INFINITE NORMAL FORMS FOR THE $\lambda$-CALCULUS
AND SEMANTICS OF PROGRAMMING LANGUAGES

by

Reiji Nakajima

Memorandum No. ERL-M525

June 1975

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720
INFINITE NORMAL FORMS FOR THE $\lambda$-CALCULUS AND SEMANTICS OF PROGRAMMING LANGUAGES*

Reiji Nakajima

Computer Science Division
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California at Berkeley

June 1975

Abstract

The semantics of programming languages is studied through the notion of infinite expansions of programs. By the infinite expansion of a program one means, for example, the thorough unwinding of the loops which are constituted by such control structures as go to's, while's and recursions. One can also view this infinite expansion as the executions for all possible inputs. One way to describe the meaning (or the semantics) of a program is to give its infinite expansion.

This idea is formalized on the domain of the $\lambda$-calculus. We define a mapping, from the $\lambda$-expressions to an algebraic domain, called C-function. The map of a $\lambda$-expression (program) by the C-function is the infinite expansion of the $\lambda$-expression which can be said to be a generalization of the normal forms for the $\lambda$-calculus. Böhm's Theorem on the normal $\lambda$-expression is extended to general $\lambda$-expressions via the C-function.

The main result of this thesis is that the semantics of the $\lambda$-expressions given by Scott's model $D_\infty$ of $\lambda$-calculus is equivalent to the semantics of the $\lambda$-expressions given by their maps.

*This work was partially supported by National Science Foundation grant GJ-34342X.
of the C-function. More precisely, the partial order among the λ-expressions in $D_\infty$ is characterized by the partial order among their maps by the C-function in the algebraic domain that includes the image of the C-function. Extending the syntactical structure of the C-function, the λ-expressions are generalized to the infinite λ-expressions and the C-function is also extended to be defined on all the infinite λ-expressions. It is shown that the image of the infinite λ-expressions by the C-function forms a smooth structure of the partial order and its lattice topology is equivalent to the lattice topology of the λ-expressions induced by $D_\infty$.

Utilizing this lattice topology, an attempt is made to give an axiomatization of the extensional model theory of the λ-calculus. Also, the formal idea described above is interpreted to realistic programming languages such as Algol-like programs and recursively defined programs.
Acknowledgment

I am deeply indebted to Dr. James Morris at Xerox Corporation for his whole-hearted support of my work, most of which went beyond his obligations and duties.

Special acknowledgment is due to Dr. Christopher Wadsworth who offered numerous helpful suggestions throughout my research.

I thank Professors Leon Henkin and Richard Karp for their useful comments on this thesis. I owe the beauty of this thesis to Ms. Ruth Suzuki's artistic talent.

A large number of people encouraged me 'unofficially' during my three year stay at Berkeley. Since such a large acknowledgment, however, does not suit this modest thesis and since any private feeling does not match the $\lambda$-calculus, I would rather express my appreciation to them by dedicating this thesis to:

Maka Hannya Haramita
# INFINITE NORMAL FORMS FOR THE λ-CALCULUS AND SEMANTICS OF PROGRAMMING LANGUAGES

Reiji Nakajima

## Table of Contents

<p>| Abstract | ii |
| Acknowledgments | iv |
| Table of Contents | v |
| Chapter 1  INTRODUCTION | 1 |
| Chapter 2  PREPARATIONS | 7 |
| 2.1 The λ-Calculus | 8 |
| 2.2 Theory of Computation on Lattice Domains, D∞ Model | 15 |
| 2.3 Wadworth's Model Theory of λ-Calculus in D∞ | 21 |
| Chapter 3  INFINITE NORMAL FORMS FOR λ-CALCULUS | 31 |
| 3.1 Pedigree | 32 |
| 3.2 Idea of Infinite Expansion | 33 |
| 3.3 L-function | 36 |
| 3.4 C-functions | 39 |
| 3.5 C-function as Infinite Normal Form — Extension of Böhm's Theorem | 56 |
| Chapter 4  CHARACTERIZATION OF THE D∞-VALUES OF THE λ-EXPRESSIONS | 58 |
| 4.1 Structural Approximation | 59 |
| 4.2 Convergence Lemma | 65 |
| 4.3 Characterization of D∞-Value of λ-Expressions | 77 |
| 4.4 Further Properties of Λ Mapped in D∞ | 79 |
| Chapter 5  INFINITE EXPANSIONS IN REAL PROGRAMMING LANGUAGES | 86 |
| 5.1 Recursively Defined Programs | 87 |
| 5.2 Algol-like Language | 93 |</p>
<table>
<thead>
<tr>
<th>Chapter 6</th>
<th>GENERALIZED λ-EXPRESSIONS, A NATURAL LATTICE STRUCTURE OF THE λ-CALCULUS</th>
<th>99</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Infinite Programs</td>
<td>100</td>
</tr>
<tr>
<td>6.2</td>
<td>Characterization of C-functions</td>
<td>105</td>
</tr>
<tr>
<td>6.3</td>
<td>Infinite λ-expressions</td>
<td>111</td>
</tr>
<tr>
<td>6.4</td>
<td>Lattice Structure of $C_{\text{fin}}$ and $C_{\text{inf}}$</td>
<td>121</td>
</tr>
<tr>
<td>Chapter 7</td>
<td>SUMMARY, CONCLUSIONS, PROSPECTS</td>
<td>129</td>
</tr>
<tr>
<td>7.1</td>
<td>Summary</td>
<td>130</td>
</tr>
<tr>
<td>7.2</td>
<td>Universality of $C_{\text{inf}}$ over $\Lambda$</td>
<td>132</td>
</tr>
<tr>
<td>7.3</td>
<td>Prospects</td>
<td>137</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>140</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

The aim of this dissertation is to study some properties of the \( \lambda \)-calculus as a computation model and contribute to a better understanding of the semantics of programming languages. The \( \lambda \)-calculus was originally introduced by Church as a logical system. A variety of formal theories on the \( \lambda \)-calculus were discussed by several mathematical logicians, e.g. [5].

The \( \lambda \)-calculus has attracted some theoretical computer scientists since it can be regarded as a model of programming languages [6, 7, 19]. Many concepts of programming languages were analyzed through the corresponding concepts in the \( \lambda \)-calculus.

However, the sound understanding of the \( \lambda \)-calculus as a model of computation became possible only after Scott developed the theory of computation on lattice domain [14, 15, 18], in which he gave the construction of \( D_\infty \), the first semantic model for the \( \lambda \)-calculus [15, 16]. On this domain, the model theory of the \( \lambda \)-calculus was developed by Wadsworth [21, 22] and many interesting properties of the behavior of the \( \lambda \)-expressions in \( D_\infty \) were shown as we see in Chapter 2. In this thesis, we make efforts to develop further the theory on \( \lambda \)-expressions vs. \( D_\infty \). In [16], Scott gives an interesting lecture on the \( \lambda \)-calculus. There, he asserts that the interpretation of the \( \lambda \)-calculus via \( D_\infty \) gives a more essential meaning to the \( \lambda \)-calculus than the conversion rules. For example, Wadsworth proved that there exists a normal \( \lambda \)-expression which is equivalent
to a non-normal expression in Scott's $D_\infty$ although they cannot be converted to each other by the applications of conversion rules. In Chapter 3, we shall define the "infinite normal forms" for the $\lambda$-expressions, normal or non-normal. Then we shall show that two $\lambda$-expressions are equivalent under Scott's interpretation (i.e. $D_\infty$) if they have the same infinite normal forms. The infinite normal form can be said to be the infinite expansion of a $\lambda$-expression.

We illustrate this idea of infinite expansion in the following discussion on flowchart programs.

We are given a flow chart program:

```
AO: \[
\begin{array}{c}
\alpha \\
\downarrow \\
\alpha \quad Yes \\
\downarrow \\
\alpha \quad No \\
\downarrow \\
S \quad S
\end{array}
\]
```

where $\alpha$ is a Boolean function and $S$ is a statement (or a list of statements)
Since $A_1$ is the result of unwinding the loop in $A_0$ one time, $A_1$ is equivalent to $A_0$. From another point of view, the transformation $A_1 \Rightarrow A_2$ can be regarded as the execution of $A_0$ for one time under an unspecified input.
Applying this operation \( n \) times, we have
Letting $n \to \infty$, we have an infinitely sequential flowchart:

\[
\begin{array}{cccc}
\text{A}^\infty: & \text{Yes} & \text{No} \\
\alpha & \downarrow & \downarrow \\
\text{S} & \downarrow & \downarrow \\
\alpha & \downarrow & \downarrow \\
\ddots & \ddots & \ddots \\
\text{S} & \downarrow & \downarrow \\
\end{array}
\]

$A^\infty$ can be regarded as the infinite expansion of $A_0$ or the result of execution of $A_0$ under all possible inputs.

It is possible to apply this idea to more complex program constructs. The infinite expansion of programs can be formalized in the following way: Let $P$ be the domain of (some category of) programs and $I$ be the domain of the infinite expansion of the programs which belong to $P$. The expansion is a mapping $E: P \to I$. (in the example, for $A_0 \in P$, $E(A_0) = A^\infty$.) We raise the following questions:

1) How can we formalize $I$ and $E$?
2) Can we say that the meaning (semantics) of a program \( P \) is given by \( E(P) \)? So, for instance, is \( P_1 \) equivalent to \( P_2 \) if and only if \( E(P_1) = E(P_2) \)?

3) What kind of structure does \( I \) have? Does it have, for instance, a lattice-like structure?

We shall answer these questions regarding \( \lambda \)-expressions as programs. Namely, we have the following correspondence:

\[
\begin{align*}
P & \rightarrow \Lambda \quad \text{(the \( \lambda \)-expressions)} \\
I & \rightarrow C_{\text{inf}} \quad \text{(a partially ordered set defined in Chapter 6)} \\
E & \rightarrow C \quad \text{(C-function in Chapter 3 or infinite normal form)}
\end{align*}
\]

Here, the transformation \( A_n \rightarrow A(n+1) \) corresponds to a \( \beta \)-reduction. \( C \) gives the map \( A_0 \rightarrow A_\infty \). On the other hand, the equivalence between two programs (i.e. \( \lambda \)-expressions) \( P_1 \) and \( P_2 \) is given by the equality as members of \( D_\infty \).

In Chapter 5 we shall try to bridge the gap between the \( \lambda \)-expressions and the real programming languages and show that, under some translation of programming languages, the infinite normal forms, in fact, correspond to the infinite expansion or execution of programs.
CHAPTER 2
PREPARATIONS

We make a review on the $\lambda$-calculus, Scott's lattice theoretic approach to computation and Wadsworth's model theory of the $\lambda$-calculus in $D_\infty$, which constitute the prerequisite for this thesis.
§1. The λ-Calculus

We shall denote the set of the integers by \( \mathbb{Z} \) and the set of the positive integers by \( \mathbb{N} \) throughout this thesis.

2.1.1 Definition (λ-expression). \( \Lambda \) is the set of all of the expressions that are formed by the following rules:

Let \( U \) be the denumerable set of the variables. Assume that there is a numbering on the members of \( U \), i.e. \( U = \{ v_1, v_2, \ldots \} \).

1) A variable \( v \in U \) standing alone is in \( \Lambda \).
2) (Application). If \( x, y \in \Lambda \), so is \( x(y) \).
3) (Abstraction). If \( v \in U \) and \( x \in \Lambda \) then \( \lambda v. x \in \Lambda \).

2.1.2 Example. We list some λ-expressions:

\[
I = \lambda v. v
\]
\[
K = \lambda x. \lambda y. x
\]
\[
H = \lambda x. \lambda y. y
\]
\[
spI = (\lambda x. x(x))(\lambda x. x(x))
\]
\[
Y = \lambda f. (\lambda x. f(x(x)))(\lambda x. f(x(x))) \quad \text{(Curry's Paradoxical Combinator)}
\]
\[
J = Y(\lambda f. \lambda x. \lambda y. x(f(y))) \quad \text{(Wadsworth)}
\]

2.1.3 Definition (Bound Variables). Given \( x \in \Lambda \), we define \( B(x) \subset U \), the set of the bound variables in \( x \).

1) \( B(v) = \emptyset \) for \( v \in U \).
2) \( B(x(y)) = B(x) \cup B(y) \) for \( x, y \in \Lambda \).
3) \( B(\lambda u. x) = B(x) \cup \{ u \} \).

In 3), \( x \) is said to be the scope of the bound variable \( v \). If a variable \( v \) occurring in \( x \) is not bound in \( x \), we say that \( v \) is free in \( x \). If \( x \in \Lambda \) has no free variables, \( x \) is said
to be closed. We denote the set of all closed λ-expressions by $\Lambda_c$.

2.1.4 Definition (Subexpression). Given $x, y \in \Lambda$, we say that $x$ is a subexpression of $y$ and write it as $x < y$ if one of the following conditions holds:

1) $x = y$.
2) $x < z$ and $y = w(z)$ or $z(w)$ for some $w \in \Lambda$.
3) $x < z$ and $y = \lambda v . z$ for $v \in U$.

2.1.5 Definition (Simultaneous Substitution). The simultaneous substitution of $x_1, x_2, \ldots, x_n \in \Lambda$ for $u_1, u_2, \ldots, u_n \in U$ in $y \in \Lambda$, $\int_{x_1, x_2, \ldots, x_n}^{u_1, u_2, \ldots, u_n} y$, is defined inductively as:

Let $u = (u_1, u_2, \ldots, u_n)$ and $x = (x_1, x_2, \ldots, x_n)$.

1) If $y \in U$ and $y \not\in u_i$ for all $i$ then $\int_x^u y = y$.
2) If $y = u_i$ for $1 \leq i \leq n$, then $\int_x^u y = x_i$.
3) If $y = a(b)$ for $a, b \in \Lambda$, then $\int_x^u y = (\int_x^u a)(\int_x^u b)$.
4) If $y = \lambda v . z$ for $v \not\in u_i$ ($i = 1, 2, \ldots, n$), then if $v$ is not free in any of the $x_i$'s then $\int_x^u y = \lambda v. \int_x^u z$ otherwise $\int_x^u y = \lambda v'. \int_x^u (\int_x^V z)$ where $v'$ is the first variable, other than any $u_i$'s or $v$ in the enumeration of the variables in $U$ such that $v'$ does not occur free in $y$ or $z$.
5) If $y = \lambda u_i . z$ for $1 \leq i \leq n$, then

$$\int_x^u y = \lambda u_i. \int_{x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n}^{u_1, u_2, \ldots, u_{i-1}, u_{i+1}, \ldots, u_n} y .$$
2.1.6 Definition. Given $x, y, z \in \Lambda$, we say that $x$ matches $y$ except at occurrences of $z$ in $x$ if there exists $w \in \Lambda$ having free occurrences of $v \in U$ and $z_0 \in \Lambda$ such that

$$ x = \int_z^v w \quad \text{and} \quad y = \int_z^{z_0} w. $$

In this case, we say that $z$ in $x$ is homologous to $z_0$ in $y$.

Notational Convention. We will use the following notational abbreviation:

1) $xy$ stands for $x(y)$.
2) $x_1 x_2 \cdots x_n$ stands for $((\cdots (x_1 x_2) x_3) \cdots) x_n$.
3) $\lambda s_1 s_2 \cdots s_n . x$ stands for $\lambda s_1 . \lambda s_2 . \cdots . \lambda s_n . x$.

So note that a $\lambda$-expression can generally be written as:

$$ \lambda t_1 t_2 \cdots t_m . x_1 x_2 \cdots x_n \text{ for } t_1, t_2, \ldots, t_m \in U \text{ and } x_1, x_2, \ldots, x_n \in \Lambda. $$

2.1.7 Definition (Conversion Rules). Let $\xi = \alpha, \beta, \eta$-red or $\eta$-ab. We will define $R_\xi \subseteq \Lambda \times \Lambda$ for each case of $\xi$. $(x, y) \in R_\xi$ is denoted by $x \xrightarrow{\xi} y$.

$(x, y) \in R_\xi$ if

I) a) $(\alpha$-conversion) $\xi = \alpha$: $x = \lambda u . z$ and $y = \lambda v \int_v^u z$ under the following restrictions:

i) $v$ does not occur free in $z$.

ii) If $v \in B(z)$, any free occurrence of $u$ in $z$ must not be in the scope of $v$.

b) $(\beta$-reduction) $\xi = \beta$: $x = (\lambda v . z) w$ and $y = \int_w^v z$.

c) $(\eta$-abstraction) $\xi = \eta$-ab: $y = \lambda v . x v$ where $v$ does not occur free in $x$. 


d) \( (\eta\text{-reduction}) \quad \xi = \eta\text{-red}: x = \lambda v.yv \) where \( v \) does not occur free in \( y \).

or

II) \( y \) is derived from \( x \) by applying \( \xi \)-conversion (reduction) to a subexpression of \( x \).

We define \( \text{CNV} \subseteq \Lambda \times \Lambda \) to be the reflective, transitive closure of \( R_\alpha \cup R_\beta \cup R_{\eta\text{-red}} \cup R_{\eta\text{-ab}} \), i.e. \( x \xrightarrow{\text{CNV}} y \) if and only if \( x = y \) or there exist \( x_1, x_2, \ldots, x_n \in \Lambda \) such that \( x = x_1 \xrightarrow{\xi_1} x_2 \xrightarrow{\xi_2} \cdots \xrightarrow{\xi_{n-1}} x_n = y \) where \( \xi_1 = \alpha, \beta, \eta\text{-ab} \) or \( \eta\text{-red} \).

2.1.8 Definition. a) A \( \beta\text{-redex} \) is a \( \lambda \)-expression in the form of \( (\lambda v.x)y \). A \( \lambda \)-expression is said to have a \( \beta\text{-redex} \) if one of its subexpressions is a \( \beta\text{-redex} \).

b) A \( \beta\text{-redex} \) \( y \) in \( x \in \Lambda \) is said to be the outermost-leftmost \( \beta\text{-redex} \) if there is no \( \beta\text{-redex} \) \( w \) such that

\[ i) \quad y < w < x \]

or

\[ ii) \quad w < a, \quad y < b \quad \text{and} \quad ab < x. \]

c) Let \( x \in \Lambda \) have \( (\lambda v.y)z \) as its outermost-leftmost \( \beta\text{-redex} \). The outermost-leftmost \( \beta\text{-reduction} \) to \( x \) is the replacement of \( (\lambda v.y)z \) in \( x \) by \( \int^v_z y \).

2.1.9 Definition. a) \( y \in \Lambda \) is said to be in a head normal form if \( y \) is in the form of \( \lambda s_1 s_2 \cdots s_m.v_y y_1 y_2 \cdots y_n \) for \( s_1, s_2, \ldots, s_m, v \in U \) and \( y_1, y_2, \ldots, y_m \in \Lambda \).
b) $y \in A$ is said to be head normal if there exists a sequence: $y = x_1 \xrightarrow{B} x_2 \xrightarrow{B} \ldots \xrightarrow{B} x_{n-1} \xrightarrow{B} x_n$ and $x_n$ is in a head normal form. Here $x_n$ is said to be a head normal form of $y$.

2.1.10 Corollary. Let $x \in A$ be head normal. If

$$x \xrightarrow{CV} \lambda s_1 s_2 \cdots s_m ux_1 x_2 \cdots x_n$$

and

$$x \xrightarrow{CV} \lambda r_1 r_2 \cdots r_p vy_1 y_2 \cdots y_q,$$

then
1) If $u$ occurs free in $x$, then $u = v$.
2) If $u = s_i$ for $i < m$, then $i \leq p$ and $v = r_i$.
3) $m - n = p - q$.

Proof. See [21].

In Corollary 2.1.10 let

$$\text{head}(x) = \begin{cases} u \in U & \text{if } u \text{ is free in } x \\ i \in \mathbb{N} & \text{otherwise} \end{cases}$$

and

$$\text{index}(x) = m - n.$$

By the corollary, $\text{head}(x)$ and $\text{index}(x)$ are uniquely defined for $x \in A$ if $x$ has a head normal form. We define the relationship $\sim \subseteq A \times A$ by:

$$x \sim y \text{ if either neither } x \text{ nor } y \text{ has a head normal form or } \text{index}(x) = \text{index}(y) \text{ and } \text{head}(x) = \text{head}(y).$$
2.1.11 Definition. \( x \in A \) is said to be in a \( \beta \)-normal form if \( x \) has no \( \beta \)-redex as its subexpression. We say that \( x \in A \) is \( \beta \)-normal if \( x \) can be reduced to a normal form by \( \beta \)-reductions.

Note that if a \( \lambda \)-expression is normal, it is also head-normal.

