restricted quantifier.

The following definitions allow us to relate ICLs to type theories.

### 3.1 Definition

$$F = \lambda xyz.\Xi x(\lambda w.y(zw))$$

$$G = \lambda xyz.\Xi x(\lambda w.yw(zw))$$

Using the above informal interpretations of application and of $\Xi XY$, we can interpret $FXYZ$ as $(\forall x \in X)(Z(x) \in Y)$, ie as $Z : X \to Y$. $GXYZ$ can be interpreted as $Z : (\Pi x{:}X.Yx)$.

Thus in this type-free extension of lambda calculus we can represent a form of type assignment.

The rules of inference for $\Pi$ in PTSs turn out to be similar to those for $G$ in ICLs, if $G$ is interpreted in the above way; however the application and abstraction rules for $\Xi$ are much simpler than those for $G$, so we will formulate ICLs using the former.

### 3.2 Definition

An illative combinatory logic (ICL) is determined by a specification $I = (\mathcal{S}, \mathcal{A}, \mathcal{R})$ where $\mathcal{S}$ is a set of **sorts**, $\mathcal{A}(\subseteq Tm)$ is a set of **axioms** and $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S} \times \mathcal{S}$.

**Derivations** in an ICL $I$, written as $\Delta \vdash_I X$ (or just $\Delta \vdash X$) are defined by the following axioms and rules:

(axioms) $\Delta \vdash X$ if $X \in \mathcal{A}$

(start rule) $\Delta \vdash X$ if $X \in \Delta$

(product rule) $$\frac{\Delta, Xx \vdash s_2 Y \qquad \Delta \vdash s_1 X}{\Delta \vdash s_3(GX(\lambda x.Y))}$$

where $x \notin FV(\Delta, X)$ and $(s_1, s_2, s_3) \in R$

($\Xi$ application rule) $$\frac{\Delta \vdash \Xi XY \qquad \Delta \vdash XU}{\Delta \vdash YU}$$

($\Xi$ abstraction rule 2) $$\frac{\Delta, Xx \vdash Y \qquad \Delta \vdash sX}{\Delta \vdash \Xi X(\lambda x \cdot Y)}$$

where $x \notin FV(\Delta, X)$ and $s \in S$

(conversion rule) $$\frac{\Delta =_\beta \Delta', \quad X =_\beta Y, \quad \Delta \vdash X}{\Delta' \vdash Y}$$

**Note 1** We define $=_\beta$ between sets as the symmetric, transitive closure of $\twoheadrightarrow_\beta$ where $\Delta \twoheadrightarrow_\beta \Delta'$ if for all $M \in \Delta$ there is an $M' \in \Delta'$ such that $M \twoheadrightarrow_\beta M'$ and for each $N \in \Delta'$ there is an $M \in \Delta$ such that $M \twoheadrightarrow_\beta N$

**Note 2** The names of the above rules (and the ones for $F$ and $G$), including particularly the "2" on the abstraction rules, are chosen to match those of the closest corresponding rules for the PTSs introduced below.

The application and abstraction rules for $F$ and $G$ given below follow from those for $\Xi$.

($F$ application rule) $$\frac{\Delta \vdash FXYZ \qquad \Delta \vdash XU}{\Delta \vdash Y(ZU)}$$

($F$ abstraction rule 2) $$\frac{\Delta, Xx \vdash YZ \qquad \Delta \vdash sX}{\Delta \vdash FXY(\lambda x.Z)}$$ where $x \notin FV(\Delta, XY), s \in \mathcal{S}$

($G$ application rule) $$\frac{\Delta \vdash GXYZ \qquad \Delta \vdash XU}{\Delta \vdash YU(ZU)}$$

($G$ abstraction rule 2) $$\frac{\Delta, Xx \vdash YZ \qquad \Delta \vdash sX}{\Delta \vdash GX(\lambda x.Y)(\lambda x.Z)}$$ where $x \notin FV(\Delta, X), s \in \mathcal{S}$

Note that if $\Xi XY$ was defined as $FXY(\lambda u.u)$ or $GX(\lambda uv.v)Y$, the $\Xi$ rules would follow from those for $F$ or $G$.