2.1.12 Theorem. If \( x \in A \) is head-normal (normal), then there exists the following sequence:

\[
x = x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_n
\]

where \( x_{i+1} \) is the result of the outermost-leftmost \( \beta \)-reduction applied to \( x_i \) for \( n = 1,2,\ldots,n-1 \) and \( x_n \) is in a head-normal form (normal form).

Proof. For the normal form case, see [5]. The proof is similar for the head normal case.

To have a certain uniqueness for the head normal form, we define \( x \xrightarrow{\beta h} y \) as follows: \( x \xrightarrow{\beta h} y \) if there is a sequence \( x = x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_n = y \) such that \( x_{i+1} \) is the result of outermost-leftmost \( \beta \)-reduction to \( x_i \). \( x_n \) is in a head normal form and \( x_i \) is not in a head normal form for \( i \neq n \).

It is easy to see that if \( x \xrightarrow{\beta h} y \) then \( y \) is uniquely determined by \( x \).

2.1.13 Theorem (Scott). It is not decidable whether a \( \lambda \)-expression is normal or whether a \( \lambda \)-expression is head-normal.

Proof. See [7].
The following theorem is fundamental in the theory of the λ-calculus.

2.1.14 **Theorem** (Church-Rosser). Given \( x, y_1, y_2 \in \Lambda \), if \( x \xrightarrow{\text{CNV}} y_1 \) and \( x \xrightarrow{\text{CNV}} y_2 \), then there exists \( z \in \Lambda \) such that both \( y_1 \xrightarrow{\text{CNV}} z \) and \( y_2 \xrightarrow{\text{CNV}} z \).

**Proof.** See [2]. \( \Box \)
§2. Theory of Computation on Lattice Domains, D∞ Model

Scott [14] proposed the following axioms that a mathematical model of computation ought to have:

Axiom 1: A domain D is a complete lattice. We denote $\bigvee D$ by $\top$ (top) and $\bigwedge$ by $\bot$ (bottom).

2.2.1 Definition. a) Let D be a partially ordered set. A subset $S \subseteq D$ is said to be directed if, for any finite subset $F$ of $S$, there exists $z$ in $S$ such that

$$x \subseteq^* z \text{ for all } x \in F.$$

b) A partially ordered set $D$ is said to be directed-complete if all directed subsets of $D$ have the least upper bound.

c) A function from a partially ordered set $D_1$ to another partially ordered set $D_2$ is said to be continuous if, for all directed sets $E \subseteq D_1$,

$$f(\cup E) = \bigvee \{f(x) | x \in E\}.$$

$f$ is said to be additive if

$$f(\cup S) = \bigvee \{f(x) | x \in S\}$$

for all subsets $S \subseteq D_1$.

As we see in Chapter 7, the completeness is not necessarily needed for the development of the theory in this thesis. At most, we would need a directed-complete partially ordered set with the least element $\bot$. Given two partially ordered sets $D_1, D_2$, we

*We use $\cup$ and $\subseteq$ instead of $\bigvee$ and $\subseteq$ for typographical convenience.
denote the set of all continuous functions from $D_1$ to $D_2$ by $[D_1 \to D_2]$ (we denote all maps $D_1 \to D_2$ by $(D_1 \to D_2)$.)

2.2.2 **Corollary** (Scott). If $D_1$, $D_2$ are directed-complete (complete), then $[D_1 \to D_2]$ is also a directed complete (complete) lattice, where we define $\subseteq$ in $[D_1 \to D_2]$ by: $f \subseteq g$ if and only if $f(x) \subseteq g(x)$ for all $x \in D_1$.

**Proof.** See, for example, [12]. □

**Axiom 2:** A map from domain $D_1$ to domain $D_2$ is continuous.

2.2.3 **Theorem** (Scott). Let $f$ be a continuous function over a directed-complete partially ordered set $D$. Then $f$ has the least fixed point $\bigcup_{n=0}^{\infty} f^n(\bot)$.

**Proof.** See [12]. □

2.2.4 **Definition.**

a) A subset $G$ of a directed-complete subset $D$ is said to be **open** if

1) For any $x \in G$, if $x \subseteq y$, then $y \in G$.

2) For any directed set $D \subseteq D$, if $\cup \mathcal{D} \in G$ then $D \cap G \neq \emptyset$.

b) For $x, y \in D$, we say $x \prec y$ ($x$ is strictly less than $y$) if there is an open set $G$ such that $y \in G$ and $G \subseteq \{z | x \leq z\}$.

c) A directed complete partially ordered set $D$ is said to be **continuous** if, for all $x \in D$, $x = \cup \{y | y \prec x\}$.

Note that a domain has a $T_0$-topology induced by the open sets defined above. Continuous mappings are continuous in this topological sense.
Axiom 3: A domain is a continuous lattice.

The last axiom is on the computability:

Axiom 4: A domain \( D \) has a subset \( E \) of the following properties:

1) The cardinality of \( E \) is at most denumerable and the elements of \( E \) are recursively enumerable.

2) For any \( x \in D \), \( x = \bigcup \{ y \in E \mid y < x \} \).

3) For all \( e_1, e_2 \in E \), \( e_1 \cup e_2 \) and \( e_1 \subseteq e_2 \) are computable.

Next, we state the construction of \( D_\infty \)-lattice. We shall confine ourselves to the description of the properties of \( D_\infty \) that are needed in our discussion in the subsequent chapters. For the complete presentation of \( D_\infty \), see [12].

Construction of \( D_\infty \)

We want to have a lattice domain \( D \) with the property \( D = [D + D] \). Let \( D_0 \) be any complete lattice. (In fact, a directed-complete partially ordered set is good enough for our purpose, but for simplicity, we assume the completeness.)

Let \( D_1 = [D_0 + D_0] \), \( D_2 = [D_1 + D_1] \), \ldots, \( D_n = [D_{n-1} + D_{n-1}] \), \ldots. Note that each \( D_n \) is a complete lattice. We define \((i_n, j_n)\) for each \( n \) such that

1) \( i_n : D_n \to D_{n+1} \), \( j_n : D_{n+1} \to D_n \)

2) \( i_n^* j_n \) are additive and \( j_n \circ i_n = 1_{D_n} \) and \( i_n \circ j_n \subseteq 1_{D_{n+1}} \) (so \( i_n \) is one-to-one and \( j_n \) is onto).
Definition of \((i_n, j_n)\)

\[
i_0(a) = \lambda \beta \in D_0 : a \quad \text{for each } a \in D_0
\]
\[
j_0(x) = x(\lvert D_0 \rvert) \quad \text{for each } x \in D_1
\]

Suppose that we have defined \((i_n, j_n)\) for \(n \leq k-1\) \((k \geq 1)\). We define \((i_k, j_k)\)

\[
D_{k-1} \leftarrow \frac{i_{k-1}}{j_{k-1}} \rightarrow D_k \leftarrow \frac{i_k}{j_k} \rightarrow D_{k+1}
\]

\[
i_k(x) = i_{k-1} \circ x \circ j_{k-1} \quad \text{for all } x \in D_k
\]
\[
j_k(y) = j_{k-1} \circ y \circ i_{k-1} \quad \text{for all } y \in D_{k+1}
\]

It is easy to see that \((i_k, j_k)\) satisfies the properties 1) and 2) by induction on \(k\).

We define \(D_\infty = \{(x_0, x_1, \ldots, x_n, \ldots) | x_n \in D_n, x_n = j_n(x_{n+1})\}\),

where, for \(x, y \in D_\infty\), \(x \subseteq y\) if and only if \(x_i \subseteq y_i\) for all \(i\).

Embedding of \(D_n\) in \(D_\infty\)

We define \(\phi_{nm} : D_m \rightarrow D_n\) as follows:

\[
\phi_{nm} = \begin{cases} 
    i_{n-1} \circ i_{n-2} \circ \ldots \circ i_{m+1} \circ i_m & \text{if } m < n \\
    1_{D_m} & \text{if } m = n \\
    j_n \circ j_{n-1} \circ \ldots \circ j_m \circ j_{m+1} & \text{if } n < m
\end{cases}
\]

Now we embed \(D_n\) into \(D_\infty\) by

\[
E_n : D_n \rightarrow D_\infty \quad x \mapsto \langle \phi_{0n}(x), \phi_{1n}(x), \ldots, \phi_{(n-1)n}(x), x, \phi_{n(n+1)}(x), \ldots \rangle
\]

By defining \(D = E_n(D_n) \subseteq D_\infty\),

1) \(E_n : D_n \rightarrow D\) is one-to-one and continuous

2) \(D_0 \subseteq D_1 \subseteq D_2 \subseteq \ldots\)
Conversely, the projection \( \pi_n: D_\infty \to \mathbb{D}_n \) is defined by:

\[
\pi_n: \langle x_0, x_1, x_2, \ldots, x_n, \ldots \rangle \in D_\infty \\
\mapsto \langle x_0, x_1, \ldots, x_n, \phi(n+1)(x_n), \phi(n+2)(x_n), \ldots \rangle \in \mathbb{D}_n.
\]

Also we define \( P_n: D_\infty \to \mathbb{D}_n \) by

\[
P_n: \langle x_0, x_1, \ldots, x_n, \ldots \rangle \mapsto x_n.
\]

It is easy to see that \( x = \bigcup_{n=0}^\infty \pi_n(x) \) for all \( x \in D_\infty \).

**Isomorphism** \( D_\infty \cong [D_\infty \to D_\infty] \)

We define

\[
\phi: D_\infty \to [D_\infty \to D_\infty] \\
\psi: [D_\infty \to D_\infty] \to D_\infty
\]

by: For all \( x \in D_\infty \)

\[
\phi(x)(y) = \bigcup_{n=0}^\infty f_n(P_{n+1}(x)(P_n(y))) \text{ for all } y \in D_\infty.
\]

For all \( f \in [D_\infty \to D_\infty] \)

\[
\psi(f) = \langle f_0, f_1, f_2, \ldots, f_n, \ldots \rangle
\]

where

\[
f_0 = P_0(f(\ottilecl)) \\
f_n = \lambda x \in D_{n-1}: P_{n-1}(f(E_{n-1}(x))) \text{ for } n \geq 1.
\]

In a straightforward way, we can verify that \( \phi, \psi \) are additive, \( \psi \circ \phi = 1_{D_\infty} \) and \( \phi \circ \psi = 1_{[D_\infty \to D_\infty]} \).
We can now define the application of $x$ to $y$ for $x, y \in D_\infty$ by $\Phi(x)y$. We denote this by $x(y)$. By the definition of $\Phi$,

$$\Phi(x)(y) = \bigcup_{n=1}^{\infty} \pi_n(x)(\pi_{n-1}(y)).$$

Lastly we list the important properties of $D_\infty$ projections:

2.2.5 **Theorem** (Scott). 1) $\pi_m \subseteq \pi_n \subseteq 1_{D_\infty}$ for $m \leq n$

2) $\bigcup_{n=0}^{\infty} \pi_n = 1_{D_\infty}$

3) $\pi_n \circ \pi_m = \pi_{\min(n,m)}$

4) $\pi_n(x)(y) = \pi_n(x)(\pi_{n-1}(y)) = \pi_{n-1}(x(\pi_{n-1}(y)))$

5) $\pi_0(x)(y) = \pi_0(x) = \pi_0(x(\|))$
§3. Wadworth's Model Theory of $\lambda$-Calculus in $D_\infty$

In this section, we state the results due to Wadsworth [21,22].

As we have seen in the last section: $D_\infty \xrightarrow{\phi} [D_\infty \to D_\infty]$ for continuous $\phi$, $\psi$ satisfying $\psi \circ \phi = 1_{D_\infty}$ and $\phi \circ \psi = 1_{[D_\infty \to D_\infty]}$.

This property of $D_\infty$ can be characterized in the following way:

1) **Extensionality**: $x(z) \subseteq y(z)$ for all $z \in D_\infty$ iff $x \subseteq y$ so, particularly, $x(z) = y(z)$ for all $z \in D_\infty$ iff $x = y$.

2) **Comprehension**: If $\cdots x \cdots$ is an expression taking values on $D_\infty$ which is continuous in the variable $x$ as $x$ ranges over $D_\infty$, then there is $f \in D_\infty$ such that

$$f(a) = \cdots a \cdots \text{ for all } a \in D_\infty .$$

2.3.1 **Definition** (Wadsworth). Let $EN$ be the set of all mappings from the set of the variables $U$ to $D_\infty$. The semantic function $W : \Lambda \to (EN \to D_\infty)$ is defined as follows:

1) For $v \in U$ and $\rho \in EN$, $W[[v]] \rho = \rho(v)$.

2) For $x(y) \in \Lambda$ and $\rho \in EN$, $W[[x(y)]] \rho = W[[x]] \rho(W[[y]] \rho)$.

3) For $\lambda v.x \in \Lambda$ and $\rho \in EN$,

$$W[[\lambda v.x]] \rho = \lambda \beta \in D_\infty : W[[x]] \rho[v/\beta]$$

where $\rho[v/\beta]$ is defined by

$$\rho[v/\beta](u) = \begin{cases} 
\rho(u) & \text{if } u \neq v \\
\beta & \text{if } u = v .
\end{cases}$$

Since $W[[x]] \rho[v/\beta]$ is continuous in the variable $\beta$, $\lambda \beta \in D_\infty : W[[x]] \rho[v/\beta]$ is a member of $D_\infty$ due to the comprehension of $D_\infty$. 
2.3.2 \textbf{Proposition} (Wadsworth). If $x \xrightarrow{\text{CNV}} y$ for $x, y \in \Lambda$, then 
\[ \forall [x] \rho = \forall [y] \rho \quad \text{for all } \rho \in \mathcal{E}. \]

\textit{Proof.} The result is obvious for the $\alpha$-conversion and $\beta$-reduction. The $\eta$-conversions preserve the $D_\infty$ value due to the extensionality of $D_\infty$. \hfill \square

2.3.3 \textbf{Definition.} We say $x \subseteq y$ for $x, y \in \Lambda$ if 
\[ \forall x [x] \rho \subseteq \forall y [y] \rho \quad \text{for all } \rho \in \mathcal{E}. \] 
Similarly $x = y$ if 
\[ \forall [x] \rho = \forall [y] \rho \quad \text{for all } \rho \in \mathcal{E}. \]

2.3.4 \textbf{Corollary.} $\subseteq$ is reflective and transitive.

\textit{Proof.} Obvious. \hfill \square

However, $\subseteq$ is obviously not antisymmetric, so $\subseteq$ is not a partial ordering.

We first show that $\Lambda$ is not trivial in $D_\infty$, namely, $\Lambda$ is not mapped into one element in $D_\infty$.

2.3.5 \textbf{Proposition.} $K \not\equiv H$ and $I \not\equiv \bot_\infty$.

\textit{Proof.} See [21]. \hfill \square

2.3.6 \textbf{Theorem} (Wadsworth). Let $I = \lambda x.x$ and $J = Y(\lambda f x.y(fy))$.

Then $I = J$.

\textit{Proof.} See [22] for the proof based on the type construction. Also see Example 4.3.4. \hfill \square
Since I is normal and J is non-normal, I and J are not convertible to each other. So this shows that $=_{D_{\infty}}$ is strictly larger than $\text{CNV_\lambda}$, i.e. $\text{CNV_\lambda} \subsetneq D_{\infty}$.

The next theorem shows that Curry's Y gives the least fixed point operator in $D_{\infty}$.

2.3.7 Theorem (Park). $Y = \lambda f \in D_{\infty}: \bigcup_{n=0}^{\infty} f^n(\bot)$.

Proof. See [12], also Corollary 4.2.3. □

We can introduce $\cup$, $\cap$ operations in $\Lambda$ as follows: Given $S \subseteq \Lambda$, $\cup S$ is a syntactic object with the semantic value in $D_{\infty}$ of:

$$\forall \{\cup S\} \rho = \bigcup \{\forall \{\lambda x\} \rho | x \in S\}$$

for $\rho \in \text{EN}$.

We define $\cap S$ in the similar manner.

$\lambda$-$\Omega$-Calculus

It is convenient to have a syntactical symbol in $\Lambda$ that represents $\bot$. The $\lambda$-$\Omega$-expressions, $\lambda_{\Omega}$, are formed according to the following rules:

1) $\Omega$ is in $\lambda_{\Omega}$.

2) Same as Definition 2.1.1.

Semantic function $\forall$ is $\bot$ on $\Omega$, i.e.

$$\forall \{\Omega\} \rho = \bot$$

for all $\rho \in \text{EN}$.

We include two conversion rules for $\lambda_{\Omega}$ in addition to those for $\lambda$. 
1) $\lambda v. \Omega \rightarrow \Omega$ for $v \in U$
2) $\Omega(x) \rightarrow \Omega$ for $x \in \Lambda_{\Omega}$

These rules are semantically sound since $\lambda \beta \in D_\infty$: $\bot = \bot = \bot(a)$ for $a \in D_\infty$.

Type Assignments of $\lambda$-Expressions

This part of the section is needed to prove Lemma 4.2.1. For the details of the discussion, we refer to [23].

As a member of $D_\infty$, each $\lambda$-expression has a component in each $D_n$. The typed $\lambda$-expressions defined below are introduced to, in a sense, approximate the components of $\lambda$-expressions in $D_\infty$.

2.3.8 Definition (Typed $\lambda$-expressions).

Syntax of $\Lambda^t$

The typed $\lambda$-expression, $\Lambda^t$, is the set of all expressions that are formed by the rules below:

1) For $v \in U$, $v^n \in \Lambda^t$ for each $n \in \mathbb{N}$.
2) If $x, y \in \Lambda^t$, $(x(y))^{(n)} \in \Lambda^t$ for $n \in \mathbb{N}$. $(x(y))^{(n)}$ is abbreviated as $(xy)^{(n)}$.
3) For $v \in U, x \in \Lambda^t$, $(\lambda v.x)^n \in \Lambda^t$ for $n \in \mathbb{N}$.
4) $\Omega^{(n)} \in \Lambda^t$ for $n \in \mathbb{N}$.

Semantics of $\Lambda^t$

The semantic function, $\llbracket : \Lambda^t \rightarrow (\mathbb{N} \rightarrow D_\infty)$ is defined as:

1) $\llbracket v^{(n)} \rrbracket \rho = \pi^n(\rho(v))$
2) $\llbracket (x(y))^{(n)} \rrbracket \rho = \pi^n(\llbracket x \rrbracket \rho(\llbracket y \rrbracket \rho))$
3) $\llbracket (\lambda v.x)^{(n)} \rrbracket \rho = \pi^n(\lambda \beta \in D_\infty: \llbracket x \rrbracket \rho[v/\beta])$
We define several auxiliary functions:

**type:** $\Lambda^t \rightarrow \mathbb{N}$ is a mapping.

1) $\text{type}(v(n)) = n$
2) $\text{type}((xy)(n)) = n$
3) $\text{type}((\lambda v.y)(n)) = n$

**W:** $\Lambda^t \rightarrow \Lambda_\Omega$ is a mapping defined as:

1) $W(v(n)) = v$
2) $W((xy)(n)) = W(x)W(y)$
3) $W((\lambda v.z)(n)) = \lambda v.W(z)$
4) $W(\Omega(n)) = \Omega$

i.e. $W(x)$ is the $\lambda$-$\Omega$-expression obtained from $x \in \Lambda^t$ by deleting all type superfixes of $x$.

**T:** $\Lambda_\Omega \rightarrow P(\Lambda^t)$ (power set of $\Lambda^t$) is defined by:

$$T(x) = \{ y \mid x = W(y) \} \subseteq \Lambda^t,$$

i.e. $T(x)$ is the set of all typed $\lambda$-expressions generated from $x$ by putting a type superfix to each subexpression of $x$.

2.3.9 **Lemma.** $x = U\bar{\epsilon}(x)$.

**Proof.** See [23]. □

**Notation**

For $x \in \Lambda^t$ and $n \in \mathbb{N}$, let $[x]_n$ be the typed $\lambda$-expression determined by the following rule:

$$[x]_n = x \quad \text{if} \quad \text{type}(x) \leq n$$

$$[x]_n = y^{(n)} \quad \text{if} \quad \text{type}(x) > n,$$

where $x \equiv y^{(m)}$ for $m = \text{type}(x)$. 
Typed Conversions

In the similar manner to the conversion rules in the ordinary \(\lambda\)-calculus, we define typed conversion rules for \(\Lambda^t\).

2.3.10 Definition (Typed Substitution). For \(v \in V\) and \(x, y \in \Lambda^t\), we define \(\int^v y x\) to be:

1) If \(x = u^{(n)}\) for \(u \in V\), \(u \neq v\), then \(\int^v y x = u^{(n)}\).
2) If \(x = v^{(n)}\), then \(\int^v y x = [y]_n\).
3) If \(x = \Omega^{(n)}\), then \(\int^v y x = \Omega^{(n)}\).
4) If \(x = (ts)^{(n)}\), then \(\int^v y x = [(\int^v y t)(\int^v y s)]^{(n)}\).
5) If \(x = (\lambda u.w)^{(n)}\) for \(u \neq v\), then \(\int^u y x = [\lambda u.([\int^v y w]^{(n)}\)) if \(u\) does not occur free in \(w\). If \(u\) occurs free in \(w\), \(\int^u y x = [\lambda u'.([\int^v y w']^{(n)}\)) where \(w'\) is \(w\) with each \(u\) replaced by \(u'\) and \(u'\) is the first variable other than \(u\) or \(v\) in \(U\) such that \(u'\) does not occur free in \(w\) or \(y\).
6) If \(x = (\lambda v.w)^{(n)}\), then \(\int^v y x = (\lambda v.w)^{(n)}\).

2.3.11 Lemma (Wadsworth). For \(v \in V\), \(x, y \in \Lambda^t\) and \(\rho \in EN\),

\[ U[I_\int^v y x]^{(n)} \rho = U[I_x]^{(n)} \rho[V\cup U[I_y]^{(n)} \rho].\]

Proof. See [23]. \(\square\)

Typed \(\beta\)-Reduction

\(((\lambda v.x)^{(i)} y)^{(j)} \xrightarrow{t_\beta} \int^v [y]_{i-1} x]_{\min(i-1,j)}\) if \(i > 0\)
$$((\lambda v.x)(0)y)^j \stackrel{t^\beta}{\longrightarrow} [\int^v_\Omega(0)x]_0$$

We extend $t^\beta$ in the same manner as the non-typed case, i.e., $x \stackrel{t^\beta}{\longrightarrow} y$ if $y$ is the result of applying several typed $\beta$-reductions to some subexpressions of $x$.

**Typed $\alpha$-conversion**

Given $(\lambda v.x)^{(n)} \in \Lambda^t$ and let $v_1, v_2, \ldots, v_k$ be all the occurrences of $v$ in $x$. Then

$$(\lambda v.x)^{(n)} \stackrel{t^\alpha}{\longrightarrow} [\lambda u. \int^v_{(\max[n_i])} x]^{(n)}$$

For $x, y \in \Lambda^t$, $x \stackrel{t^\alpha}{\longrightarrow} y$ if $y$ is derived from $x$ by applying several typed $\alpha$-conversions to some subexpressions of $x$.

**Typed $\eta$-abstraction**

Let $\text{type}(x) = n$ for $x \in \Lambda^t$.

$$x \stackrel{t_n\text{-ab}}{\longrightarrow} (\lambda t.(x^t(n-1))(n-1))(n) \quad \text{if} \quad n \geq 1$$

$$x \stackrel{t_n\text{-ab}}{\longrightarrow} (\lambda t.(x^\Omega(0))(0))(0) \quad \text{if} \quad n = 0$$

For $x, y \in \Lambda^t$, $x \stackrel{t_n}{\longrightarrow} y$ if $y$ is derived from $x$ by applying several typed $\eta$-abstraction to some subexpressions of $x$.

2.3.12 Theorem (Wadsworth). For $x, y \in \Lambda^t$, if $x$ and $y$ are type-convertible to each other, then $x = y$.

Proof. See [23]. □
2.3.13 **Lemma** (Wadsworth). Given any $x \in \Lambda^t$, then there is a typed $\beta$-reduction sequence: $x = x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_n$ such that $x_n$ has no typed $\beta$-redex.

**Proof.** See [23]. □

2.3.14 **Lemma** (Wadsworth). Let $x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_n$ be a sequence of typed $\beta$-reductions for $x_1, x_2, \ldots, x_n \in \Lambda^t$. Then there exists an ordinary $\beta$-reduction sequence $y_1 \rightarrow y_2 \rightarrow \cdots \rightarrow y_n$ where $y_i$ matches $W(x_i)$ except at occurrences of $\Omega$ in $W(x_i)$.

**Proof.** See [23]. □

**Notion of Reduced Approximant**

Given $x \in \Lambda$, $\varepsilon \in \Lambda_\Omega$ is said to be a **direct approximant** of $x$ if $\varepsilon$ has no $\beta$-redex and $\varepsilon$ matches $x$ except at occurrences of $\Omega$ in $\varepsilon$. For example, a $\lambda$-$\Omega$-expression, $\varepsilon$, that is obtained from $x$ by replacing each $\beta$-redex in $x$ by $\Omega$ is a direct approximant of $x$.

$\varepsilon \in \Lambda_\Omega$ is said to be a **reduced approximant** of $x$ if $\varepsilon$ is a direct approximant of $x$ itself or of some $y \in \Lambda$ that is $\beta$-reducible from $x$ (i.e. there is a $\beta$-reduction sequence from $x$ to $y$).

For $x \in \Lambda$, we denote the set of all reduced approximants of $x$ by $A(x)$.

2.3.15 **Theorem** (Wadsworth). For any $x \in \Lambda$, $x = \bigcup_{D_{\infty}} A(x)$.

**Proof.** See [23]. □
From this theorem, the following theorem is directly deduced.

2.3.16 **Theorem (Wadsworth).** If \( x \in \Lambda \) is not head normal, \( x = \bot \).

**Proof.** Since any \( y \in \Lambda \) that is \( \beta \)-reducible from \( x \) is in a form \( \lambda s_1 s_2 \cdots s_m.(\lambda v.w)x_1 x_2 \cdots x_n \), its direct approximant is:

\[
\lambda s_1 s_2 \cdots s_m.\Omega x_2 \cdots x_n + \Omega
\]

So \( A(x) = \{ \Omega \} \). Thus \( x = \bot \). \( \square \)

Conversely

2.3.17 **Theorem (Wadsworth).** If \( x \in \Lambda \) is head normal, \( x \neq \bot \).

**Proof.** Let \( x = \lambda s_1 s_2 \cdots s_m.\nu x_1 x_2 \cdots x_n \) be a head normal form of \( x \). Let \( y = \lambda r_1 r_2 \cdots r_m.I \). Then

\[
\int_y x s_1 s_2 \cdots s_m \xrightarrow{\beta} I \neq \bot \quad D_\infty
\]

So under some environment \( \rho \), \( W[[x]] \rho = \bot \). \( \square \)

2.3.18 **Corollary (Wadsworth).** For \( x \in \Lambda \), \( x = \bot \) if and only if \( x \) has no head normal form. \( \square \)

This corollary implies that we can replace any non-head normal subexpression of \( x \in \Lambda \) by \( \text{spl} \) without affecting the \( D_\infty \)-value of \( x \) since \( \text{spl} = \Omega \). Hereafter we take the following convention. If \( \Omega \) is regarded as a member of \( \Lambda \), it stands for \( \text{spl} \).
We introduce the following non-effective conversion rule to $\Lambda$.

**$\Omega$-conversion**

$$(x,y) \in R_\Omega \text{ or } x \xrightarrow{\Omega} y \text{ if } y \text{ derives from } x \text{ by replacing some subexpressions of } x \text{ that have no head normal form by } \Omega.$$ We define $\approx$ as the reflective, transitive and symmetric closure of $R \cup R_\beta \cup R_\xi \cup R_\Omega$. We conclude that

$$CNV \frac{\approx \neq}{\neq} D_\infty.$$

$CNV \frac{\approx \neq}{\neq}$ since, for example, $(\lambda x.xx)(\lambda x.xx) \approx (\lambda x.xxx)(\lambda x.xxx)$ though it is not that $(\lambda x.xx)(\lambda x.xx) \xrightarrow{CNV} (\lambda x.xxx)(\lambda x.xxx)$.

On the other hand $\frac{\approx \neq}{\neq}$ since $I = J$ although it is not the case that $I \approx J$. $\frac{\approx \neq}{\neq}$ will be characterized in Chapter 4.
CHAPTER 3
INFINITE NORMAL FORMS FOR $\lambda$-CALCULUS

We formalize the <Infinite Expansions> of programs in the domain of the $\lambda$-expression -- called C-function. We show that the C-function can be regarded as an extension of the conventional normal forms. Böhms Theorem on the normal expressions is extended to the general expressions via the C-function.
§1. Pedigree

First, we introduce an infinite set which will be used to characterize the behavior of \( \lambda \)-expressions.

3.1.1 Definition. Pedigree, \( \Delta \), is the set \( \{0\} \cup \{(n_1, n_2, \ldots, n_k) | k, n_j \in \mathbb{N}\} \). There is a natural partial order, \(<\), in \( \Delta \) defined by:

for \( \delta_1, \delta_2 \in \Delta \), \( \delta_1 < \delta_2 \) if and only if

either 1. \( \delta_1 = 0 \)

or 2. \( \delta_1 = (m_1, m_2, \ldots, m_i) \) and \( \delta_2 = (n_1, n_2, \ldots, n_j) \)

where \( i < j \) and \( m_1 = n_1, \ldots, m_i = n_i \).

We say \( \delta_1 < \delta_2 \) if \( \delta_1 = \delta_2 \) or \( \delta_1 < \delta_2 \), i.e. \( \leq \) means "is a prefix of." 

3.1.2 Definition. Given \( \delta \in \Delta \), length of \( \delta \), is defined by:

\[
|\delta| = 0 \text{ if } \delta = 0 \\
|\delta| = k \text{ if } \delta = (n_1, n_2, \ldots, n_k) .
\]

3.1.3 Definition. Map \( \text{Pr}: \Delta \rightarrow \Delta \) is defined by:

\[
\text{Pr}(\delta) = \begin{cases} 
\text{undefined} & \text{if } \delta = 0 \\
0 & \text{if } |\delta| = 1 \\
(n_1, n_2, \ldots, n_{k-1}) & \text{if } \delta = (n_1, n_2, \ldots, n_k) .
\end{cases}
\]

3.1.4 Definition. Given \( \delta \in \Delta \) and a positive integer \( m \),

\[
\delta \cdot m = (m) \quad \text{if } \delta = 0 \\
\delta \cdot m = (n_1, n_2, \ldots, n_k, m) \quad \text{if } \delta = (n_1, n_2, \ldots, n_k) .
\]
§2. Idea of Infinite Expansion

Before going to the formal definition of C-functions, we try here to illustrate informally the idea of infinite normal form.

3.2.1 Example. $R$ is defined by $Y(\lambda f \lambda x \lambda y. x(fy)((\lambda x.xx)(\lambda x.xxx)))$ where $Y$ is the fixed point operator. Since we know that $(\lambda x.xx)(\lambda x.xxx)$ does not have a head normal form, we replace it by $\Omega$. The recursive definition of $R$ is:

$$R \xrightarrow{B} \lambda x \lambda y. x(Ry)(Ry\Omega)$$

Arrange it in the form:

```
\lambda x \lambda y. x
  \ \
Ry       Ry\Omega
```

with operands below the leading operator. Renaming the bound variables according to their position:

```
\lambda t_1 \lambda t_2. t_1
  \ \
Rt_2       Rt_2\Omega
```

$Rt_2 \xrightarrow{B} \lambda y. t_2(Ry)(Ry\Omega)$ and, so, the left sub-tree is depicted as:

```
\lambda t_{11}. t_2
  \ \
Rt_{11}       Rt_{11}\Omega
```

$Rt_{2\Omega} \xrightarrow{B} t_2(R\Omega)(R\Omega\Omega)$, so the right sub-tree is depicted as:
However, since $R\Omega \xrightarrow{\beta} \Omega$, this sub-tree becomes:

Applying $\beta$-reductions further, we have

Now, several applications of $\eta$-abstraction lead us to:
In this way, we can expand any expression infinitely by applying \( \eta \)-abstractions and \( \beta \)-reductions. In the illustrations, the arrangement of the head normal forms should be noted. Each head variable is situated higher than its operands since it is dominant over them. This situation is similar to the program of the form

\[
\text{begin A;B end. A can be said to be dominant over B since execution may never reach B depending on the control structure in A. This point will be further discussed in Chapter 5.}
\]

Here note that there are four operations involved in the process of expansion:

a) \( \beta \)-reduction  

b) \( \Omega \)-conversion  

c) Renaming the bound variables according to their position  

d) \( \eta \)-abstraction  

Also note that there are two important bases to consider this process.

1) We consider the \( \lambda \)-calculus with \( \eta \)-convertibility. Operation d) depends upon this assumption.

2) The head normality is an undecidable property. Thus operation b) is not effective and the functions and \( L, \hat{C} \) and \( C \) to be defined in this chapter are non-computable.
§3. **L-function**

To make the argument easier in the rest of this chapter, we make the following conventions:

Let $U$ be the enumerably infinite set of the variables. We take two mutually disjoint subsets $F$ and $T_\Delta$ of $U$ and set $V = F \cup T_\Delta$, where

$$F = \{f_i | i = 1, 2, \ldots\}$$

and

$$T_\Delta = \{t_\delta | \delta \in \Delta - \{0\}\}.$$

We assume, in the rest of this chapter, that *if any given expression has some occurrences of a free variable, it is one of the $f_i$'s in $F$. Our intention is to convert any given expression into one whose bound variables are in $T_\Delta$ by applying $\alpha$-conversions. We will be using $z$ to represent a variable which is either in $F$ or $T_\Delta$.

Let $\Sigma = \{(z, m, n) | z \in U, m, n \in \mathbb{N} \cup \{0\}\}$ and $\Omega$ be a symbol not in $\Sigma$.

An auxiliary function $h: \Lambda \rightarrow \Sigma \cup \{\Omega\}$ is defined by:

$$h(x) = \begin{cases} 
\Omega & \text{if } x \text{ has no head normal form} \\
(z, m, n) & \text{if } x \xrightarrow{\beta} \lambda x_1 \ldots x_m. z x_1 \ldots x_n 
\end{cases}$$

It is easy to see that $h$ is well-defined. Note that $h$ is not a computable function since the existence of a head normal form is not recursively decidable.

3.3.1 **Definition (L-function).** We define $L: \Lambda \rightarrow (\Delta + \Lambda)$ inductively as follows:

Given $x \in \Lambda$, assume that any $t_\delta$ in $T_\Delta$ does not appear in $x$ (by applying $\alpha$-conversions if necessary).
Step 0. 
\[ L(x,0) = \begin{cases} 
\Omega & \text{if } h(x) = \Omega \text{ (Operation b)} \\
\lambda t_1 \cdots t_m z X_1 \cdots X_n & \text{if } h(x) \neq \Omega \text{ (Operation a, c)}
\end{cases} \]

and \[ x \xrightarrow{\beta h} \lambda s_1 s_2 \cdots s_m x_1 x_2 \cdots x_n \]

and \[ z X_1 x_2 \cdots x_n = \int \lambda s_1 s_2 \cdots s_m x_1 x_2 \cdots x_n \]

Step 0. Suppose that we have defined \( L(x, \delta') \) for all \( \delta' \in \Delta \) such that \( \delta' \leq \delta \). We are to define \( L(x, \delta^o i) \) for each \( i \in \mathbb{N} \).

Case I. If \( L(x, \delta) = \Omega \) then \( L(x, \delta^o i) = \Omega \) for all \( i \in \mathbb{N} \).

Case II. If \( L(x, \delta) = \lambda t_{\delta^o i_1} t_{\delta^o i_2} \cdots t_{\delta^o i_m} z X_1 x_2 \cdots x_n \) then

(i) If \( i \leq n \) then
   (a) \( L(x, \delta^o i) = \Omega \) if \( h(x_i) = \Omega \) (Operation b)
   (b) \( L(x, \delta^o i) = \lambda t_{\delta^o i_1} t_{\delta^o i_2} \cdots t_{\delta^o i_m} u Y_1 Y_2 \cdots Y_q \) (Operation a, c)

   and \[ x_i \xrightarrow{\beta h} \lambda r_1 r_2 \cdots r_p u Y_1 Y_2 \cdots Y_q \]

   and \[ z Y_1 Y_2 \cdots Y_q = \int \lambda r_1 r_2 \cdots r_p u Y_1 Y_2 \cdots Y_q \]

(ii) If \( i > n \) then \( L(x, \delta^o i) = t_{\delta^o (m-n+i)} \) (Operation d)

We should note that in (ii) of Case II above that we are applying \( \eta \)-abstractions. Also each head variable of \( L(x, \delta) \) is in \( F \) if it is free in \( x \) or in \( T_\Delta \) if it is bound in \( x \).
3.3.2 Example. We look at $L(R,\delta)$ for $R$ defined in §2.

\[
\begin{align*}
L(R,0) &= \lambda t_1 \lambda t_2 . t_1 (Rt_2)(Rt_2\Omega) \\
L(R,1) &= \lambda t_11 . t_2 (Rt_11)(Rt_11\Omega) \\
L(R,2) &= t_2 (R\Omega)(R\Omega\Omega) \\
L(R,3) &= t_3 \\
&\vdots \\
L(R,i) &= t_i \\
&\vdots \\
L(R,11) &= \lambda t_{111} . t_{111} (Rt_{111})(Rt_{111}\Omega) \\
L(R,12) &= t_{11}\Omega \\
L(R,13) &= t_{12} \\
L(R,14) &= t_{13} \\
&\vdots \\
L(R,1i) &= t_1(i+1) \\
&\vdots \\
L(R,21) &= \Omega \\
L(R,22) &= \Omega \\
L(R,23) &= t_{21} \\
&\vdots \\
L(R,2i) &= t_2(i-2) \\
&\vdots \\
\end{align*}
\]

3.3.3 Corollary. If $x \approx y$, then $L(x,\delta) \approx L(y,\delta)$ for all $\delta \in \Delta$. 
§4. C-functions

Now we are ready to present the definition of C-functions, \( \hat{C} \) and \( C \). \( \hat{C} \) is not essentially necessary to state our result, but, \( \hat{C} \) gives a way to simplify our discussion.

3.4.1 Definition (\( \hat{C} \)-function). Let \( S = \{(z,k) | k \in \mathbb{Z}, z \in \mathcal{F}_\Delta \} \cup \{\omega\} \). We define \( \hat{C} : \Lambda \to (\Delta + S) \) by:

\[
\hat{C}(x,\delta) = \begin{cases} 
\omega & \text{if } L(x,\delta) = \Omega \\
(z,k) & \text{if } h(L(x,\delta)) = (z,m,n) \text{ and } k = m - n .
\end{cases}
\]

We note that if \( \hat{C}(x,\delta) = (z,k) \) for \( x \in \Lambda \) and \( \delta \in \Delta \), then there exists a positive integer \( M \) such that

\[
\hat{C}(x,\delta^\cdot N) = (t_{N+k},0) \text{ for all } N \geq M.
\]

Although \( \hat{C} \) is defined as above, is the second component, \( k \), of \( C(x,\delta) \) is not necessary to uniquely specify a \( \lambda \)-expression. Thus we define \( C \), simplified version of \( \hat{C} \).

3.4.2 Definition (C-function). \( C \) is a function \( \Lambda \to (\Delta + V \cup \{\omega\}) \) defined by:

\[
C(x,\delta) = \begin{cases} 
z & \text{if } \hat{C}(x,\delta) = (z,k) \\
\omega & \text{if } \hat{C}(x,\delta) = \omega .
\end{cases}
\]

3.4.3 Corollary. If \( x \approx y \), then \( \hat{C}(x) = \hat{C}(y) \) and \( C(x) = C(y) \).

We will now state the theorem which plays the central role in this thesis. Proof of this theorem was essentially given in Böhm [4] (also in Wadsworth [21]), however since the arguments used in the
proof are fundamental in this thesis and that it is not yet widely
published, we present a complete proof for the theorem, hope-
fully, with notational improvements.

Firstly, the theorem is stated using \( \hat{C} \) and thereafter, it
will be modified for \( C \).

3.4.5 Lemma. Given \( x \) and \( y \) in \( \Lambda \), suppose that, for \( \delta_0 \in \Delta \),
\( \hat{C}(x,\delta) = \hat{C}(y,\delta) \) for all \( \delta \) such that \( \delta < \delta_0 \), and that
\[
\hat{C}(x,\delta_0) = (u,i) \quad \hat{C}(y,\delta_0) = (v,j)
\]
where \( (u,i) \neq (v,j) \). Then for arbitrary \( a, b \in D_\infty \) we can
choose \( e_1, e_2, \ldots, e_n \in \Lambda \) and an environment \( \rho \) for which
\[
V Ixe_1e_2\cdots e_n \rho = a \\
V Iye_1e_2\cdots e_n \rho = b
\]
Moreover, if \( a, b \in \Lambda_C \), we can choose \( \rho \) so that \( \rho(V) \subseteq \Lambda_C \).