Note also if under our informal interpretation any $X$ could be interpreted by $X^i$

and $\Delta$ by $\Delta^i$, these $F$ rules could be interpreted as:

$$\frac{\Delta^i \vdash Z^i : X^i \to Y^i \qquad \Delta^i \vdash U^i : X^i}{\Delta^i \vdash Z^i U^i : Y^i}$$

$$\frac{\Delta^i, x : X^i . Z^i : Y^i \qquad \Delta^i \vdash X^i : s}{\Delta^i \vdash \lambda x . Z^i : X^i \to Y^i}.$$

With $\mathcal{S}$ representing the set of simple types, these are the postulates of simple type theory ($TA_\lambda$ of Hindley [12]). A similar translation is used, more formally, below for $\Pi$ and $G$.

## 4. SIMILARITIES AND DIFFERENCES BETWEEN PTSs AND ICLs

The main difference between PTSs and ICLs is obviously in the notation. This we can overcome with the following translation $*$ from $\mathcal{T}$ to $Tm$:

$$x^* = x \qquad \text{for } x \in V$$

$$(MN)^* = M^* N^*$$

$$c^* = c \qquad \text{for } c \in \mathcal{C}$$

$$(\Pi x{:}\, A.B)^* = GA^*(\lambda x.B^*)$$

$$(\lambda x : A.B)^* = \lambda x.B^*$$

$$(M : A)^* = A^* M^*$$

A problem we will face in Section 8 is: under what conditions does the translation $*$ of deductions have an inverse?

Given $*$, the product and application rules of PTSs translate into the product rule of ICLs and the $G$ application rule.

The abstraction rule of PTSs, however, translates into:

$$\frac{\Gamma^*, Ax \vdash BM \qquad \Gamma^* \vdash s(GA(\lambda x \cdot B))}{\Gamma^* \vdash GA(\lambda x.B)(\lambda x.M)}$$

A second difference therefore is that in the PTS abstraction rule the new type $GXY$ (or $\Pi x{:}\, X.Yx$) ), being formed, must be in a sort $s$, while in the ICL abstraction rule the type of the class $X$ over which we are abstracting must be in a sort $s$.

The remaining differences involve the use of **sequences** for **contexts** in PTSs and **sets** of **arbitrary terms** in ICLs and the seemingly more general conversion rule for ICLs.

For each PTS we will consider, in this paper, a version with an ICL-like start rule and sets as contexts as well as the stronger conversion rule. We will call these SPTSs.

For each PTS we will consider one with something like the ICL-abstraction rule and a more liberal conversion rule. These we will call APTSs. Also we will consider PTSs with both of these changes -SAPTSs.

In the next three sections we will outline results connecting PTSs, SPTSs, APTSs and SAPTSs, these are to appear in [5]. We will then list some anticipated results (partly conjectures at this stage) linking the $*$ translated versions of these systems - the $*$SAPTS being the ICLs corresponding to given PTSs - with each other and the four PTS systems. The connections between the systems will often depend on the specification of the PTS being considered.

## 5. SET PTSs

### 5.1 Definition

Given a PTS with specification $\lambda S$, the corresponding **Set PTS** (SPTS) has the axioms and the start, weakening and conversion rules replaced by:

(axioms) $\quad \Delta \vdash^S c : s \qquad$ where $c : s \in \mathcal{A}$

(start rule) $\quad \Delta \vdash^S M : A \qquad$ where $M : A \in \Delta$

(conversion rule) $\quad \dfrac{\Delta \vdash^S M : A \qquad \Delta =_\beta \Delta',\ M =_\beta M',\ A =_\beta A'}{\Delta' \vdash^S M' : A'}$

In each case $\Delta$ is an arbitrary finite set of statements. In all other rules $\Gamma$ is replaced by such a $\Delta$ and $\vdash$ by $\vdash^S_{\lambda S}$ or just $\vdash^S$.

It is clear that not every provable SPTS judgement has a valid PTS counterpart,

for example

$$s_1 : s_2 \vdash^S s_1 : s_2$$

but $s_1 : s_2$ cannot form a context as $s_1$ is not a variable.

The following class of sets $\Delta$ will correspond to the contexts of PTSs.

**5.2 Definition**

A set of statements $\Delta$ is **S-legal** in an SPTS with specification $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ if $\Delta \twoheadrightarrow_\beta \{x_1 : A_1, \ldots, x_n : A_n\}$ and:

(i) $(\forall i, j)(1 \leq i < j \leq n \rightarrow x_i \neq x_j)$

(ii) $(\forall i)(1 \leq i \leq n \rightarrow (\exists s \in \mathcal{S})(x_1 : A_1, \ldots, x_{i-1} : A_{i-1} \vdash^S A_i : s)$

(iii) $(\forall i)(1 \leq i \leq n \rightarrow x_i, \ldots, x_n \notin FV(A_i))$.

**Notation**

$M : A \twoheadrightarrow_\beta N : B$ iff $M \twoheadrightarrow_\beta N$ and $A \twoheadrightarrow_\beta B$.

**5.3 Definition**

If $\Gamma$ is a (pseudo-) context $S(\Gamma)$ is the corresponding set.

We then prove (in [5]):

**5.4 Theorem**

For a PTS and SPTS with the same specification:

(i) $\Gamma \vdash M : A \;\Rightarrow\; S(\Gamma)$ is $S$-legal & $S(\Gamma) \vdash^S M : A$

(ii) $\Delta$ is $S$-legal & $\Delta \vdash^S M : A$

$\Rightarrow\; (\exists \Gamma, M', A')\Delta \twoheadrightarrow_\beta S(\Gamma),\ M \twoheadrightarrow_\beta M',\ A \twoheadrightarrow_\beta A'\ \&\ \Gamma \vdash M' : A'$.

**Corollary**

For a PTS and SPTS with the same specification:

(i) $\vdash M : A \;\Rightarrow\; \vdash^S M : A$

(ii) $\vdash^S M : A \;\Rightarrow\; (\exists M')\ M \twoheadrightarrow_\beta M'\ \&\ \vdash M' : A$.

Thus the systems are "theorem equivalent", modulo $\beta$-equality.

## 6. ABSTRACTION-ALTERED PTSs

The difference between the PTS abstraction rule and one like the ICL version

proposed in Section 4:

(abstraction rule 2) $$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash A : s}{\Gamma \vdash (\lambda x : A \cdot M) : (\pi x : A.B)} \text{ where } s \in \mathcal{S}$$

turns out, for most PTSs, to be an essential one. To explain this we need a further definition.

**6.1 Definition**

$B \in \mathcal{T}$ is a **toptype** in a PTS with a given specification, if $(\exists M, \Gamma)$ $\Gamma \vdash M : B$ and $\sim (\exists A, \Gamma)\ \Gamma \vdash B : A$.

A sort $s$ is a **topsort** if it is a top type.

The PTS $\lambda C$ - the calculus of constructions - has $\Box$ as topsort. Using $\vdash * : \Box$, a $\lambda C$ axiom, and weakening we can prove:

$$x : * \vdash * : \Box$$

and by the abstraction rule 2 and the axiom we get:

$$\vdash (\lambda x{:}*.*) : (\Pi x{:}*.\Box)$$

Using the PTS abstraction rule we would need $\vdash (\Pi x{:}*.\Box) : s$ to obtain this and this, in turn, would require the product rule applied to

$$x : * \ \vdash\ \Box : s_2$$

and

$$\vdash\ * : s_1$$

where $(s_1,\ s_2,\ s) \in \mathcal{R}$.