3.4.6 Lemma. Given \( x \) and \( y \) in \( \Lambda \), suppose that, for
\( \delta_0 \in \Delta \), \( \hat{C}(x,\delta) = \hat{C}(y,\delta) \) for all \( \delta \) such that \( \delta < \delta_0 \) and that
\( \hat{C}(x,\delta_0) = \omega \) and \( \hat{C}(y,\delta_0) \neq \omega \). Then, for arbitrary \( a \) in \( D_\infty \),
we can choose \( e_1, e_2, \ldots, e_n \in \Lambda \) and an environment \( \rho \) for which
\[
V Ixe_1e_2\cdots e_n \rho = \bot \\
V Iye_1e_2\cdots e_n \rho = a
\]
Moreover, if \( a \in \Lambda_C \), we can choose \( \rho \) so that \( \rho(V) \subseteq \Lambda_C \).
We prove only Lemma 3.4.5. The proof for Lemma 3.4.6 is straightforward from the proof given for Lemma 3.4.5.

Proof of Lemma 3.4.5.

Case I. $\delta_0 = 0$: Since $(u,i) \neq (v,j)$, $u \neq v$ or $i \neq j$.

Case a. $u \neq v$: By the definition of $\hat{C}$, we conclude that

$$L(x,0) = \lambda t_1 t_2 \cdots t_m u x_1 \cdots x_n$$
$$L(y,0) = \lambda t_1 t_2 \cdots t_p v y_1 \cdots y_q$$

Take a positive integer $K$ such that $K > \max(m,p) + 1$. Then

$$L(x,0)t_1 t_2 \cdots t_K \xrightarrow{\text{CNV}} u x_1 x_2 \cdots x_n t_{m+1} \cdots t_K$$
$$L(y,0)t_1 t_2 \cdots t_K \xrightarrow{\text{CNV}} v y_1 y_2 \cdots y_q t_{p+1} \cdots t_K$$

Note that $u, v \in V = \text{FUT}$.

Now take an environment, $\rho$, so that

$$\rho(u) = \lambda s_1 s_2 \cdots s_{K-m+n} s_k$$
$$\rho(v) = \lambda s_1 s_2 \cdots s_{K-p+q} s_k$$
$$\rho(t_K) = a$$
$$\rho(t_{K-1}) = b$$

Then

$$\text{V}[[xt_1 t_2 \cdots t_K]] \rho = a$$
$$\text{V}[[yt_1 t_2 \cdots t_K]] \rho = b$$

Case b. $u = v$ and $i \neq j$: We can assume, without loss of generality, that $i < j$. As in Case a,
Let $K$ be a positive number such that
\[ K > \max(m, p, j+n) \]

By substituting $\lambda s_1 s_2 \cdots s_{K-j} s_{K-j}$ for $u$ we have
\[ L(x,0)t_1 t_2 \cdots t_K \xrightarrow{\text{CNV}} uX_1 X_2 \cdots X_{m+1} \cdots t_K \]
\[ L(y,0)t_1 t_2 \cdots t_K \xrightarrow{\text{CNV}} uY_1 Y_2 \cdots Y_{p+1} \cdots t_K \]

since $K-j > n$.

Now we choose $\rho$ such that
\[ \rho(u) = \lambda s_1 s_2 \cdots s_{K-j} s_{K-j} \]
\[ \rho(t_K) = b \]
\[ \rho(t_{K-j+1}) = \lambda s_1 s_2 \cdots s_{j-i} a \]

and we have
\[ \forall [xt_1 t_2 \cdots t_K] \rho = a \]
\[ \forall [yt_1 t_2 \cdots t_K] \rho = b \]

**Case II.** $\delta_0 > 0$: Let $\delta_0 = (d_1, d_2, \ldots, d_L)$ and set
$\delta^0 = 0, \delta^1 = (d_1), \ldots, \delta^L = (d_1, \ldots, d_L), \ldots, \delta^L = \delta_0$ (set $d_0 = 0$ for convenience). By the assumption of the theorem, for $0 \leq \ell \leq L-1$,
\[ \hat{C}(x, \delta^\ell) = \hat{C}(y, \delta^\ell) = (z_\ell, k_\ell) \]
for some \((z_k, k_k) \in S\) and
\[
\hat{C}(x, \delta_0) = (u, i) \\
\hat{C}(y, \delta_0) = (v, j).
\]

The proof technique for Case II is the following:

Due to the fact that \(\hat{C}(x, \delta) = \hat{C}(y, \delta)\) for \(\delta < \delta_0\), we can choose a context under which \(L(x, \delta_0)\) and \(L(y, \delta_0)\) can be 'dug out' of \(x\) and \(y\), respectively. Thereafter the problem is reduced to the difference between \(L(x, \delta_0)\) and \(L(y, \delta_0)\) which, applying the technique used in Case I, leads us to the conclusion of the Lemma.

Case II-1. Suppose that all \(z_k\)'s are distinct, i.e. for \(\ell_1 \neq \ell_2\), \(z_\ell_1 \neq z_\ell_2\): Let, for \(\ell = 0, \ldots, L-1,\)
\[
L(x, \delta^\ell) = x_1^{\delta_{01}} x_2^{\delta_{02}} \cdots x_n^{\delta_{0n}}
\]
and
\[
L(y, \delta^\ell) = y_1^{\delta_{01}} y_2^{\delta_{02}} \cdots y_n^{\delta_{0n}}
\]
where \(m_\ell - n_\ell = p_\ell - q_\ell\) by the assumption of the lemma. Let \(k_\ell = m_\ell - n_\ell\) and take a positive integer \(K\) such that
\[
K > \max_{\ell=0, \ldots, L-1} (m_\ell, p_\ell, k_\ell + d_\ell + 1)
\]

If we substitute \(s_1 s_2 \cdots s_{K-k_\ell} s_{d_\ell + 1}\) for \(z_\ell\)
\[
L(x, \delta^\ell) t^{\delta_{01}} t^{\delta_{02}} \cdots t^{\delta_{0K}} \rightarrow_{\text{CNV}} L(x, \delta^\ell+1) \\
L(y, \delta^\ell) t^{\delta_{01}} t^{\delta_{02}} \cdots t^{\delta_{0K}} \rightarrow_{\text{CNV}} L(y, \delta^\ell+1)
\]
Thus by induction on \( l \), if we substitute 
\[ Q = \lambda s_1 s_2 \cdots s_{k-1} s_d \text{ for } z_k \] 
for \( z_k \) for \( l = 0,1,...,L-1 \), we have

\[
\int_{Q}^Z t_1 t_2 \cdots t_{k-1} t_k t_{k+1} \cdots t_{L-1} t_L \xrightarrow{\text{CNV}} \int_{Q}^Z \delta o_{k} \delta o_{k+1} \delta o_{k+2} \delta o_{L-1} \delta o_{L}
\]

Combining this result with Case I, we conclude that the lemma holds for Case II-1.

**Case II-2.** \( z_k, l = 0,1,2,...,L-1 \), are not necessarily distinct:

We should note that the proof technique used in Case II-1 is no longer valid here since it is not possible to substitute different combinators for \( z_k \)'s. To get around this difficulty, we introduce a combinator \( R^k = \lambda s_1 s_2 \cdots s_{k-1} s_d \). Roughly speaking, we will substitute \( R^k \) for each \( z_k \) in \( x \) and \( y \) so that \( x \) and \( y \) will have distinct head variables after the substitutions.

Before we start working on \( x \) and \( y \), we give two observations for \( R^k \).

**Claim 1.** Let \( S, T \in \mathcal{A} \). If \( S \sim T \), then, for a sufficiently large \( K \), \( \int_{R^k}^Z S \sim \int_{R^k}^Z T \) where \( z \) is a free variable appearing in \( S \) and \( T \).

**Proof.** Suppose that neither \( S \) nor \( T \) has a head normal form. Then obviously neither \( \int_{R}^Z S \) nor \( \int_{R}^Z T \) has a head normal form.

Suppose \( S \) and \( T \) have a head normal form and (after several \( \alpha \)-conversions)
\[ S = \lambda r_1 r_2 \cdots r_m w_{S_1 S_2} \cdots S_n \]
\[ T = \lambda r_1 r_2 \cdots r_p w_{T_1 T_2} \cdots T_q \]

where \( m - n = p - q \).

**Case i.** \( z \neq w \):

\[ \int_{R_K}^Z S = \lambda r_1 r_2 \cdots r_m w_{S_1 S_2} \cdots S_n \]
\[ \int_{R_K}^Z T = \lambda r_1 r_2 \cdots r_p w_{T_1 T_2} \cdots T_q \]

where \( S_j = \int_{R_K}^Z S_j \) and \( T_j = \int_{R_K}^Z T_j \). Thus \( \int_{R_K}^Z S \sim \int_{R_K}^Z T \).

**Case ii.** \( z = w \): We take \( K \) so that \( K > \max(n, q) \). Then

\[ \int_{R}^Z S = \lambda r_1 r_2 \cdots r_m (\lambda s_1 s_2 \cdots s_K s_1 s_2 \cdots s_{K-1}) S_1 S_2 \cdots S_n \]
\[ \xrightarrow{\text{CNV}} \lambda r_1 r_2 \cdots r_m s_{n+1} s_{n+2} \cdots s_K s_1 s_2 \cdots s_{n+1} \cdots s_{K-1} \]

In the same manner

\[ \int_{R}^Z T \xrightarrow{\text{CNV}} \lambda r_1 r_2 \cdots r_p s_{q+1} \cdots s_K s_{T_1 T_2} \cdots T_{q+1} \cdots s_{K-1} \]

From \( m - n = p - q \), we conclude that

\[(m + K - n) - (K - 1) = (p + K - q) - (K - 1) \]

Thus \( \int_{R}^Z S \) and \( \int_{R}^Z T \) have the same index. Also the heads in \( \int_{R}^Z S \) and \( \int_{R}^Z T \) are \( K - m + n = K - p + q \th \) bound variable in both.
Thus \[ \int_{R_K}^Z S \sim \int_{R_K}^Z T. \]

Claim 2. If \( S \vdash T \), then for a sufficiently large \( K \),
\[ \int_{R_K}^Z S \vdash \int_{R_K}^Z T \]
for a free variable \( z \) in \( S \) and \( T \).

Proof. Suppose that \( S \) has a head normal form and \( T \) does not.
Then as we have seen in the proof of Claim 1, \[ \int_{R_K}^Z S \] has a head normal form for a sufficiently large \( K \), but \[ \int_{R_K}^Z T \] does not have a head normal form. Thus
\[ \int_{R_K}^Z S \vdash \int_{R_K}^Z T \]

On the other hand, suppose
\[
S = \lambda r_1 r_2 \cdots r_m w_1 S_1 S_2 \cdots S_n \quad T = \lambda r_1 r_2 \cdots r_p w_2 T_1 T_2 \cdots T_q
\]
where \((m-n, w_1) \neq (p-q, w_2)\). If \( w_1 \neq z \) and \( w_2 \neq z \), obviously \( \int_{R_K}^Z S \vdash \int_{R_K}^Z T \) since the substitution does not alter the heads nor the indices. If \( w_1 \neq w_2 \) and \( w_1 = z \), then we take \( K \) so that \( K > \max(n, p+n-m) \). As above,
\[
\int_{R_K}^Z S \xrightarrow{\text{CNV}} \lambda r_1 r_2 \cdots r_m s_{n+1} \cdots s_{K-1} s_{K+1} s_{K+2} \cdots s_{n+s_{n+1} + \cdots + s_{K-1}}
\]
while
\[
\int_{R_K}^Z T = \lambda r_1 r_2 \cdots r_p w_2 T_1 T_2 \cdots T_q
\]
The head $s_K$ in $\int_{R_K}^Z S$ is the $K-n+m \text{th}$ bound variable but since $K-n+m > p$, $w_2$ cannot be $K-n+m \text{th}$ bound variable in $\int_{R_K}^Z T$.

Thus

$$\int_{R_K}^Z S \neq \int_{R_K}^Z T$$

On the other hand, if $w_1 = w_2 = z$ and $m-n \neq p-q$, then we take $K$ such that $K > \max(n,q)$. It is easy to see that $\int_{R_K}^Z S$ and $\int_{R_K}^Z T$ are different in their indices. □

Now we are ready to prove our lemma for Case II-2. We determine $K$ of $R_K$ in the following way: Let

$$L(x,\delta^k) = \lambda t_{\delta^k l_1} t_{\delta^k l_2} \cdots t_{\delta^k m_2} \cdot u_{\delta^k l_1} X_1X_2 \cdots X_{n_k}$$

$$L(y,\delta^k) = \lambda t_{\delta^k l_1} t_{\delta^k l_2} \cdots t_{\delta^k q_2} \cdot v_{\delta^k l_1} Y_1Y_2 \cdots Y_{q_k}$$

for $k = 0,1,\ldots,L$. We take $K$ such that $K > \max(n_k,q_k,\delta_k)+1$.

Step 0. We want a context to select $L(x,\delta^i)$ and $L(y,\delta^i)$ out of $x$ and $y$, respectively. Let

$$L(x,0) = \lambda t_1 t_2 \cdots t_{m_0} z_0 X_1X_2 \cdots X_{n_0}$$

$$L(y,0) = \lambda t_1 t_2 \cdots t_{p_0} z_0 Y_1Y_2 \cdots Y_{q_0}$$

where $m_0 - n_0 = p_0 - q_0$. Substituting $R_K$ for $z_0$ in
\[ L(x,0)t_1t_2 \cdots t_H \] and \[ L(y,0)t_1t_2 \cdots t_H \] where \[ \max(m_o, p_o, d_1 + m_o - n_o) < H < K + m_o - n_o, \] we have

\[ X_0 = \lambda^{s_{H-m_o+n_o+1}} \cdots s_{K-1} \] from \[ L(x,0)t_1t_2 \cdots t_H \]

\[ Y_0 = \lambda^{s_{H-p_o+q_o+1}} \cdots s_{K-1} \] from \[ L(y,0)t_1t_2 \cdots t_H \]

where \( X'_j = \int_{R_K}^{Z_0} X_j \) and \( Y'_j = \int_{R_K}^{Z_0} Y_j \).

Now substituting \( \lambda r_1r_2 \cdots r_{K-1} \cdot r_d \) for \( s_K \) in both \( X_0^{s_{H-m_o+n_o+1}} \cdots s_{K-1} \) and \( Y_0^{s_{H-p_o+q_o+1}} \cdots s_{K-1} \), we have

\[ X_0^{s_{H-m_o+n_o+1}} \cdots s_{K-1} \xrightarrow{\text{CNV}} \int_{R_K}^{Z_0} L(x, \delta^l) \]

\[ Y_0^{s_{H-p_o+q_o+1}} \cdots s_{K-1} \xrightarrow{\text{CNV}} \int_{R_K}^{Z_0} L(y, \delta^l) \]

Step \( \ell \). Suppose that, under some context, we have selected \( \int_{R_K}^{Z_0} L(x, \delta^l) \) and \( \int_{R_K}^{Z_0} L(y, \delta^l) \) out of \( x \) and \( y \) respectively. Let

\[ L(x, \delta^l) = \lambda t_{\delta^l_o1} t_{\delta^l_o2} \cdots t_{\delta^l_m\ell} z_{\ell} X_1 X_2 \cdots X_{n_{\ell}} \]

\[ L(y, \delta^l) = \lambda t_{\delta^l_o1} t_{\delta^l_o2} \cdots t_{\delta^l_p\ell} z_{\ell} Y_1 Y_2 \cdots Y_{q_{\ell}} \].
We take a positive integer, $H$, such that

$$\max(m_\ell, p_\ell, d_{\ell+1} + m_\ell - n_\ell) < H < K + m_\ell - n_\ell$$

and, substituting $R_\ell$ for $z_\ell$ in

$$L(x, \delta_\ell) t_{\delta_{\ell, 1}} t_{\delta_{\ell, 2}} \cdots t_{\delta_{\ell, H}} \text{ and } L(y, \delta_\ell) t_{\delta_{\ell, 1}} t_{\delta_{\ell, 2}} \cdots t_{\delta_{\ell, H}}$$

(if $z_\ell \neq z_k$ for any $k = 0, 1, \ldots, \ell - 1$, $z_\ell$ is already substituted by $R_\ell$), we have

$$X_\ell^* = s_{H-m_\ell+n_\ell+1} \cdots s_{K-1} s_{X_1^*} \cdots s_{X_n^*} t_{m_\ell+1} \cdots t_{H-s_{H-m_\ell+n_\ell+1}} \cdots s_{K-1}$$

$$Y_\ell^* = s_{H-p_\ell+q_\ell+1} \cdots s_{K-1} s_{Y_1^*} \cdots s_{Y_n^*} t_{p_\ell+1} \cdots t_{H-s_{H-p_\ell+q_\ell+1}} \cdots s_{K-1}$$

where

$$X_j = \int_{R_K R_K' \cdots R_K} X_j$$
$$Y_j = \int_{R_1 R_2 \cdots R_K} Y_j$$

Now, following exactly the same procedure as in Step 0, we can select $\int_{R_K R_K' \cdots R_K} L(x, \delta_{\ell+1})$ and $\int_{R_K R_K' \cdots R_K} L(y, \delta_{\ell+1})$ out of $X_\ell^*$ and $Y_\ell^*$ respectively.

By the discussion above we conclude that under some appropriate context, we can select

$$\int_{R_K R_K' \cdots R_K} L(x, \delta_0) \text{ and } \int_{R_K R_K' \cdots R_K} L(y, \delta_0)$$
out of $X$ and $Y$ respectively.

Applying Claim 2 repeatedly, we conclude that

$$
\bigvee_{z_0, z_1, \ldots, z_{L-1}} L(x, \delta_0) \bigvee_{R_k, R_{K-k}, \ldots, R_K} L(y, \delta_0)
$$

where neither of them is without a head normal form. By the result of Case I, we have proved the lemma for Case II-2. $\Box$

3.4.7 **Theorem.** Given $x$, $y$ in $\Lambda$:

1. If there exists $\delta \in \Delta$ such that, for different $u$, $v$ in $V$,

   $$C(x, \delta) = u \text{ and } C(y, \delta) = v$$

   then, for arbitrarily given $a$, $b$ in $D_\omega$, we can choose $e_1, e_2, \ldots, e_n$ in $\Lambda$ and an environment $\rho$ for which

   $$W[[xe_1 e_2 \cdots e_n]] \rho = a$$
   $$W[[ye_1 e_2 \cdots e_n]] \rho = b$$

   Moreover, if $a, b \in \Lambda_c$, we can choose $\rho$ such that $\rho(V) \subseteq \Lambda_c$.

2. If there exists $\delta \in \Delta$ such that, for all $\delta_0$ satisfying $|\delta_0| < |\delta|$, $C(x, \delta_0) = C(y, \delta_0)$

   and that $C(x, \delta) = u \in V$, $C(y, \delta) = \omega$

   then, for arbitrarily given $a$ in $D_\omega$, we can choose $e_1, e_2, \ldots, e_n$ in $\Lambda$ and an environment $\rho$ for which

   $$W[[xe_1 e_2 \cdots e_n]] \rho = a$$
   $$W[[ye_1 e_2 \cdots e_n]] \rho = \perp$$
Moreover, if $a \in \Lambda_c$, we can choose $\rho$ so that $\rho(\Lambda) \subseteq \Lambda_c$.

Proof. Straightforwardly deduced from Lemma 3.4.5 and Lemma 3.4.6, using a technique similar to the proof of Corollary 3.4.9.
3.4.8 Definition. Let
\[ C = \{ c \mid c \in \Delta \rightarrow V \cup \{ \omega \} \} \]
and
\[ \hat{C} = \{ c \mid c \in \Delta \rightarrow S \} . \]

We define relation \( \preceq \) over \( C \) and \( \hat{C} \) as follows:

For \( c_1, c_2 \in C(\hat{C}) \), \( c_1 \preceq c_2 \) if and only if, for all \( \delta \in \Delta \), \( c_1(\delta) = \omega \) or \( c_1(\delta) = c_2(\delta) \).

3.4.9 Corollary. For \( x, y \in \Lambda \), \( C(x) \preceq C(y) \) if and only if \( \hat{C}(x) \preceq \hat{C}(y) \).

Proof. If \( \hat{C}(x) \preceq \hat{C}(y) \), \( C(x) \preceq C(y) \) is immediate from the definition of \( C \).

On the other hand, assume that \( \hat{C}(x) \prec \hat{C}(y) \) does not hold. This means that there is \( \delta \in \Delta \) such that

either \underline{Case 1.} \( \hat{C}(x,\delta) = (u,i) \) and \( \hat{C}(y,\delta) = (v,j) \) where 
\[ (u,i) \neq (v,j) \]
or \underline{Case 2.} \( \hat{C}(x,\delta) \neq \omega \) and \( \hat{C}(y,\delta) = \omega \).

\( C(x) \prec C(y) \) is immediate from Case 2 by the definition of \( C \).

For Case 1, let \( \delta' \) be such that, for any \( \delta' \) satisfying \( \delta' < \delta \),

\[ \hat{C}(x,\delta') = \hat{C}(y,\delta') \]
and
\[ \hat{C}(x,\delta) = (u,i) \]
\[ \hat{C}(y,\delta) = (v,j) . \]
If \( u \neq v \), \( C(x) \nleq C(y) \) is immediate.

Suppose \( u = v \) and \( i \neq j \). Let

\[
L(x, \delta) = \lambda t_{\delta_1} t_{\delta_2} \cdots t_{\delta_m} u x_1 x_2 \cdots x_n
\]
\[
L(y, \delta) = \lambda t_{\delta_1} t_{\delta_2} \cdots t_{\delta_p} u y_1 y_2 \cdots y_q
\]

where \( m - n = i \) and \( p - q = j \). If we take \( K \) larger than \( n \) and \( q \),

\[
C(x, \delta^oK) = L(x, \delta^oK) = t_{\delta^o(K-n+m)}
\]
\[
C(y, \delta^oK) = L(x, \delta^oK) = t_{\delta^o(K-q+p)}
\]

Since \( i \neq j \), \( K - n + m \neq K - q + p \). Thus \( C(x, \delta) \nleq C(y, \delta) \). □

3.4.10 Corollary. For \( x, y \) in \( \Lambda \), \( C(x) = C(y) \) if and only if \( \hat{C}(x) = \hat{C}(y) \). □

3.4.11 Corollary. For \( x, y \in \Lambda \), if \( x \subseteq y \), then \( C(x) \leq C(y) \).

Proof. Let us negate that \( C(x) \leq C(y) \). Then there must exist \( \delta \in \Delta \) such that, for some \( u, v \in V \),

\[
\begin{align*}
C(x, \delta) &= u & \text{where } u \neq v \\
C(y, \delta) &= v \\
\end{align*}
\]

or

\[
\begin{align*}
C(x, \delta) &= u \\
C(y, \delta) &= \omega
\end{align*}
\]

In either case, there must be at least one \( \delta \in \Delta \) for which the condition of part 1 or part 2 of Theorem 3.4.7 holds. Thus by
the conclusion of the theorem, there exist $e_1, e_2, \ldots, e_n \in \Lambda$ and an environment $\rho$ such that,

either

\[ W [[x_1 e_2 \cdots e_n]] \rho = K \]

and

\[ W [[y_1 e_2 \cdots e_n]] \rho = H \]

or

\[ W [[x_1 e_2 \cdots e_n]] \rho = K \]

and

\[ W [[y_1 e_2 \cdots e_n]] \rho = \bot \]

which contradicts $x \subset y$ by Proposition 2.3.5 and Corollary 2.3.18.

\[ \square \]

3.4.12 Corollary. For $x, y \in \Lambda$, if $x = y$, then $C(x) = C(y)$.

Proof. Similar to the proof of Corollary 3.4.11. \[ \square \]

In fact, the converses of Corollary 3.4.11 and Corollary 3.4.12 are also true and will be proved in Chapter 4.

We should note here that we can further formalize $C$-functions. Since each variable $z$ in $V$ is in $F$ or $T^\Delta$, we encode $z$ as follows:

If $z = f_i$ in $F$, $\text{En}(z) = i \in \mathbb{N}$.

If $z = t_\delta$ in $T^\Delta$, $\text{En}(z) = \delta \in \Delta$.

Now the new version of $C$, $\tilde{C} : \Lambda \rightarrow (\Delta + \Delta \cup \mathbb{N} \cup \{\omega\})$ is defined by:

For $x \in \Lambda$, $\tilde{C}(x, \delta) = \text{En}(z)$ if $C(x, \delta) = z \in V$

$\tilde{C}(x, \delta) = \omega$ if $C(x, \delta) = \omega$.
Thus we can discard the notion of variables. We do not take this convention since this does not provide us with any substantial improvement other than formalism.
55. C-function as Infinite Normal Form -- Extension of Böhm's Theorem

Theorem 3.4.7 can be regarded as an extension of Böhm's Theorem [4] which is stated as follows:

3.5.1 Böhm's Theorem. Let \( x, y \) in \( \Lambda \). If \( x \) and \( y \) have different normal forms, then, for any two variables \( u, v \in V \), we can choose \( \lambda \)-expressions \( e_1, e_2, \ldots, e_n \in \Lambda \), variables \( z_1, z_2, \ldots, z_m \in V \) and closed \( \lambda \)-expressions \( h_1, h_2, \ldots, h_m \in \Lambda_c \) such that

\[
\begin{align*}
\left( \int_{h_1, h_2, \ldots, h_m} z_1, z_2, \ldots, z_m \right) x e_1 e_2 \cdots e_n & \xrightarrow{\text{CNV}} u \\
\text{and}
\end{align*}
\]

\[
\left( \int_{h_1, h_2, \ldots, h_m} z_1, z_2, \ldots, z_m \right) y e_1 e_2 \cdots e_n & \xrightarrow{\text{CNV}} v
\]

If we translate Theorem 3.4.7 into one stated in pure \( \lambda \)-calculus language:

3.5.2 Theorem. Let \( x, y \) in \( \Lambda \). If \( C(x) \neq C(y) \), then, for any \( u, v \in V \), we can choose \( \lambda \)-expressions, \( e_1, e_2, \ldots, e_n \in \Lambda \), variables \( z_1, z_2, \ldots, z_m \in V \) and closed \( \lambda \)-expressions \( h_1, h_2, \ldots, h_m \in \Lambda_c \) so that one of the following 1), 2) and 3) holds:

Let

\[
x^* = \left( \int_{h_1, h_2, \ldots, h_m} z_1, z_2, \ldots, z_m \right) x e_1 e_2 \cdots e_n
\]

and
\[ y^* = \left( \left\{ z_1, z_2, \ldots, z_m \right\}_{e_1 e_2 \cdots e_n} \right)_{h_1, h_2, \ldots, h_m} \]

1) \( x^* \xrightarrow{\text{CNV}} u \) and \( y^* \xrightarrow{\text{CNV}} v \).
2) \( x^* \xrightarrow{\text{CNV}} u \) and \( y^* \) has no head normal form.
3) \( x^* \) has no head normal form and \( y^* \xrightarrow{\text{CNV}} u \). □

The main point of the extension of Böhm's Theorem is that we are no longer concerned with conventional normal forms. Theorem 3.5.2 is a statement about general \( \lambda \)-expressions no matter whether or not they are normal. In this respect, we might as well call \( C \)-functions \langle infinite normal form\rangle or generalized normal form. Refer to [21] for an alternative extension of Böhm's Theorem.
CHAPTER 4
CHARACTERIZATION OF THE $D_\infty$-VALUES OF THE $\lambda$-EXPRESSIONS

The main result in this chapter is to characterize, using the $C$-function, the partial ordering among the $\lambda$-expressions that is induced by $D_\infty$. Namely, it is shown that, given two $\lambda$-expressions $x$, $y$, the relation $x \preceq y$ in $D_\infty$ is equivalent to $C(x) \preceq C(y)$ in the algebraic domain which includes the range of $C$. 
§1. Structural Approximation

In Chapter 2, we defined the notion of approximant of \( \lambda \)-expressions. Roughly speaking, the approximant of \( x \in A \) is obtained from \( x \) by replacing the \( \beta \)-redexes in \( x \) by \( \perp \).

In this section, we define another notion of approximation which is more closely related to the \( C \)-function. For this purpose, we need a class of subsets of \( \Delta \), called \( \Delta \)-trees.

4.1.1 Definition (\( \Delta \)-trees). A \( \Delta \)-tree, \( T \), is an infinite subset of \( \Delta \) such that

1) \( 0 \in T \).
2) If \( \delta \in T \), then \( \Pr(\delta) \in T \).
3) For all \( \delta \in T \), there exists a positive integer \( N \) such that \( \delta \circ 1, \delta \circ 2, \ldots, \delta \circ N \in T \) and \( \delta \circ K \notin T \) for all \( K > N \).

For a \( \Delta \)-tree, \( T \), we call \( N \) in 3) above \( \gamma_T(\delta) \), i.e. \( \gamma_T(\delta) = \{ \delta' | \delta' \in T, \Pr(\delta') = \delta \} \).

4.1.2 Example.

\[
\begin{array}{c}
0 \\
1 \\
11 \\
111 \\
1111 \\
11111
\end{array}
\begin{array}{c}
2 \\
12 \\
121 \\
122 \\
1211 \\
1221
\end{array}
\begin{array}{c}
22 \\
21 \\
211 \\
221 \\
2111 \\
2211
\end{array}
\]

Let \( T \) be \( \{ \alpha \in \{1,2\}^* | \alpha = \beta \circ 2 \gamma \text{ or } \beta \circ 22 \gamma \text{ for } \beta, \gamma \in \{1\}^* \} \cup \{0\} \)

\(= 0 \cup 1^* \cup 1 \circ 21^* \cup 1 \circ 221^* \).
Obviously, $T$ is a $\Delta$-tree. Here, for example, $\gamma_T(11) = 2, \gamma_T(121) = 1$.

In Chapter 3, we formulated the expansion of $\lambda$-expressions. $\Delta$-trees give in a way the opposite operation. Suppose, given the C-function of a certain $\lambda$-expression, we want to synthesize the original $\lambda$-expression from the C-function. Since the C-function contains arbitrary numbers of $\eta$-abstractions, we should restrict our attention to finite and meaningful parts of the C-function, which are represented by a $\Delta$-tree.

4.1.3 Definition. Given a $\Delta$-tree, $T$, $n \in \mathbb{N}$ and $x$ in $\Lambda$, we define $T^n(x)$ to be $T^0_{x,n}$ where $T^\delta_{x,n}$ is defined for $\delta \in T$ by:

1) If $|\delta| < n$, then
   a) if $\hat{C}(x,\delta) = \omega$, then $T^\delta_{x,n} = \Omega$.
   b) if $\hat{C}(x,\delta) = (z,k)$, then

   $$T^\delta_{x,n} = \lambda t_{\delta_0}^1 T_{\delta_0 2} \cdots T_{\delta_k}^{\hat{C}(x,\delta)+k} T_{x,n}x,x,n \cdots T_{x,n}.$$ 

2) If $|\delta| = n$, then $T^\delta_{x,n} = L(x,\delta)$, where $[\ ]$ is the Gauss notation.

4.1.4 Definition. A $\Delta$-tree, $T$, is said to be admissible to $x$ in $\Lambda$ if and only if, for all $n \in \mathbb{N}$,

$$x \xrightarrow{\alpha} T^n(x).$$

Intuitively, a $\Delta$-tree, $T$, is admissible to $x$ in $\Lambda$ if it is wide enough to cover the whole significant portion of $x$, i.e.
the significant parts which were derived by \( \beta \)-reductions rather than by \( \eta \)-abstractions.

4.1.5 Definition. Let \( x \in A \). We define \( N(x) \) to be a subset of \( A \) such that

1) \( 0 \in N(x) \)

2) Let \( \delta \in N(x) \). Then \( \delta \circ i \in N(x) \) if and only if

\[
L(x, \delta) \xrightarrow{\beta} \lambda z_1 s_2 \cdots s_m. z_1 x_2 \cdots x_n
\]

for \( i \leq n \).

We should note that \( N(x) \) may be infinite or finite. For example, if \( x \) is normal, or \( x \) is not head normal, \( \#(N(x)) \) is finite. We give \( L(x, \delta) \) a special name \( LT(x, \delta) \) if \( \delta \in N(x) \).

4.1.6 Corollary. Let \( x \in A \) and \( T \) be a \( \Delta \)-tree. Then \( T \) is admissible to \( x \) if and only if \( N(x) \subseteq T \). So, \( x \) has at least one admissible \( \Delta \)-tree. \( \Box \)

4.1.7 Corollary. Let \( T_1, T_2 \) be \( \Delta \)-trees. If \( T_1 \subseteq T_2 \) and \( T_1 \) is admissible to \( x \), so is \( T_2 \). \( \Box \)

4.1.8 Corollary. If \( T \) is a \( \Delta \)-tree admissible to \( x \), then

\[
T(x) = x \quad \text{for any} \ n.
\]

4.1.9 Corollary. If \( T \) is an admissible \( \Delta \)-tree to \( x \in A \), then \( LT(x, \delta) \prec T(x) \) for all \( LT(x, \delta) \) such that \( |\delta| = n \). \( \Box \)
4.1.10 Example. Let us look at $T$ in Example 4.1.2. It is easy to see that $T$ is admissible to $R$ in Example 3.3.2.

\[ T^3(R) = \lambda_t \cdot t_2 \cdot t_1(\lambda t_{11} \cdot t_2(\lambda t_{11} \cdot t_{11}(Rt_{111})(Rt_{111}))(t_{11} \Omega))(t_{2} \Omega) \]

\{\delta | \delta \in N(R), |\delta| = 3\} = \{111,112,121,122\} ,

so

\[
\begin{align*}
\LT(R,111) &= Rt_{111} \\
\LT(R,112) &= Rt_{111} \Omega \\
\LT(R,121) &= \Omega \\
\LT(R,122) &= \Omega .
\end{align*}
\]

Let us see how $T^n(x)$ is generated from $x$. First apply to $x$ all the $\beta$-reductions that were necessary to generate all $\LT(x,\delta)$'s for $|\delta| \leq n$ ($x \xrightarrow{\beta} Z_1$). So, for each $\LT(x,\delta)$ such that $|\delta| = n$, $\LT(x,\delta) \xrightarrow{\alpha} Z_1$. Now, apply to $Z_1$ $\alpha$-conversions so that each bound variable, which occurs outside of all $\LT(x,\delta)$ with $|\delta| = n$, will be renamed and belong to $T_{\Delta}(Z_1 \xrightarrow{\alpha} Z_2)$. Now apply $\Omega$-conversions and replace each subexpression, which is not head normal, by $\Omega$ ($Z_2 \xrightarrow{\Omega} Z_3$). Finally apply $n$-abstractions for each node in $\{\delta | \delta \in T-N(x), |\delta| \leq n\}$ ($Z_3 \xrightarrow{n-ab} T^n(x)$).

Note that no $n$-abstraction is made inside of $\LT(x,\delta)$ with $|\delta| = n$. We state the process above as a corollary:

4.1.11 Corollary. Let $x$ be in $\Lambda$ and $T$ be a $\Delta$-tree admissible to $x$. For $n \in \mathbb{N}$, there exist $Z_1, Z_2, Z_3 \in \Lambda$ such that

\[
x \xrightarrow{\beta} Z_1 \xrightarrow{\alpha} Z_2 \xrightarrow{\Omega} Z_3 \xrightarrow{n-ab} T^n(x)
\]
where \( Z_2 \) matches \( Z_3 \) except at occurrences of \( \Omega \) in \( Z_3 \) and if any of the \( n \)-abstractions applied in \( Z_3 \xrightarrow{n-ab} T^n(x) \) is of the form \( A + \lambda s.A_s \), \( A \) cannot be a proper subexpression of one of \( LT(x,\delta) \)'s, \( |\delta| = n \). Moreover, if \( Z_2 \) has any \( \beta \)-redex, it must be contained in either \( LT(x,\delta) \) with \( |\delta| = n \), or in one of the subexpressions which were converted to \( \Omega \) in \( Z_2 \xrightarrow{\Omega} Z_3 \).

4.1.12 Definition (Structural Approximation). Given \( x \) in \( \Lambda \), \( n \in \mathbb{N} \) and a \( \Delta \)-tree, \( T \), which is admissible to \( x \), \( A^n_p(x,T) \), structural approximation of \( x \), of order \( n \), with respect to \( T \), is defined by: \( A^n_p(x,T) = A^0_{x,n} \) where \( A^\delta_{x,n} \) is recursively defined for \( \delta \in T \) by:

1) If \( |\delta| < n \), then
   a) if \( \hat{C}(x,\delta) = \omega \) then \( A^\delta_{x,n} = \Omega \)
   b) if \( \hat{C}(x,\delta) = (z,k) \), then

   \[ A^\delta_{x,n} = \lambda t^\delta \circ t^\delta \circ 2 \cdots t^\delta \circ (k+\gamma(\delta)) \circ z^\delta \circ A^\delta_{x,n} \cdot A^\delta_{x,n} \cdots A^\delta_{x,n} \]

2) If \( |\delta| = n \), then \( A^\delta = \Omega \).

4.1.13 Corollary. For \( x \in \Lambda \), \( n \in \mathbb{N} \) and \( \Delta \)-tree, \( T \), admissible to \( x \),

1) \( A^n_p(x,T) \) is obtained from \( T^n(x) \) by replacing, by \( \Omega \), each \( L(x,\delta) \) in \( T^n(x) \) such that \( |\delta| = n \).

2) \( A^n_p(x,T) \subseteq x \)

3) If \( m \leq n \), then \( A^m_p(x,T) \subseteq A^n_p(x,T) \).
4.1.14 Example. For $T$ in Example 4.1.2 and $R$ in Example 3.3.2

$$A_p^3(x,T) = \lambda t_1 t_2. t_1(\lambda t_{11}. t_2(\lambda t_{111}. t_{11\Omega})(t_{11\Omega})) (t_{2\Omega}) .$$

The following corollary characterizes the relation between the $C$-function and the structural approximation.

4.1.15 Corollary. Let $x$ be in $\Lambda$ and $T$ be a $\Delta$-tree admissible to $x$. Then, for each $n \in \mathbb{N}$,

$$C(A_p^n(x,T), \delta) = \begin{cases} C(x, \delta) & \text{if } |\delta| < n \text{ or } \delta \notin T \\ \omega & \text{if } |\delta| \geq n \text{ and } \delta \in T \end{cases} .$$

The assertion remains valid when we replace $C$ by $\check{C}$.

Proof. Since $x \simeq T^n(x)$, $L(x, \delta) \simeq L(T^n(x), \delta)$ for all $\delta$ and $C(x) = C(T^n(x))$. The conclusion of the corollary is immediate since, by Corollary 4.1.13, $A_p^n(x,T)$ matches $T^n(x)$ except at occurrences of $\Omega$ in $A_p^n(x,T)$. $\square$
§2. Convergence Lemma

Our objective here is to prove Lemma 4.2.2 which is fundamental throughout the rest of this thesis. We use the proof technique of typed $\lambda$-calculus in Chapter 2.

4.2.1 Lemma. Let $x \in \Delta$ and $T$ be a $\Delta$-tree admissible to $x$. For any $x^t$ in $I(x)$, there is a sufficiently large $n$ such that, $x^t \subseteq A^n(x, T)$.

Since the proof for Lemma 4.2.1 is very long with a high complexity, we first give the outline of the proof in order to ease the unreadability of the full proof.

1: There exists a typed $\beta$-reduction sequence $x^t \rightarrow x^t_p$ so that $x^t_p$ has no typed $\beta$-redex (by Lemma 2.3.13).

2: There exists a (usual) $\beta$-reduction sequence $x \rightarrow x_p$ so that $W(x^t_p)$ is a reduced approximant of $x_p$ (by Lemma 2.3.14).

3: For $k \in \mathbb{N}$, there are $Q^k_1, Q^k_2$ and $W^k \in \Delta$, so that $x^t_p \rightarrow^\beta Q^k_1 \rightarrow^\alpha Q^k_2 \rightarrow^{\text{nb}} W^k$ and $W^k$ matches $T^k(x)$ except at occurrences of $\Omega$ and $LT(x, \delta)$ with $|\delta| = k$ in $T^n(x)$ (by Corollary 4.1.11 and the Church-Rosser Theorem). The parts in $W^k$ to match $\Omega$ in $T^k(x)$ are non-head normal.

4: Correspondingly, there is a typed conversion sequence $x^t_p \rightarrow^t y^t_q \rightarrow^a y^t \rightarrow y^t$ where $y^t_q \in I(Q^k_1)$, $y^t \in I(Q^k_2)$ and $y^t \in I(W^k)$ ($*$ is 'modified' tn-ab) and $x^t_p \subseteq y^t_p$.

5: By the definition of $T^k(x)$, $T^k(x)$ contains each variable $z^\delta$ such that $C(x, \delta) = z^\delta$ for each $\delta \in T$ with $|\delta| = k - 1$. 
6: By 3, \( W^k \) contains each \( z^\delta \) in 5 and, by 4, \( Y^t \) contains a typed variable \( t[z^\delta] \in I(z^\delta) \) for each \( z^\delta \) in 5.