As $\Box$ is a topsort the former of these is not available.

It is clear that for any PTS with a topsort abstraction rule 2 is not valid.

In contrast, for any PTS with abstraction rule 2, the PTS abstraction rule is valid. Abstraction rule 2 is therefore strictly stronger than the abstraction rule.

We now propose the following weaker version of abstraction rule 2, which for most of the PTSs in current use, allows us to prove an equivalence between PTSs and the altered PTSs (APTSs) with this rule:

(abstraction rule 3) $$\frac{\Gamma, x : A \vdash M : B \quad \Gamma \vdash A : s}{\Gamma \vdash (\lambda x : A \cdot M) : (\Pi x : A \cdot B)}$$

where $(\exists s_2, s_3)(s_1, s_2, s_3) \in \mathcal{R}$ & $(B \in_\beta \mathcal{C} \Rightarrow (B : s_2) \in \mathcal{A})$ (+)

While condition (+) may look a little complex for most PTSs it is equivalent to: For some subset $\mathcal{S}_1$ of $\mathcal{S}$, $B \notin \mathcal{S}_1$.

We now define APTSs.

**6.2 Definition**

Given a PTS with specification $\lambda S$ the corresponding **abstraction altered PTS (APTS)** has the abstraction rule replaced by abstraction rule 3 and the $\Gamma \vdash B' : s$ omitted from the conversion rule. Derivations are written as $\Gamma \vdash^A_{\lambda S} M : A$ (or just $\Gamma \vdash^A M : A$).

**6.3 Definition**

$$\mathcal{R}_1 = \{s_1 \in \mathcal{S} \mid (\exists s_2, s_3)(s_1, s_2, s_3) \in \mathcal{R}\}$$

$$\mathcal{S}_1 = \{s \in \mathcal{S} \mid (\exists A, \Gamma)\Gamma \vdash A : s\}$$

**6.4 Theorem**

For PTSs and an APTS with the same specification,

(i) $\Gamma \vdash M : B \Rightarrow \Gamma \vdash^A M : B.$

(ii) If the PTS satisfies:

$(\forall s \in \mathcal{S}_1)$ [$s$ is not a topsort $\vee$ $(\exists s_1, s_2)(s_1, s_2, s) \in \mathcal{R} \Rightarrow$

$(\forall s_1 \in \mathcal{S}_1 \cap \mathcal{R}_1)(\exists s_3)(s_1, s, s_3) \in \mathcal{R})]$, (*)

then if

$$\Gamma \vdash^A M : B$$

then there is a $C$ such that

$$\Gamma \vdash M : C.$$

In the theorem below $(s_1, s_2)$ as an element of $\mathcal{R}$ is short for $(s_1, s_2, s_2)$.

**6.5 Theorem**

The PTSs specified below satisfy $(*)$ and so are equivalent to the corresponding APTSs. In each case we note the restriction imposed by $(+)$ and give $\mathcal{S}_1$.

(i) $\lambda\!\rightarrow$: $\mathcal{S} = \{*, \Box\}$, $\mathcal{A} = \{* : \Box\}$, $\mathcal{R} = \{(*, *, *)\}$

Restriction: $B \not\in_\beta \{*, \Box\}$; $\mathcal{S}_1 = \{*\}$

(ii) $\lambda^\tau$: $\mathcal{S} = \{*\}$, $\mathcal{A} = \{0 : *\}$, $\mathcal{R} = \{(*, *)\}$

Restriction: $B \neq_\beta *$ ; $\mathcal{S}_1 = \{*\}$

(iii) $\lambda *$: $\mathcal{S} = \{*\}$, $\mathcal{A} = \{* : *\}$, $\mathcal{R} = \{(*, *)\}$

Restriction: (none); $\mathcal{S}_1 = \{*\}$

(iv) $\lambda 2$: $\mathcal{S} = \{*, \Box\}$, $\mathcal{A} = \{* : \Box\}$, $\mathcal{R} = \{(*, *), (\Box, *)\}$

Restriction: $B \not\in_\beta \{*, \Box\}$; $\mathcal{S}_1 = \{*, \Box\}$.

(v) $\lambda C = \lambda Pw$: $\mathcal{S} = \{*, \Box\}$, $\mathcal{A} = \{* : \Box\}$ , $\mathcal{R} = \{(*, *), \{*, \Box\}, (\Box, *), (\Box, \Box)\}$

Restriction: $B \neq_\beta \Box$; $\mathcal{S}_1 = \{*, \Box\}$

(vi) $\lambda P$: $S = \{*, \Box\}$, $\mathcal{A} = \{* : \Box\}$, $\mathcal{R} = \{(*, *), (*, \Box)\}$

Restriction: $B \neq_\beta \Box$ ; $\mathcal{S}_1 = \{*\}$.

(vii) $\lambda$ -AUT-68 : $\mathcal{S} = \{*, \Box, \Delta\}$, $\mathcal{A} = \{* : \Box\}$,

$\mathcal{R} = \{(*, *), (*, \Box, \Delta), (\Box, *, \Delta), (\Box, \Delta, \Delta), (\Box, \Box, \Delta), (*, \Delta, \Delta)\}$

Restriction: $B \not\in_\beta \{\Box, \Delta\}$; $\mathcal{S}_1 = \{*, \Box\}$

(viii) $\lambda$-AUT-QE : $\mathcal{S} = \{*, \Box, \Delta\}$, $\mathcal{A} = \{* : \Box\}$,

$\mathcal{R} = \{(*, *), (*, \Box)\ (\Box, *, \Delta), (\Box, \Box, \Delta), (*, \Delta), (\Box, \Delta)$

Restriction: $B \not\in_\beta \{\Box, \Delta\}$; $\mathcal{S}_1 = \{*, \Box\}$

(ix) $\lambda$-PAL : $\mathcal{S} = \{*, \Box, \Delta\}$, $\mathcal{A} = \{* : \Box\}$,

$\mathcal{R} = \{(*, *, \Delta), (*, \Box, \Delta), (\Box, *, \Delta), (\Box, \Box, \Delta), (*, \Delta), (\Box, \Delta)\}$

Restriction: $B \not\in_\beta \{\Box, \Delta\}$; $\mathcal{S}_1 = \{*, \Box\}$.

Of the above, $\lambda\!\rightarrow$ is the $\lambda$-calculus of Church [8], $\lambda C$ that of Coquand and Huet [7] and $\lambda$-AUT-68, $\lambda$-AUT-QE and $\lambda$-PAL are some of the de Bruin AUTOMATH systems.

Of the remaining PTSs mentioned in Barendregt [2] six do not satisfy $(*)$ and one is not singly sorted.

## 7. SET-ABSTRACTION ALTERED PTSs

### 7.1 Definition

Given a PTS with specification $(\mathcal{S},\ \mathcal{A},\ \mathcal{R})$, the corresponding set-abstraction altered PTS (SAPTS) with the same specification has the SPTS axioms, start and conversion rules and the APTS altered abstraction rule.

Define SA-legal similarly to S-legal. We have:

### 7.2 Theorem

For an APTS and an SAPTS with the same specification,

(i) $\Gamma \vdash^{A} M : A \ \Rightarrow\ S(\Gamma) \vdash^{SA} M : A$ & $S(\Gamma)$ is SA-legal

(ii) If $\Delta$ is SA-legal and $\Delta \vdash^{SA} M : A$, then

$$(\exists \Gamma, M')\ \Delta \twoheadrightarrow_{\beta} S(\Gamma),\ M \twoheadrightarrow_{\beta} M' \ \&\ \Gamma \vdash^{A} M' : A$$