7: We consider the process in which \( z^\delta \) in \( W^k \) is derived:
   
   [1] \( z^\delta \) already occurs in \( x_p \).
   
   [2] \( z^\delta \) is derived in \( x_p \xrightarrow{\beta} Q^k_1 \).
   
   [3] \( z^\delta \) is derived in \( Q^k_2 \xrightarrow{n-ab} W^k \).

8: If we take \( k \) large enough, we can reassign type 0 to each \( t[z^\delta] \) in \( Y^t \) so that \( |\delta| = k-1 \) and still have \( x^t_p \subseteq Y^t \) valid, for,

   a. [1] cannot occur for \( z^\delta \) if \( k \) is large enough.
   
   b. In [2], \( z^\delta \) is from a \( \beta \)-redex in \( x_p \), but \( x^t_p \) has no typed \( \beta \)-redex. (The \( \beta \)-redexes have degenerated to \( \Omega \) in \( x^t_p \).)
   
   c. In [3], note that typed \( n \)-abstraction reduces the type of the variable by 1 (i.e. \( \Delta(n) + \lambda s.(\Delta(n)(n-1)(n-1)) \).

   So, if \( k \) is large enough, \( t[z^\delta] \) is of type 0.

9: Since we have reassigned 0 to each \( t[z^\delta] \) in \( Y^t \) with \( |\delta| = n-1 \), we can replace the subexpressions applied to \( t[z^\delta] \) by \( \Omega(0) \) and still have \( x^t_p \subseteq Y^t \) (by Theorem 2.2.5). Furthermore, we replace, by \( \Omega(0) \), each subexpression in \( Y^t \) that corresponds to a non-head normal subexpression in \( W^k \) (pointed out in 3).

10: Since \( W^k \) matches \( T^k(x) \) except at \( LT(x,\delta) \) (\( |\delta| = n \)) and occurrences of \( \Omega \) in \( T^k(x) \), and \( A^k_p(x,T) \) is derived from \( T^k(x) \) by replacing \( L(x,\delta) \) (\( |\delta| = k \)) by \( \Omega \), we conclude that \( Y^t \) obtained in 9 is in \( I(A^k_p(x,T)) \). So, \( x^t = x^t_p \subseteq Y^t \subseteq A^k_p(x,T) \).
The following is the reason why the proof is so difficult: The transformation $x \to T^n(x)$ involves both $\beta$-reductions and $\eta$-abstractions. The structure, $T$, is arbitrarily given regardless of the structure of $x$. In addition, the transformation $T^n(x) \to A^n_p(x,T)$ is rather an artificial deformation and the parts in $T^n(x)$ that are replaced by $\Omega$ include variables added by $\eta$-abstractions as well as subexpressions generated by $\beta$-reductions.

Proof. By Lemma 2.3.13, there exists the following typed $\beta$-reduction sequence:

$$x^t + x'_1 + x'_2 + \ldots + x'_p$$

where $x_i^t \in \Lambda^t$ and $x_{i+1}^t$ derives from $x_i^t$ by one application of typed $\beta$-reduction and $x_p^t$ has no typed $\beta$-redex. Correspondingly, by Lemma 2.3.14, there is a $\beta$-reduction sequence:

$$x + x'_1 + x'_2 + \ldots + x'_p$$

where $x_{i+1}$ derives from $x_i$ by one application of $\beta$-reduction and $x_i$ matches $W(x_i^t)$ except at occurrences of $\Omega$ in $W(x_i^t)$. Since $x_p^t$ has no $\beta$-redex, $W(x_p^t)$ is a reduced approximant of $x$, and so, every $\beta$-redex in $x_p$ is contained in a part of $x_p$ which has no corresponding part in $W(x_p^t)$ except $\Omega$.

\[
\begin{array}{c}
W(x_p^t) : \hline \\
\Omega & \Omega \\
\end{array}
\]

\[
\begin{array}{c}
x_p : \hline \\
\hline \\
\hline \\
\hline \\
\end{array}
\]

a $\beta$-redex can exist only in these areas.
Given any \( k \) in \( \mathbb{N} \), by Corollary 4.1.11 there exist \( Z_1^k, Z_2^k, Z_3^k \in \Lambda \) such that

\[ x \xrightarrow{\beta} Z_1^k \xrightarrow{\alpha} Z_2^k \xrightarrow{\Omega} Z_3^k \xrightarrow{n-ab} T^k(x) \]

where \( Z_2^k \) matches \( Z_3^k \) except at occurrences of \( \Omega \) in \( Z_3^k \) and every \( \beta \)-redex in \( Z_2^k \) is contained in a part which corresponds to \( \Omega \) in \( Z_3 \) or in \( LT(x, \delta) \) for \( |\delta| = k \). Also, if \( A \Rightarrow^{\lambda s. A.} A_s \) is made in \( Z_3 \Rightarrow T^k(x) \), \( A \) is not a proper subexpression of \( LT(x, \delta) \). By the Church-Rosser Theorem, there exist \( Q_1^k, Q_2^k \in \Lambda \) such that

\[
\xymatrix{
  x 
  \ar[r]^\beta & x_p 
  \ar[r]^\beta & Q_1^k 
  \ar[r]^\alpha & Q_2^k 
  \ar[r]^\beta & Z_2^k 
  \ar[r]^\alpha & Z_1^k 
  \ar[r]^\beta & \Omega 
  \ar[r]^n-ab & T^k(x) 
}
\]

So

\[
\xymatrix{
  x 
  \ar[r]^\beta & x_p 
  \ar[r]^\beta & Q_1^k 
  \ar[r]^\alpha & Q_2^k 
  \ar[r]^\beta & Z_2^k 
  \ar[r]^\alpha & Z_1^k 
  \ar[r]^\beta & \Omega 
  \ar[r]^n-ab & T^k(x) 
}
\]

Since every \( \beta \)-redex in \( Z_2^k \) is either in a part which corresponds to no part except \( \Omega \) in \( Z_3 \) or in \( LT(x, \delta) \) for \( |\delta| = k \), it follows that \( Z_3^k \) matches \( Q_2^k \) except at occurrences of \( \Omega \) and \( LT(x, \delta) \) in \( Z_3^k \).

On the other hand, since all the \( n \)-abstractions in
Z_3 \xrightarrow{n-ab} T^k(x) are made externally to LT(x,\delta) with |\delta| = k, we conclude that there are n-abstractions which, applied to Q_2, yield W_k \in \Lambda which matches T^k(x) except at occurrences of \Omega and LT(x,\delta) in T^k(x) such that |\delta| = n. Thus we have:

\[ x_p \xrightarrow{\beta} Q_1^k \xrightarrow{\alpha} Q_2^k \xrightarrow{n-ab} W_k \]

It follows that the structure of W_k is described as follows:

\[ W_k = \omega_k^0 \text{ where } \omega_k^0 (\delta \in T) \text{ is of the form:} \]

1) if |\delta| = k, then \( \omega_k^0 = w(\delta) \) for some \( w(\delta) \in \Lambda \) such that \( w(\delta) \approx L(x,\delta) \)

2) if |\delta| < k, then

a) if \( \hat{C}(x,\delta) = \omega \), then \( \omega_k^0 \) is a certain expression which has no head normal form

b) if \( \hat{C}(x,\delta) = (z^r, \sigma^r) \), then

\[ \omega_k^0 = \lambda t_{\delta_01} t_{\delta_02} \cdots t_{\delta_0(\gamma T(\delta) + r^\sigma)} z^\delta \omega_k^0 z_{\delta_01}^\delta \omega_k^0 z_{\delta_02}^\delta \cdots \omega_k^0 Y_T(\delta) \]

We examine each \( z^\delta \) which appears in \( W_k \). \( z^\delta \) in \( W_k \) must satisfy one of the following conditions:

1. This occurrence of \( z^\delta \) is homologous to one in \( x_p \) (i.e. it occurs already in \( x_p \)).

2. It was derived in the process of \( x_p \xrightarrow{\beta} Q_1^k \).

3. It was derived in the process of \( Q_3^k \xrightarrow{n-ab} W_k \).

(In any case (1), (2) and (3) above, \( z^\delta \) may have been renamed in \( Q_2^k \xrightarrow{\alpha} Q_3^k \).)

Let \( j = 1, 2 \) or 3. We define \( \eta_j^k(i) \) to be the subset of \( T \) determined by:
\[ n^k(i) = \{ \delta \mid \delta \in T, |\delta| = i, C(x, \delta) \not\in \omega \text{ and } z^\delta \text{ in } W^k \text{ is derived as in } [j] \} . \]

Since \( x_p \) is a finite expression, there is \( m_1 \) in \( \mathbb{N} \) for which (1) above cannot occur for \( z^\delta \) if \( |\delta| \geq m_1 \). Let \( m_2 \) be the maximum type among the types that were assigned to the components of \( x^t \). We set \( n = m_1 + m_2 \). (To simplify the description of the proof, we set \( n = m_1 + 1 \) if \( m_2 = 0 \).) We set \( k = n \) and develop \( x \xrightarrow{\beta} x_p \xrightarrow{\beta} Q^1 \) into

\[
 x = x_0 + x_1 + x_2 + \cdots + x_p + x_{p+1} + \cdots + x_q = Q^n
\]

where \( x_{i+1} \) is the result of an application of \( \beta \)-reduction to \( x_i \). Correspondingly, we define a sequence typed \( \lambda \)-expressions:

\[
y^t_0 + y^t_1 + y^t_2 + \cdots + y^t_p + y^t_{p+1} + \cdots + y^t_q
\]

in the following way:

**Step 1:** Set \( y^t_0 = x^t \in \mathcal{I}(x^t_0) \).

**Step i:** Suppose \( y^t_{i-1} \) is in \( \mathcal{I}(x^t_{i-1}) \) and that the redex which is reduced in \( x^t_{i-1} + x^t_i \) is \( (\lambda s.M)N \xrightarrow{\beta} fM \) and that the corresponding occurrence of \( (\lambda s.M)N \) in \( y^t_{i-1} \) is of the form \( ((\lambda s.M^t)^{(h_1)})(N^t)^{(h_2)} \) where \( M^t \in \mathcal{I}(M) \) and \( N^t \in \mathcal{I}(N) \). We define \( y^t_i \) as follows:

**Case 1.** If \( h_1 > 0 \), replace \( ((\lambda s.M^t)^{(h_1)})(N^t)^{(h_2)} \) in \( y^t_{i-1} \) by

\[
\left[ \int^S M^t \right]_{\min(h_1-1, h_2)}.
\]

**Case 2.** If \( h_1 = 0 \), replace \( ((\lambda s.M^t)^{(h_1)})(N^t)^{(h_2)} \) by
Obviously $y_i^t$ defined above is in $I(x_i)$. Moreover,

$$y_{i-1}^t \subseteq y_i^t \quad (\text{for } i \geq 1)$$

for, if Case 1 occurs, then it is exactly typed $\beta$-reduction. So by Theorem 2.3.12,

$$y_{i-1}^t = y_i^t$$

On the other hand, in Case 2, we could have replaced $((\lambda s.M^t)(\Omega(0))_{N^t})^{(h_2)}$ in $y_i^t$ by

$$\left[\int_{\Omega(0)}^S M^t\right]_0$$

without changing $D_\infty$-value of $y_{i-1}^t$. Since

$$\left[\int_{\Omega(0)}^S M^t\right]_0 \subseteq \left[\int_{\Omega(0)}^S M^t\right]_0$$

$$y_{i-1}^t \subseteq y_i^t$$

By induction, we conclude that, for any $i$ such that $1 \leq i \leq q$,

$$y_i^t \in I(x_i) \quad \text{and} \quad y_{i-1}^t \subseteq y_i^t.$$
Thus we have proved that there exists a typed $\lambda$-expression $y^t_q$ in $T(Q^n_1)$ such that $x^t \subseteq y^t_q$. Also it is easy to see that, for $i$ such that $1 \leq i \leq p$, $y^t_i$ matches $x^t_i$ except at occurrences of $\Omega$ in $x^t_i$.

Now we apply typed $\alpha$-conversions to $y^t_q$ which correspond to $Q^n_1 \overset{\alpha}{\longrightarrow} Q^n_2$. Let this result be $y^t \in T(Q^n_2)$. (So, $y^t_q \overset{\tau \alpha}{\longrightarrow} y^t$.)

Next, let us develop $Q^n_2 \overset{n\text{-ab}}{\longrightarrow} W^n$ into the sequence:

$$Q^n_1 = Y_1 + Y_2 + \cdots + Y_{d-1} + Y_d = W^n$$

where $Y_i$ is derived from $Y_{i-1}$ by an application of $n$-abstraction.

Correspondingly, we define a sequence of typed $\lambda$-expression $y^t_1 + y^t_2 + \cdots + y^t_d$ as follows:

**Step 1:** $y^t_1 = y^t$

**Step i:** Suppose that $Y_{i-1} \overset{n\text{-ab}}{\longrightarrow} Y_i$ is the replacement of $A$ in $Y_{i-1}$ by $\lambda s. A s$ and that $Y^t_{i-1}$ is in $T(Y_{i-1})$. Let $A^t \in T(A)$ be the corresponding occurrence of $A$ in $Y^t_{i-1}$. We replace it with $(\lambda s. (A^t s ([j-1])))((j-1)) (j)$ to have $Y^t_i$. Now it is easy to see that $Y^t_i$ is in $T(Y_i)$ and $Y^t_{i-1} = Y^t_{i-1}$. We set $Y^t = Y^t_d$.

We have proved here that there exists a typed expression $Y^t$ in $T(W^n)$ so that $x^t \subseteq Y^t$. Since $Y^t$ is in $T(W^n)$, we name each component of $Y^t$ via the corresponding component of $W^n$, that is,
and \( t[z^\delta] \) in \( Y^t \) corresponds to \( z^\delta \) in \( W^n \)

\[ t[W^n_0] \] in \( Y^t \) corresponds to \( W^n_0 \) in \( W^n \).

Our next stage is to transform \( Y^t \) into another typed expression in \( T(A^n_p(x,T)) \) without violating \( x^t \subseteq Y^t \). Consider all \( t[z^\delta] \) in \( Y^t \) such that \( |\delta| = m_1 \). As we have observed as for \( W^n \),

either \( \delta \in \eta^n_1(m_1), \eta^n_2(m_1) \) or \( \eta^n_3(m_1) \),

but it is impossible that \( \delta \in \eta^n_1(m_1) \) because of the definition of \( m_1 \). Let \( \delta \in \eta^n_2(m_1) \). Then \( z^\delta \) has been derived in \( x_p \xrightarrow{\beta} Q^n_1 \).

Since \( x^t_p \) matches \( y^t_p \) except at occurrences of \( \Omega \) in \( x^t_p \) and \( x^t_p \) has no \( \beta \)-redex, every \( \beta \)-redex of \( y^t_p \) is in a part which has no corresponding part in \( x^t_p \) except \( \Omega \). Thus, by replacing each \( \beta \)-redex in \( y^t_p \) by \( \Omega(0) \), \( x^t_p \subseteq y^t_p \) still holds. Since \( t[z^\delta] \) in \( Y^t \) was derived from some \( \beta \)-redexes in \( y^t_p \), we can reassign to \( t[z^\delta] \) the minimum type 0 and still have \( x^t = x^t_p \subseteq Y^t \).

For each \( \delta \) in \( \eta^n_2(m_1) \), \( t[W^n_\delta] \) is as

\[
(\cdots((t[z^\delta]t[W^n_{\delta o_1}])(*)t[W^n_{\delta o_2}])(*)...)(*)t[W^n_{\delta o_Y T(\delta)}])(*)
\]

in \( Y^t \) where \((*)\)'s are types.

Since \( t[z^\delta] \) is now of type 0, we can replace each \( t[W^n_{\delta o_i}] \) (\( i = 1, 2, \ldots, Y_T(\delta) \)) by \( \Omega(0) \) without affecting the \( D_\infty \)-value of \( Y^t \).

(If \( a \in \eta_0(D_\infty) \), \( (\cdots((ab_1)b_2)\cdots)b_n = (\cdots((a)\cdots)b_n \) by Theorem 2.2.5.) So, at least, we can reassign the minimum type 0 to each component of \( t[W^n_{\delta o_1}] \) in \( Y^t \) and still have \( x^t \subseteq Y^t \).
Especially, for $\delta' \in T$ with $|\delta'| = n-1$ for which there exists $\delta \in \eta_2^n(m_1)$ such that $\delta < \delta'$, $t[z^{\delta'}]$ is of type 0.

Next, we consider $\delta$ such that $\delta \in \eta_3^n(m_1)$. Then $z^{\delta}$ was derived in $Q_2^n \xrightarrow{n\text{-ab}} W^n$. Since the highest type among those that are attached to the components of $x^t$ is $m_2$ and the conversion $x^t \rightarrow y^t \in T(Q_2^n)$ does not increase any type, the highest type in $y^t$ is not more than $m_2$. This means that $t[z^{\delta}]$ is of type at most $m_2 - 1$ by the way the sequence $Y_1^t \rightarrow Y_2^t \rightarrow \ldots \rightarrow Y_d^t$ is defined.

On the other hand, if $t[W^{n}]$ is as

$$(\cdots((t[z^{\delta}]t[W^{n}_{\delta \circ i}])(*)t[W^{n}_{\delta \circ 2}])(*)\ldots)(*)t[W^{n}_{\delta \circ \gamma_T(\delta)}])(*)$$

in $Y^t$, since $z^{\delta}$ comes from $n$-abstraction, so does each $W^{n}_{\delta \circ i}$ ($i = 1, 2, \ldots, \delta \circ \gamma_T(\delta)$) and, so, $W^{n}_{\delta \circ i}$ is, in fact, the variable $z^{\delta \circ i}$. Since $t[z^{\delta}]$ is of type at most $[m_2-1]$, $t[z^{\delta \circ 1}]$ is of type at most $[m_2-2]$,

$$
\vdots
$$

$t[z^{\delta \circ i}]$ is of type at most $[m_2-(i+1)]$, again by the way the sequence $Y_1^t \rightarrow Y_2^t \rightarrow \ldots \rightarrow Y_d^t$ is defined. This indicates that if we take $\delta' \in T$ with $|\delta'| = m_1 + s$ for which there is $\delta \in \eta_3^n(m_1)$ such that $\delta < \delta'$, then $t[z^{\delta'}]$ is of type at most $[m_2-(s+1)]$. Especially, if $s = m_2 - 1$, (i.e. $|\delta'| = n-1$) then $t[z^{\delta'}]$ is of type at most 0.

What we have proved is that we have transformed $Y^t$ so that, for any $\delta \in T$ such that $|\delta| = n-1$, the type of $t[z^{\delta}]$ in $Y^t$ is 0. So we can replace each $t[W^{n}_{\delta}]$ with $|\delta| = n$ by $\Omega^{(0)}$ without affecting the $D_\infty$-value of $Y^t$. 

Finally, we replace $t[w^n]$ in $y^t$ by $\Omega^{(0)}$ if $C(x,\delta) = \omega$.

Since $W^n_\delta$ in $W^n$ has no head normal form, this transformation does not affect the $D_\infty$-value of $Y^t$, either.

We remember that $W^n$ matches $T^n(x)$ except at occurrences of $\Omega$ and $LT(x,\delta)$ ($|\delta| = n$) in $T^n(x)$. So $W^n$ matches $A^n_p(x,T)$ except at occurrences of $\Omega$ in $A^n_p(x,T)$ since each $L(x,\delta)$ ($|\delta| = n$) in $T^n(x)$ is replaced by $\Omega$ in $A^n_p(x,T)$.

It follows that $Y^t \in I(A^n_p(x,T))$.

We conclude that

$$x^t \subseteq y^t \subseteq A^n_p(x,T) \quad .$$

4.2.2 Lemma (Convergence Lemma). Let $x$ be in $A$ and $T$ be a $A$-tree admissible to $x$. Then $x = \cup_{n=0}^{\infty} A^n_p(x,T)$.

Proof. Since $A^n_p(x,T) \subseteq T^n(x) = x$, for any $n$,

$$\cup_{n=0}^{\infty} A^n_p(x,T) \subseteq x \quad .$$

On the other hand, by Lemma 4.2.1, for all $x^t \in I(x)$, there exists $n$ such that $x^t \subseteq A^n_p(x,T)$. By Lemma 2.3.9, $x = \cup_{n=0}^{\infty} I(x)$.

Thus $x \subseteq \cup_{n=0}^{\infty} A^n_p(x,T)$.

4.2.3 Corollary (Park). Let $Y$ be the fixed point operator.

Then $Yf = \cup_{n=0}^{\infty} f^n(\emptyset)$ for $f \in A$. 
Proof. Let $T = \{ \delta \mid \delta = (11\cdots1) \text{ for some } n \in \mathbb{N} \cup \{0\} \}$.