### 7.3 Theorem

For an SPTS and an SAPTS with the same specification,

(i) If $\Delta$ is $S$-legal then:

$$\Delta \vdash^{S} M : A \ \Rightarrow\ \Delta \vdash^{SA} M : A$$

(ii) For a PTS which satisfies

$$(\forall s \in \mathcal{S})\ [\, s \text{ is not a topsort of } \vee\ (\exists s_1,\ s_2)(s_1,\ s_2,\ s) \in \mathcal{R} \ \Rightarrow\ (\forall s_1 \in \mathcal{S}_1 \cap \mathcal{R}_1)(\exists s_3)(s_1,\ s,\ s_3) \in \mathcal{R}\ ], \qquad (*)$$

if $\Delta$ is SA-legal then:

$$\Delta \vdash^{SA} M : A \ \Rightarrow\ \Delta \vdash^{S} M : A\ .$$

## 8. RICLs AND SAPTSs

The following is the ICL version of abstraction rule 3:

($\Xi$-abstraction rule 3) $$\frac{\Delta,\ Xx \vdash Y \quad \Delta \vdash sX}{\Delta \vdash \Xi X(\lambda x \cdot Y)}$$

where $x \notin FV(\Delta,\ X)$, $s \in \mathcal{S}$ and $(+)$ holds (with $s_2 B$ for $B : s_2$).

### 8.1 Definition

A **Restricted ICL** (RICL) is an ICL with $\Xi$ -abstraction rule 2 replaced by $\Xi$-abstraction rule 3. We will denote a derivation in an RICL with specification $I$ by $\Delta \vdash_I^R Z$ (or just $\Delta \vdash^R Z$).

Clearly we have

### 8.2 Theorem

For an ICL and an RICL with the same specification,

$$\Delta \vdash Z \Rightarrow \Delta \vdash^R Z$$

In a later paper we expect to prove:

### 8.3 Conjecture

For an SAPTS and an RICL with the same specification,

(i) $\Delta \vdash^{SA} M : A \Rightarrow \Delta^* \vdash^R A^* M^*$

(ii) $\Delta \vdash^R X \Rightarrow (\exists \Delta_1,\ M,\ A)\Delta_1^* \twoheadrightarrow_\beta \Delta,\ A^* M^* \twoheadrightarrow_\beta X\ \&\ \Delta_1 \vdash^{SA} M : A$.

## 9. VARIANTS OF RICLs

In the later paper mentioned above we will introduce for each RICL, a variant corresponding to an SPTS, an APTS and a PTS with the same specification.

### 9.1 Definition

An **abstraction modified ICL** (AICL) is an RICL with $\Xi$-abstraction rule 3 replaced by

$$\text{G-abstraction 1} \quad \frac{\Delta,\ Xx \vdash YZ \quad \Delta \vdash s(GX\lambda x \cdot Y)}{\Delta \vdash GX(\lambda x \cdot Y)(\lambda x \cdot Z)}$$

where $x \notin FV(\Delta, X)$.

In an AICL, $\vdash$ will be written as $\vdash^{AI}$.

We hope to prove:

**9.2 Conjecture**

For an SPTS and a AICL with the same specification:

(i) $\Delta \vdash^S M : A \Rightarrow \Delta^* \vdash^{AI} A^* M^*$.

(ii) $\Delta \vdash^{AI} X \Rightarrow (\exists \Delta_1, M, A)\Delta_1^* =_\beta X\Delta, A^*M^* =_\beta \ \& \ \Delta_1 \vdash^S M : A.$

**9.3 Definition**

A **context-modified ICL (CICL)** is an RICL with the sets $\Delta$ replaced by sequences $\Gamma$ in the rules, with the start and conversion rules altered as follows and with $\vdash^{CI}$ for $\vdash^R$:

(axioms) $\qquad \vdash^{CI} X \qquad$ if $X \in \mathcal{A}$

(start rule) $\qquad \dfrac{\Gamma \vdash^{CI} sX}{\Gamma, Xx \vdash^{CI} Xx} \qquad$ if $X \notin FV(\Gamma)$

(weakening rule) $\qquad \dfrac{\Gamma \vdash^{CI} X \quad \Gamma \vdash^{CI} sY}{\Gamma, Yx \vdash^{CI} X}$

(conversion rule) $\qquad \dfrac{\Gamma \vdash^{CI} XY \quad \Gamma \vdash^{CI} sX \quad x =_\beta Z}{\Gamma \vdash^{CI} ZY}$

Note that the $\Gamma$s above are sequences as for PTSs.

**9.4 Definition**

A **context and abstraction altered ICL** is a CICL with $G$-abstraction rule 1, with $\Gamma$ for $\Delta$.

**9.5 Conjecture**

For a CICL and an RICL with the same specification:

(i) $\Gamma \vdash^{CI} X \Rightarrow S(\Gamma) \vdash^R X$

(ii) If $\Delta$ is $R$-legal & $\Delta \vdash^R X \Rightarrow (\exists \Gamma, Y)\Delta \twoheadrightarrow_\beta S(\Gamma),\ X \twoheadrightarrow_\beta Y \&\ \Gamma \vdash^{CI} Y$

(Note $\Delta$ is $R$- (or $AI$-) legal is defined similarly to S-legal in Definition 5.2 .)

**9.6 Conjecture**

For a CAICL and a CICL with the same specification:

(i) $\Gamma \vdash^{CAI} X \Rightarrow \Gamma \vdash^{CI} X$

(ii) If $(*)$ holds:

$\Gamma \vdash^{CI} X \Rightarrow \Gamma \vdash^{CAI} X$ .

**9.7 Conjecture**

For a CAICL and a AICL with the same specification:

(i) $\Gamma \vdash^{CAI} X \Rightarrow S(\Gamma) \vdash^{AI} X$

(ii) If $\Delta$ is $AI$-legal,

$\Delta \vdash^{AI} X \Rightarrow (\exists \Gamma, Y)\ \Delta \twoheadrightarrow_\beta S(\Gamma), X \twoheadrightarrow_\beta Y\ \&\ \Gamma \vdash^{CAI} Y$ .

**9.8 Conjecture**

For a AICL and a RICL with the same specification:

(i) If $\Delta$ is $AI$-legal:

$\Delta \vdash^{AI} X \Rightarrow \Delta \vdash^{R} X$

(ii) If the system is singly sorted satisfies $(*)$ and $\Delta$ is $R$-legal:

$\Delta \vdash^{R} X \Rightarrow \Delta \vdash^{AI} X$ .

**9.9 Conjecture**

For an APTS and a CICL with the same specification:

(i) $\Gamma \vdash^{A} M : A \Rightarrow \Gamma^* \vdash^{CI} A^* M^*$

(ii) $\Gamma \vdash^{CI} X \Rightarrow (\exists \Gamma_1, M, A)\ \Gamma_1^* \equiv \Gamma, A^* M^* \equiv X\ \&\ \Gamma_1 \vdash M : A$ .

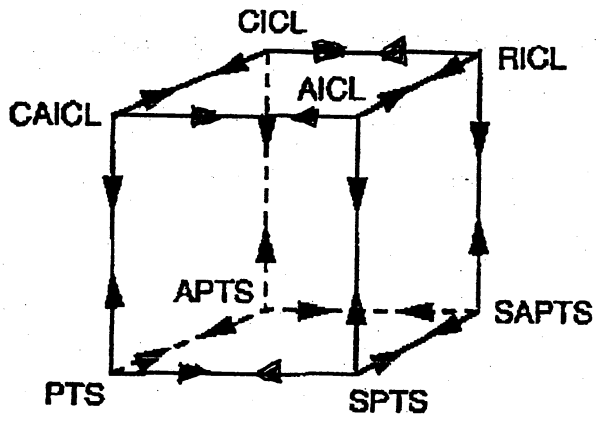
**9.10 Conjecture**

For a PTS and a CAICL with the same specification:

(i) $\Gamma \vdash M : A \Rightarrow \Gamma^* \vdash^{CAI} A^* M^*$

(ii) $\Gamma \vdash^{CAI} X \Rightarrow (\exists \Gamma_1, M, A)\ \Gamma^* \equiv \Gamma,\ A^* M^* \equiv X \& \Gamma_1 \vdash M : A$ .