Then

$$A^n_p(Yf,T) = f(f(\cdots(f(f(\delta))))).$$

The result is immediate from Lemma 4.2.2. $\square$
§3. Characterization of $D_\infty$-Value of $\lambda$-Expressions

In this section, we prove the converse of Corollary 3.4.11 and 3.4.12 using the Convergence Lemma 4.2.2.

4.3.1 Theorem. Let $x, y$ be in $\Lambda$. If $C(x) \leq C(y)$, then $x \subseteq y$.

Proof. Suppose $C(x) \leq C(y)$. By Corollary 3.4.9, $\hat{C}(x) \leq \hat{C}(y)$. We take a sufficiently large $\Delta$-tree, $T$, which is admissible to both $x$ and $y$. We compare $A_{p}^{n}(x, T)$ and $A_{p}^{n}(y, T)$. Since $\hat{C}(x) \leq \hat{C}(y)$, $A_{p}^{n}(x, T)$ matches $A_{p}^{n}(y, T)$ except at occurrences of $\Omega$ in $A_{p}^{n}(x, T)$ by Definition 4.1.12. So

$$A_{p}^{n}(x, T) \subseteq A_{p}^{n}(y, T).$$

By Lemma 4.2.2, $x \subseteq y$. □

4.3.2 Theorem. For $x, y$ in $\Lambda$, $x \subseteq y$ if and only if $C(x) \leq C(y)$, and so, $x = y$ if and only if $C(x) = C(y)$.

Proof. By Corollary 3.4.11 and Theorem 4.3.1. □

4.3.3 Example. Let $Y_{0} = Y$, the fixed point operator, and define inductively $Y_{i} = Y_{i-1}G$ where $G = \lambda x \lambda f.f(xf)$. We can show that $Y_{i} = Y_{j}$ for any pair $(i, j)$. For example, we prove $C(Y_{0}) = C(Y_{1})$.

$$Y_{0} = Y \xrightarrow{B} \lambda f.f((\lambda h.f(hh))(\lambda h.f(hh)))$$
$$\xrightarrow{B} \lambda f.f((\lambda h.f(hh))(\lambda h.f(hh))).$$

*Refer to [24] for an alternative characterization of $C$. □
Let $G^*$ denote $(\lambda h. G(hh))(\lambda h. G(hh))$. Note that
$$G^* \xrightarrow{B} G((\lambda h. G(hh))(\lambda h. G(hh))) .$$

$$Y_1 = YG \xrightarrow{B} G^* \xrightarrow{B} GG^* \xrightarrow{B} \lambda f.f(G^*f) \xrightarrow{B} \lambda f.f(GG^*f)$$
$$\xrightarrow{B} \lambda f.f(f(G^*f)) \xrightarrow{B} \lambda f.f^n(G^*f) .$$

Now it is obvious that $C(Y_0) = C(Y_1)$. We can see the proof for $Y_1 \not\vdash Y_j$ (i $\neq$ j) in [3].

4.3.4 Example (Wadsworth). Let $F = \lambda f\lambda x\lambda y.x(fy)$ and $J = YF$.

So
$$J \xrightarrow{B} \lambda x\lambda y.x(Jy) .$$

Let $I = \lambda x.x$. Then it is easy to show that $C(I) = C(J)$. Thus $I = J$. Obviously $I \not\vdash J$. It was a surprising fact that a normal $D_\infty$ expression $I$ is equal to a non-normal expression $J$. $J$ might be considered to be an infinite computation process.

Given an input, it returns the computation result little by little taking an infinite amount of time. The limit of this infinite computation turns out to be equal to the computation of $I$. The conversion rules alone cannot describe the outcome of this infinite computation. It is possible only after $\Lambda$ is mapped into a lattice space such as $D_\infty$ where the limit of such infinite computation can exist. As Scott claims in [16], $=_{D_\infty}$ is a more essential relation than the convertibilities. Further discussion on computational interpretations of normality, non-normality, head-normality of $\lambda$-expressions will be given in Chapter 5.
§4. Further Properties of $\Lambda$ Mapped in $D_\infty$

In this section, we state further properties of $\Lambda$ mapped in $D_\infty$ lattice. These properties will be the basis of the theory on lattice $\Lambda^\infty$ which will be introduced in Chapter 6.

4.4.1 Theorem. Let $x$ be in $\Lambda$. Suppose $C(x,\delta) \neq \omega$ for any $\delta \in \Lambda$, then $x$ is maximal in $\Lambda$, that is, there is no $y$ in $\Lambda$ such that $x \sqsubseteq y$.

Proof. If $x \sqsubseteq y$ for some $y$ in $\Lambda$, it must be that $C(x) < C(y)$ by Theorem 4.3.2. Since there is no $\delta \in \Delta$ such that $C(x,\delta) = \omega$, $x = y$. □

4.4.2 Corollary. Let $x$ be in $\Lambda$. If $x$ has a normal form, then $x$ is maximal in $\Lambda$.

Proof. If $x$ is normal, $C(x,\delta) \neq \omega$ for any $\delta \in \Delta$. □

4.4.3 Definition.

1. Let $\mathcal{D}$ be a subset of $\Lambda$. $\mathcal{D}$ is said to be directed (with respect to $D_\infty$ partial order) if $\mathcal{D}$ satisfies the following property: For $F$ any finite subset of $\mathcal{D}$, there exists an element $z$ of $\mathcal{D}$ such that, for each $x \in F$,

$$W[[x]] \rho \subseteq W[[z]] \rho$$

for all environments $\rho$.

2. Let $\mathcal{D} \subseteq \Lambda$ be directed. $\mathcal{D}$ is said to be interesting if there is no $x$ in $\mathcal{D}$ for which
for all environments \( \rho \).

The following theorem is a generalization of Lemma 4.2.2.

4.4.4 **Theorem** (General Convergence Lemma). Let \( D \) be a directed subset of \( \Lambda \). We define \( c_D \in \mathcal{C} \) by:

\[
c_D(\delta) = \begin{cases} 
\omega & \text{if } C(y,\delta) = \omega \text{ for all } y \in D \\
\pi \in V & \text{if } C(y,\delta) = \pi \text{ for some } y \in D.
\end{cases}
\]

Then \( c_D = C(x) \) for \( x \in \Lambda \) if and only if \( x = \bigcup D \).

**Proof.** To prove that \( c_D \) is well defined, assume that, for some \( y_1, y_2 \in D \) and \( \delta \in \Lambda \),

\( C(y_1,\delta) \neq \omega, \ C(y_2,\delta) \neq \omega \) and \( C(y_1,\delta) \neq C(y_2,\delta) \). Since \( D \) is directed, there must be \( z \) in \( D \) for which both \( y_1 \subseteq z \) and \( y_2 \subseteq z \), but this is impossible by Theorem 4.3.2. So given any \( \delta \in \Lambda \), either \( C(y,\delta) = \omega \) for all \( y \in D \) or there is \( \pi \in V \) such that \( C(y,\delta) = \pi \) for all \( y \in D \) such that \( C(y,\delta) \neq \omega \).

Let \( T \) be a \( \Delta \)-tree which is admissible to \( x \). Now suppose \( c_D = c(x) \). For any \( \delta \in T \), there is at least one \( y \in D \) such that \( C(x,\delta) = C(y,\delta) \). Given \( n \in \mathbb{N} \), since \( \#\{\delta \mid \delta \in T \} \) is finite, there is a finite subset, \( F \), of \( D \) such that, for any \( \delta \in T \) with \( |\delta| < n \), there is at least one \( y \in F \) for which \( C(x,\delta) = C(y,\delta) \). By directedness of \( D \), there is \( z \) in \( D \) such that, for any \( y \in F \), \( y \subseteq z \). It follows that \( C(A^p_n(x,T)) \subseteq C(z) \). So, by Theorem 4.3.1, \( A^p_n(x,T) \subseteq z \). By
Lemma 4.2.2, \( x = \bigcup_{n=0}^{\infty} A_n(x, T) \). Thus \( x \subseteq \bigcup D \). On the other hand, since \( C(y) \leq C(x) \) for any \( y \in D \), \( \bigcup D \subseteq x \).

Conversely, suppose that \( x = \bigcup c_D \). By Theorem 4.3.1, \( C(y) \leq C(x) \) for all \( y \in c_D \). Assume, for some \( \delta \in \Delta \),

\[
C(x, \delta) \neq \omega \quad \text{and} \quad C(y, \delta) = \omega \quad \text{for all } y \in D.
\]

Using the fact that \( c_D \) is directed, a discussion similar to the proof of Lemmas 3.4.5 and 3.4.6 leads us to prove that there exists an environment \( \rho \) and \( e_1, e_2, \ldots, e_n \in \Lambda_c \) such that

\[
\downarrow = \bigwedge \left[ \left[ y \cdot e_1 \cdot e_2 \cdot \ldots \cdot e_n \right]_\rho \right] \subseteq \bigwedge \left[ x \cdot e_1 \cdot e_2 \cdot \ldots \cdot e_n \right]_\rho
\]

for all \( y \in c_D \). So, under \( \rho \),

\[
(\bigcup c_D) e_1 e_2 \ldots e_n = \downarrow \subseteq x e_1 e_2 \ldots e_n.
\]

This means that \( \bigcup c_D \not\subseteq x \) contradicting the assumption. \( \square \)

However, it is not always the case that a directed subset of \( \Lambda \) has a least upper bound in \( \Lambda \). But as we see in the next theorem, every element of \( \Lambda \) that is not the bottom is the least upper bound of a directed subset of \( \Lambda \) which does not include the original element. In Chapter 6, this situation will be discussed more uniformly.
4.4.5 **Definition.** Let $D$ be a directed-complete lattice and $F$ be a directed subset of $D$. Then $F$ is said to be interesting if $F$ does not contain its own least upper bound, i.e. $\cup F \notin F$.

Note that any finite directed subset is not interesting and that any infinite non-interesting directed subset can become interesting by removing its least upper bound.

4.4.6 **Theorem.** Let $x$ be in $\Lambda$. If $x$ has a head normal form, then there is a subset $\mathcal{D}$ of $\Lambda$ which is an interesting directed set such that

$$x = \bigcup_{D \in \mathcal{D}} D.$$ 

**Proof.** We take a sufficiently large $\Delta$-tree $T$ so that

1) $T$ is admissible to $x$.

2) For any $n > 0$, there exists at least one $\delta \in T$ with $|\delta| = n$, for which $C(x, \delta) \neq \omega$.

For example, $T$ as defined below satisfies 1) and 2) above: Take any $\Delta$-tree $T'$ admissible to $x$. Let $T$ be a $\Delta$-tree which includes $T' \cup \{\delta_{(\eta_T(\delta) + 1)} | \delta \in T\}$. In the first place, it is obvious that such a $\Delta$-tree exists. In the second place $T$ is admissible to $x$. Thirdly, since $T'$ is admissible, each $L(x, \delta)$ with $\delta \in T - T'$ is obtained by $\eta$-abstraction, and so, in fact, $L(x, \delta)$ is $t_{\delta \cdot k}$ for some $k > 0$. 

Since \( \{ \delta \mid |\delta|=n, \delta \in T-T' \} \neq \emptyset \) for each \( n \) by the definition of \( T \), it follows that \( T \) satisfies 2), too. Since there is at least one \( \delta \in T \) with \( |\delta|=n \) for each \( n \) such that \( C(x,\delta) \neq \omega \) and \( C(A_{\rho}^n(x,\delta),T) = \omega \), we conclude that \( A_{\rho}^n(x,T) \subseteq x \) for all \( n \) by Theorem 4.3.2.

On the other hand, \( x = \bigcup_{n=0}^{\infty} A_{\rho}^n(x,T) \) by Lemma 4.2.2. Thus, \( D = \{ A_{\rho}^0(x,T), A_{\rho}^1(x,T), A_{\rho}^2(x,T), \ldots \} \)

is an interesting directed set whose least upper limit is \( x \). \( \square \)

**4.4.7 Theorem.** Given \( x, y \in A \) such that \( x \subseteq y \). Then there is \( z \in A \) with \( x \subseteq z \subseteq y \).

**Proof.** Let \( T \) be a \( \Delta \)-tree which is admissible to both \( x \) and \( y \). Let \( n \in \mathbb{N} \) be such that there exists \( \delta \) with \( |\delta|=n \) for which \( C(x,\delta) = \omega \) and \( C(y,\delta) \neq \omega \). By Definition 4.1.3, \( T^n(y) \) contains \( L(y,\delta) \) as a subexpression. Since \( C(y,\delta) \neq \omega \),

\[
L(y,\delta) \rightarrow_{B} \lambda s_1 s_2 \cdots s_p \cdot zY_1 Y_2 \cdots Y_q
\]

Let \( z \) be derived from \( T^n(y) \) by replacing \( L(y,\delta) \) in \( T^n(y) \) by \( \lambda s_1 s_2 \cdots s_p s_{p+1} \cdot zY_1 Y_2 \cdots Y_q \) where \( s_{p+1} \) does not appear free in \( zY_1 Y_2 \cdots Y_q \). Now it is easy to see

\[
x \subseteq z \subseteq T^n(y) = y \quad \square
\]
The following fact is interesting in relation to \( \omega \)-completeness discussions in Barendregt [2] and Plotkin [11].

4.4.8 Theorem. Let \( x, y \in A \). If, for any \( z \) in \( A_c \),

\[ xz = yz, \quad \text{then} \quad x = y. \]

Proof. Suppose \( x \neq y \). By Theorem 4.3.2, \( C(x) \neq C(y) \).

By Theorem 3.4.7, there exist \( e_1, e_2, \ldots, e_n \in A \) and an environment \( \rho \) such that

\[
W [[xe_1 e_2 \cdots e_n]] \rho \neq W [[ye_1 e_2 \cdots e_n]] \rho .
\]

Since, by Proposition 2.3.5 and Corollary 2.3.18, the \( \not\in \) can be realized with both sides being in \( \Lambda_c \), so we can choose \( \rho \) such that

\( \rho(V) \subseteq \Lambda_c \).

Let

\[
x = \int_{\rho(u_1), \rho(u_2), \ldots, \rho(u_p)}^{u_1, u_2, \ldots, u_p} x,
\]

\[
y = \int_{\rho(v_1), \rho(v_2), \ldots, \rho(v_q)}^{v_1, v_2, \ldots, v_q} y,
\]

and

\[
\bar{e}_i = \int_{\rho(w_1^i), \rho(w_2^i), \ldots, \rho(w_m^i)}^{w_1^i, w_2^i, \ldots, w_m^i} e_i,
\]

where \( u_1, u_2, \ldots, u_p \) are the free variables occurring in \( x \),
\( v_1, v_2, \ldots, v_q \) are the free variables in \( y \) and \( w_1^i, w_2^i, \ldots, w_{m(i)}^i \) are the free variables in \( e_i \) for \( i = 1, 2, \ldots, n \).

Now the inequality (*) is written as:

\[
\bar{x} \bar{e}_1 \bar{e}_2 \cdots \bar{e}_n \not\equiv \bar{y} \bar{e}_1 \bar{e}_2 \cdots \bar{e}_n
\]

where \( \bar{x}, \bar{y}, \bar{e}_i \in \Lambda_C \). By extensionality of \( D_\infty \), we conclude that

\[
\bar{x} \bar{e}_1 \cdots \bar{e}_{n-1} \not\equiv \bar{y} \bar{e}_1 \cdots \bar{e}_{n-1}
\]

and so

\[
\bar{x} \bar{e}_1 \cdots \bar{e}_{n-2} \not\equiv \bar{y} \bar{e}_1 \cdots \bar{e}_{n-2} \\
\vdots \\
\bar{x} \bar{e}_1 \not\equiv \bar{y} \bar{e}_1
\]

This indicates that if \( x \not\equiv y \), there exists \( \bar{e}_1 \in \Lambda_C \) for which

\[
\bar{x} \bar{e}_1 \not\equiv \bar{y} \bar{e}_1 . \quad \Box
\]

We translate this theorem into one which is stated by C-functions:

4.4.9 Corollary. Let \( x, y \) be in \( \Lambda \). If, for any \( z \in \Lambda_C \), \( C(xz) = C(yz) \), then \( C(x) = C(y) \). \( \Box \)

Theorem 4.3.5 is obvious if we replace \( z \in \Lambda_C \) by \( z \in D_\infty \) since \( D_\infty \) is extensional. The theorem says that the extensionality holds in \( \Lambda_C \) modulo \( D_\infty \).
CHAPTER 5
INFINITE EXPANSIONS IN REAL PROGRAMMING LANGUAGES

We discuss informally how the concepts and formulations introduced for the \( \lambda \)-calculus in the previous chapters are applied to more realistic programming languages such as recursively defined programs and Algol-like programs. We present algorithms to translate a program written in these languages into a \( \lambda \)-expression. Using this translation, we show that the C-function for the \( \lambda \)-expressions, in fact, corresponds to the infinite expansion or the executions on all possible inputs for the programs in realistic languages. Most of the results here are not essentially new.
§1. Recursively Defined Programs

We consider the programs which are defined by recursive equations. (Discussions on this type of program are found, for example, in [20]). The following is the syntax of the language R.

Syntax of R

Elements

1. $A_1, A_2, \ldots$: Symbols for constants
2. $X_1, X_2, \ldots, X_k$: Symbols for variables
3. $G_1, G_2, \ldots, G_m$: Symbols for known functions
4. $F_1, F_2, \ldots, F_n$: Symbols for unknown functions

$<\text{term}> ::=$ $A_1 | A_2 | \ldots$

$|X_1 | X_2 | \ldots |X_k |$

$|G_1 (<\text{term}>_1, \ldots, <\text{term}>_{k_1}>)$

$\vdots$

$|G_m (<\text{term}>_1, \ldots, <\text{term}>_{k_m}>)$

$|F_1 (<\text{term}>_1, \ldots, <\text{term}>_{p_1}>)$

$\vdots$

$|F_n (<\text{term}>_1, \ldots, <\text{term}>_{p_n}>)$

$<\text{program}> ::= \begin{cases} 
F_1 (X_1, X_2, \ldots, X_p) + <\text{term}>_1 \\
\vdots \\
F_n (X_1, X_2, \ldots, X_p) + <\text{term}>_n 
\end{cases}$

where we assume that $<\text{term}>_k$ does not contain any variable symbol other than $X_1, X_2, \ldots, X_{p_k}$ for $k = 1, 2, \ldots, n$. 
For example

\[ n = 1: \ F(X) + F(F(X)) \]

\[ n = 2: \ \begin{cases} 
F_1(X_1, X_2) + G_1(X_2, F_2(X_1)) \\
F_2(X_1) + G_2(F_2(X_1), F_1(X_1)) 
\end{cases} \]

are programs of \( R \).

**Computation of Terms**

Given a program \( \xi \) where \( \xi \) is:

\[
\begin{cases} 
F_1(X_1, X_2, \ldots, X_{k_1}) + \psi_1(X_1, X_2, \ldots, X_{k_1}) \\
\vdots \\
F_n(X_1, X_2, \ldots, X_{k_n}) + \psi_n(X_1, X_2, \ldots, X_{k_n})
\end{cases}
\]

where \( \psi_i(X_1, \ldots, X_{k_i}) \) is a term with occurrences of \( X_1, X_2, \ldots, X_{k_i} \) for \( i = 1, 2, \ldots, n \). For a term \( T \), a computation of \( T \) according to \( \xi \) is defined to be a sequence of terms:

\[ T_1, T_2, \ldots, T_n \]

where \( T_1 = T \) and \( T_i \) is obtained from \( T_{i-1} \) by replacing an occurrence of \( F_j(s_1, s_2, \ldots, s_{k_j}) \) by \( \psi_j(s_1, s_2, \ldots, s_{k_j}) \) where \( s_i \)'s are terms. We write \( T \xrightarrow{\xi} T_n \).

**Translation of \( R \) to \( \Lambda \)**

Given a term \( T \) and a program \( \xi \), we want to synthesize a \( \lambda \)-expression \( \Sigma_\xi(T) \) such that each computation of \( T \) according to \( \xi \) corresponds to a \( \beta \)-reduction sequence from \( \Sigma_\xi(T) \).
Let $a_1, a_2, \ldots, x_1, x_2, \ldots, x_l, g_1, g_2, \ldots, g_m, \phi_1, \phi_2, \ldots, \phi_n$ be distinct variables in $A$.

**Algorithm**

(a) If $T$ is $A_i$,

$$\Sigma^i_\xi(T) = a_i.$$  

(b) If $T$ is $X_i$, then $\Sigma^i_\xi(T) = x_i$.