For each PTS the relation between it, the corresponding ICL, with the same specification, and the intermediate systems can best be summarised in the cube below:




with each arrow showing an equivalence between systems, given certain conditions. Each $\rightarrow$ arrow has no condition, but $\Gamma$ is changed to $S(\Gamma)$. Each $\leftarrow$ requires $\Delta$ to be legal and certain $\beta$ reductions. Each $\nearrow$ requires legality of any $\Delta$ and each $\swarrow$ $(*)$ and legality of any $\Delta$. Each $\uparrow$ requires the $*$ translation and each $\downarrow$ reductions of $*$ translations.

We expect the proofs of the Conjectures 9.5 to 9.8 to be similar to those of Theorems 7.2, 6.4, 5.4 and 7.3 and the proofs of Conjectures 9.2, 9 .9 and 9.10 to be similar to the proof of Conjecture 8.3.

**REFERENCES**

[1] Barendregt, HP "Lambda calculus with types" in Handbook of Logic in Computer Science Vol II, S Abramsky, DM Gabbay and TSE Maibaum eds. Oxford University Press, 1992.

[2] Berardi, S "Towards a mathematical analysis of the Coquand-Huet calculus of constructions and the other systems of the Barendregt cube" Dept Computer Science, Carnegie Mellon University and Dipartimento di Matematica, Universita di Torino, 1988.

[3] Bunder, MW "Propositional and predicate calculuses based on combinatory logic" Notre Dame Journal of Formal Logic Vol XV (1974) pp 25-32.

[4] Bunder, MW "Conjunction without conditions in illative combinatory logic" The

Bulletin of the Section of Logic, Polish Academy of Sciences, Vol 13 (1984) pp207-212.

[5] Bunder, MW and Dekkers, WAM "Some alternative pure type systems", to appear.

[6] Coquand, T and Huet, G"The calculus of constructions" Information and Computation, Vol 76 (1988) pp95-120.

[7] Church, A "A set of postulates for the foundation of logic" Annals of Mathematics Vol 33 pp346-366 and Vol 34 pp839-864.

[8] Church, A "A formulation of the simple theory of types" Journal of Symbolic Logic Vol 5 (1940) pp56-68.

[9] Curry, HB "Grundlagen der Kombinatorische Logik" American Journal of Mathematics, Vol 52 (1930) pp509-536, 789-834.

[10] de Bruin, NG "A survey of the AUTOMATH project" in To HB Curry: Essays on combinatory logic, lambda calculus and formalism, eds JR Hindley and JP Seldin, Academic Press 1980 pp580-606.

[11] Girard, J-Y "Une extension de l'interpretation functionnelle de Gödel à l'analyse et son application à l'êlimination des coupures dans l'analyse et la théorie des types" Proceedings of the Second Scandinavian Logic Symposium, ed JE Fenstad, North Holland 1971.

[12] Hindley, JR, Basic Simple Type Theory, Cambridge University Press 1997.

[13] Hindley, JR and Seldin, JP, Introduction to Combinators and $\lambda$-Calculus, Cambridge University Press 1986.

[14] Martin-Lof, P "An intuitionistic theory of types: predicative part" Logic Colloquium '73, North Holland 1982 pp 153-175.

[15] Terlouw, J "Een nadere bewijstheoretische analyse van GSTT's" Faculty of Mathematics and Computer Science, University of Nijmegen, 1989.