(c) If $T$ is $G_i(s_1, s_2, \ldots, s_{p_i})$ where $\Sigma^i_\xi(s_j) = S_j$ for $j = 1, 2, \ldots, p_i$, then $\Sigma^i_\xi(T) = g_i(s_1)(s_2) \cdots(s_{p_i})$. The parentheses are omitted if $S_j$ is a variable.

(d) If $T$ is $F_i(s_1, s_2, \ldots, s_{k_i})$ then

$$\Sigma^i_\xi(T) = \phi_i(s_1)(s_2) \cdots(s_{k_i}).$$

By applying the transformations (a), (b), (c), we obtain a $\lambda$-expression $\Sigma^i_\xi(T)$ in $\{a_i, g_j, \phi_k\}^*$ for a term $T$. Next we substitute a $\lambda$-expression for $\phi_1, \phi_2, \ldots, \phi_n$ in $\Sigma^i_\xi(T)$. For example we see what to substitute for $\phi_1$.

Let

$$y_n = Y(\lambda \phi_n, \Sigma^i_\xi(\psi_n))$$

$$y_{n-1} = Y(\lambda \phi_{n-1}, \int \phi_n \Sigma^i_\xi(\psi_{n-1}))$$

$$\vdots$$

$$y_1 = Y(\lambda \phi_2, \phi_3, \ldots, \phi_n, \Sigma^i_\xi(\psi_1))$$

$$\vdots$$

$$y_1 = Y(\lambda \phi_2, \phi_3, \ldots, \phi_n, \Sigma^i_\xi(\psi_1))$$
Then \( Y_1 = y_1 \) is the \( \lambda \)-expression we want to substitute for \( \phi_1 \).

In the similar manner, we synthesize a \( \lambda \)-expression \( Y_i \) to substitute for \( \phi_i \) (\( i = 1, 2, \ldots, n \)). Now

\[
\Sigma_\xi(T) = \int_{Y_1, Y_2, \ldots, Y_n} \Sigma_\xi'(T).
\]

It is easy to see that:

5.1.1 Theorem. For terms \( T_1, T_2 \) and a program \( \xi \), \( T_1 \xrightarrow{\xi} T_2 \) if and only if \( \Sigma_\xi(T_1) \xrightarrow{\beta} \Sigma_\xi(T_2) \).

To translate the result of Chapter 4 to \( R \), we introduce the notion of semantics to \( R \).

Theorem 4.3.2 can be read as:

"Given \( \lambda \)-expressions (programs) \( x, y \), if \( x \) and \( y \) have the same \( C \)-function (infinite expansion), then \( x \) is equivalent to \( y \) under the interpretation of the \( D_\infty \) semantics."

Here, instead of \( D_\infty \), we use general domains to specify the semantics of \( R \) (as in [20]).

Semantics of \( R \)

We define an interpretation, \( I \), of \( R \) to be the pair \((D_I, v_I)\) where \( D_I \) is a directed complete partially ordered set with \( \bot = \cap D_I \) and \( v_I \) is the semantic function which maps:

- the constant symbols \( A_i \) to elements \( a_i \),
- the variable symbols \( X_i \) to variables \( x_i \) which range over \( D_I \),
- the known function symbols
$G_i \rightarrow g_i \in [D_I \times D_I]$. $v_I$ is extended in the obvious manner to the terms generated from $G_i$, $A_j$ and $X_k$. Now a program $\xi$ of $R$ can be translated via $v_I$ into an equation $\xi^*$ with the unknown functions $F_i$'s over $D_I$. By Scott's fixed point theorem (Theorem 2.2.3) we can conclude that there exist continuous functions

$$f_1, f_2, \ldots, f_n \in [D_I \times D_I]$$

which satisfy $\xi^*$. By way of the correspondence $F_i \mapsto f_i$ we extend the definition of $v_I$ onto all terms of $R$. We denote this extension by $v^E_I$.

Now we have the following fact which corresponds to Theorem 4.3.2:

"Given programs $\xi_1$, $\xi_2$ and terms $T_1$, $T_2$, then

$$v^E_I \llbracket T_1 \rrbracket = v^E_I \llbracket T_2 \rrbracket$$

for all interpretation $I$ if and only if

$$C(\Sigma_{\xi_1}(T_1)) = C(\Sigma_{\xi_2}(T_2))"$$

which says the semantic equivalence can be described by the equivalence of the infinite expansion (C-function). It is easy to see that "given a program $\xi$ and a term $T$, $v^E_I \llbracket T \rrbracket = \bot_{D_I}$ for all interpretations $I$ if and only if $\Sigma_{\xi}(T) \in ^* \Lambda$ has no head normal form."

On the other hand, in a straightforward application of the formal language theory, we see that the property:
P(ξ, T) ≜ \"v^x_1 [ T ] = \perp^y_1 \forall interpretations I\" 

is decidable. So the head normality is a decidable property on R.

However, we do not know the answer to the following* question:

"Given programs \( \xi_1, \xi_2 \) and terms \( T_1, T_2 \), is it decidable whether or not 

\[ C(\xi_1 (T_1)) = C(\xi_2 (T_2)) \] 

which is equivalent to:

"Given programs \( \xi_1, \xi_2 \) and terms \( T_1, T_2 \), is it decidable whether or not for all interpretations I 

\[ v^x_1 [ T_1 ] = v^y_1 [ T_2 ] \] 

This property is, of course, undecidable on A, for the head normality is already undecidable on A.

*This problem is equivalent to the equivalence problem of the deterministic pushdown automata. Refer to B. Courcelle, "Recursive schemes, algebraic trees, deterministic languages," Proceedings of the 15th SWAT Symposium (1974).
§2. Algol-like Language

We take the translation algorithm based on the continuation technique in [1]. The continuation is explained as follows:

Given a program:

\[ S = S_1; S_2; \ldots; S_n \]

where \( S_i \)'s are a (block of) statement(s).

We can regard \( S \) as a function over the program domain \( D \) and ';;' is understood in the following two ways:

(I) Each block \( S_i \) is a function over \( D \). Thus ';;' is the composition of two functions. Let \( f_{S_i} \) be the \( \lambda \)-expression that corresponds to \( S_i \). Then the translation of \( S \) is:

\[ \lambda x. f_{S_n}(f_{S_{n-1}}(\ldots(f_{S_1}(x)))) \]

(II) Consider \( S; S' \). Let \( f' \) be the function over \( D \) which is defined by \( S' \). We regard \( S \) as a functional \( S \) which, applied \( f' \), yields a new function. So the translation of \( S; S' \) is

\[ \lambda x. (\phi_S(f'))(x) \]

If \( S' \) is null, \( f' \) is \( I = \lambda x.x \). So, for example, the translation of \( S_1; S_2 \) is

\[ \lambda x. (\phi_{S_1}(\phi_{S_2}(I)))(x) \]

For \( S = S_1; S_2; \ldots; S_n \), we give

\[ \lambda x. (\phi_{S_1}(\phi_{S_2}(\ldots(\phi_{S_n}(I))\ldots))(x) \]

(I) is not accurate when the program contains such statements as
goto or halt since, in that case, execution of the program is not necessarily sequential.

Here we show how some of the program constructs of Algol can be translated into λ-expressions based on (II).

Given a program

\[ S \equiv S_1; S_2; \ldots; S_n \]

each \( S_i \) is translated into a λ-expression of the form:

\[ s_i \equiv \lambda x_1 x_2 \ldots x_m. S_i(x_1, x_2, \ldots, x_m, \phi) \]

where \( S_i \) is a λ-expression that contains the variables \( x_1, x_2, \ldots, x_m, \phi \). The \( x_i \)'s are the program variables and \( \phi \) is called the continuation variable and stands for the remaining part of the program execution that follows the execution of \( S_i \).

Now \( S \) is translated into:

\[ s_1(s_2(\ldots(s_n I)\ldots)) \]

Algorithm. We state the translation algorithm in [1] for some of the important program constructs. For the complete and detailed description, we refer to [1]. For simplicity, we do not consider the block structures and the program is assumed to have the global variables \( x_1, x_2, \ldots, x_n \).

i) Assignment: \( x_j \leftarrow f(x_1, x_2, \ldots, x_n) \) is translated as:

\[ \lambda x_1 x_2 \ldots x_n. \phi x_1^j x_2 \ldots x_i^j f(x_1, x_2, \ldots, x_n) x_{i+1} \ldots x_n \]

ii) Conditional Statement: if \( \alpha \) then \( S_1 \) else \( S_2 \)

where \( \alpha \equiv \alpha(x_1, x_2, \ldots, x_n) \) is a Boolean expression. Let \( s_1 \).
and $s_2$ be the translation of $S_1$ and $S_2$, respectively. Let $\langle \alpha \rangle \equiv \langle \alpha(x_1, x_2, \ldots, x_n) \rangle$ be the translation of $\alpha$ such that

- $\langle \alpha \rangle AB \xrightarrow{B} A$ if $\alpha(x_1, x_2, \ldots, x_n) = \text{true}$
- $\langle \alpha \rangle AB \xrightarrow{B} B$ if $\alpha(x_1, x_2, \ldots, x_n) = \text{false}$

Then if $\alpha$ then $S_1$ else $S_2$ is translated as

$$\lambda x_1x_2\cdots x_n. \langle \alpha \rangle ((s_1) x_1x_2\cdots x_n)((s_2) x_1x_2\cdots x_n)$$

iii) goto \(\ell\): We associate a certain part of the program $P$ to each label \(\ell\). Let $m$ be the label which is defined in $P$ next to $\ell$ and $\ell$ and $m$ occur in $P$ as

$$\ell: S_1; S_2; \ldots; S_q; m: S_{q+1}$$

Then we associate $S_1; S_2; \ldots; S_q$ to $\ell$. So the translation of $\ell$ is:

$$[\ell] \equiv s_1(s_2(\cdots(s_q([m])\cdots))$$

where $[m]$ is the translation of $m$ and $s_i$ is the translation of $S_i$ for $1 \leq i \leq q$. If no label appears after $\ell$,

$$[\ell] \equiv s_1(s_2(\cdots(s_q(I))\cdots))$$

Now goto $\ell$ is translated as:

$$\lambda x_1x_2\cdots x_n.[\ell]x_1x_2\cdots x_n$$

Since goto $\ell$ forgets the statements following itself, $\phi$ does not occur in $[\ell]x_1x_2\cdots x_n$. 
iv) **while** $\alpha$ **do** $S$: This statement can be regarded as $W$ which is recursively defined as:

$$W \equiv \text{if } \alpha \text{ then begin } S; W \text{ end else no action}.$$ 

Since **if** $\alpha$ **then** begin $S; W$ **end** is translated into

$$\lambda x_1x_2\cdots x_n.\langle \alpha \rangle \left( (s(wf))x_1x_2\cdots x_n \right) (\phi x_1x_2\cdots x_n),$$

the translation of $W$, $w$ satisfies the equation:

$$w \beta \rightarrow \lambda x_1x_2\cdots x_n.\langle \alpha \rangle \left( (s(wf))x_1x_2\cdots x_n \right) (\phi x_1\cdots x_n).$$

So

$$w = Y(\lambda x_1x_2\cdots x_n.\langle \alpha \rangle \left( (s(f\phi))x_1\cdots x_n \right) (\phi x_1\cdots x_n)).$$

**Example.** Consider the following program:

```
begin
    input(x,y);
    i := x;
    while i>0 do
        begin y := y^2;
            if y>x^2 then goto l;
            i := i-1;
        end;
    l: end
```

**Translation:**

(1): We regard the input as an assignment and have

$$A \equiv \lambda x yi.\phi abi$$

(2): $B \equiv \lambda x yi.\phi xy$ 

(4): $C \equiv \lambda x yi.\phi xy^2i$
Now the whole program \( P = A(B(FI)) \). It is easy to see that

\[
P \xrightarrow{\beta} \lambda xyi. <a>0>(<b^2>a^2)(<a-1>0>(<b^4>a^2)(ab^4a-1)(\ldots \ldots)ab^2a-1)aba
\]

which shows all possible executions for arbitrary inputs or the infinite expansion of the program.

In the example, one might see the correspondence between \( \beta \)-reductions and program execution.

Next we ask to what programming concept the head-normality and the normality correspond under this translation. If we assume that the computation of each Boolean function terminates, the following is the answer:

"A integral part of a program is translated to a non-head normal \( \lambda \)-expression if and only if under any assignment of the Boolean values (i.e. true and false) to the Boolean functions occurring in the part, execution can never leave the part once it enters it." (Note that this property is, obviously, decidable.)

For example,

\[
\$: \text{goto } \$
\]

This goto statement is translated into
\( G \equiv \lambda x_1 x_2 \cdots x_n. [\ell] x_1 x_2 \cdots x_n \) and \([\ell] = \lambda \phi. G(\cdots)\). Obviously, \( G \) has no head normal form. On the other hand, a normal expression corresponds to a program that has no loop in it, i.e. no `while`, no `goto` that makes a loop. Thus, a normal expression is a program which terminates upon all inputs. On the other hand, a non-normal, head-normal expression corresponds to a program that may or may not terminate depending on the input condition.

Now we have the following observations. Although the discussion to support these conclusions is informal and rather shallow, they might give some intuitive insight and understanding to the formal argument in the rest of the chapters.

(i) The process to generate \( C(x) \) from \( x \in A \) corresponds to program expansion or execution upon arbitrary inputs.

(ii) The \( \Omega \)-conversion corresponds to removal of meaningless parts in the program (such as \( \ell: \text{goto} \ \ell \)).

(iii) Theorem 4.3.2 is understood as "two programs have the same meaning if (and only if) they have the same infinite expansion."
We generalize $\lambda$-expressions to the infinite $\lambda$-expressions. The results on the $\lambda$-expressions in $D_\infty$ are extended to the infinite $\lambda$-expressions. It is shown that the lattice structure of the infinite $\lambda$-expressions (including the conventional $\lambda$-expressions) induced by the $D_\infty$-partial order is equivalent to a directed complete partially ordered set $C_{\inf}$, which can be regarded as the domain of all the infinite expansion of the $\lambda$-expressions. Since $C_{\inf}$ is defined independent of $D_\infty$, $C_{\inf}$ can be said to give a natural lattice structure of the $\lambda$-calculus.
§1. Infinite Programs

How can a program be infinite? Probably in three ways.

1) non-termination, i.e. run time is infinite.

2) infinite work area, e.g. a Turing Machine is an infinite-program in the sense that it has an infinite storage.

3) infinitely many commands, e.g. an Algol program which is textually infinite.

Here, by an infinite program, we mean one in the category 3) above.

However, one may ask how such a program can be realized. In [12], Reynolds presents the following programming environment.

Let us imagine an interactive situation in which a person is programming in front of a terminal. He builds up his program in such a way that some of the integral parts (e.g., inside of a begin-end block, a procedure body, or simply a statement) are left unspecified. He can let the system execute this program. When it turns out that the system needs the specification of an undefined part of the program to continue execution, the programmer is requested to fill it with a code which could have several unspecified parts, too. The programmer meets this request probably considering the outcome of execution he has obtained so far. This process of programming can continue infinitely. Since a person with free will takes part in this process, it can become a non-recursively enumerable object.

If we are to formalize this idea of infinite programs, we shall probably have infinite λ-expressions. Then, what do infinite λ-expressions look like?
A "\(\lambda\)-like-expression" can be infinite in two ways:

1) Infinitely wide expressions

a) number of applications: We define \(x\) by

\[
x := \cdots ((A_1 A_2) A_3) \cdots) A_n A_{n+1}) \cdots
\]

that is, \(x\) is the outcome of infinite applications

\[
x_1 := A_1 \\
x_2 := x_1 A_2 \\
\vdots \\
x_n := x_{n-1} A_{n-1} \\
\vdots
\]

b) number of abstractions: Let \(x\) be an expression of the form

\[
x := \lambda v_1 v_2 v_3 \cdots v_n \cdots. w
\]

that is, \(x\) is a computation process which, given an infinite sequence of inputs, \(\{A_1, A_2, \ldots, A_n, \ldots\}\), returns

\[
x A_1 \rightarrow \lambda v_2 v_3 \cdots v_n \cdots. \int_{A_1}^{v_1} w \\
x A_1 A_2 \rightarrow \lambda v_3 v_4 \cdots v_n \cdots. \int_{A_1, A_2}^{v_1, v_2} w \\
\vdots \\
x A_1 A_2 \cdots A_k \rightarrow \lambda v_{k+1} v_{k+2} \cdots. \int_{A_1, A_2, \ldots, A_k}^{v_1, v_2, \ldots, v_k} w \\
\vdots
\]
c) combination of a) and b): For example,

\[ x := \lambda v_1 v_2 \ldots v_n \ldots x_1 x_2 x_3 \ldots, \]

that is,

\[ x := \lambda v_1 v_2 \ldots v_n \ldots w \]

where \( w := (\ldots(x_1 x_2) x_3) x_4 \ldots) x_n \ldots. \)

In a sense, the \( C \)-function has this structure. To apply infinitely many \( n \)-abstraction is to have an infinite expression:

\[ \lambda t_1 t_2 t_3 \ldots x t_1 t_2 t_3 \ldots \]

from \( X \in \Lambda. \)

2) Infinitely deep expressions: Consider such an expression as

\[ x := x_1 (x_2 (x_3 (\ldots (x_n (\ldots)) \ldots)) \ldots) \]

\( x \) would be the outcome of the application

\[ x_1 y_2 \]

where \( y_2 \) would be the outcome of the application

\[ x_2 y_3 \]

where \( y_3 \) \ldots

\[ \ldots \]

where \( y_n \) would be the outcome of the application

\[ x_n y_{n+1} \]
where \( y_{n+1} \ldots \)

... .

We will mainly study this <infintely deep \( \lambda \)-expression> in this chapter.

It is more likely that <infinitely deep \( \lambda \)-expressions> reflect the infinity of Reynold's infinite program. Let us take the following sequence of Algol-like commands:

\[
S \equiv \text{begin } S_1; S_2; \ldots ; S_n \text{ end} .
\]

As we saw in Chapter 5, there are two methods to translate \( S \) into a \( \lambda \)-expression.

1) Regard \( S_i \) (\( i = 1, 2, \ldots , n \)) as a function: \( \mathbb{D} \rightarrow \mathbb{D} \). Let \( s_i \) be the \( \lambda \)-expression translated from \( S_i \). Since \( S \) is the map: \( \mathbb{D} \rightarrow \mathbb{D} \) which is the composition of all \( S_i \)'s, the translation of \( S \) is:

\[
\lambda v.s_n(s_{n-1}(\ldots (s_1(v))\ldots )) .
\]

2) Regard \( S_i \) (\( i = 1, 2, \ldots , n \)) as a functional: \( (\mathbb{D} \rightarrow \mathbb{D}) \rightarrow (\mathbb{D} \rightarrow \mathbb{D}) \). Let \( s_i \) be the \( \lambda \)-expression translated from \( S_i \). Using the technique of continuation in Chapter 5, the translation of \( S \) is:

\[
\lambda v.s_1(s_2(\ldots (s_n(I))\ldots ))(v)
\]

where \( I \) is \( \lambda x.x \).

In both 1) and 2), we would have an infinitely deep expression letting \( n \rightarrow \infty \). (However, here, note that 2) is more appropriate as we see in the following discussion.)
Given an infinite program:

\[ S_1; S_2; \ldots ; S_n; \ldots, \]

this program will probably be the limit of the sequence:

\[ S_1; \bot \]
\[ S_1; S_2; \bot \]
\[ \vdots \]
\[ S_1; S_2; \ldots ; S_n; \bot \]

Using 2), we have

\[ \lambda v.(s_1(\bot))v \]
\[ \lambda v.(s_1(s_2(\bot)))v \]
\[ \vdots \]
\[ \lambda v.(s_1(s_2(\cdots(s_n(\bot))\cdots))v \]

So, probably, the infinite program above will be translated as:

\[ \bigcup_{n=0}^{\infty} \lambda v.(s_1(s_2(\cdots(s_n(\bot))\cdots))v \]

We will formalize this idea in §3.
§2. Characterization of C-functions

As in Chapter 3, let $\mathcal{C} = \{c \mid c \in \Delta \rightarrow V \cup \{\omega}\}$. The C-function is a map: $\Delta \rightarrow \mathcal{C}$. It is easy to see that the range of $\mathcal{C}$ is only a proper subset of $\mathcal{C}$, i.e., $\mathcal{C}(\Delta) \subsetneq \mathcal{C}$, but what sort of subset is $\mathcal{C}(\Delta)$?

In fact, $\mathcal{C}$ is of too arbitrary structure to attract any interest. The following conditions characterize the hierarchy of some interesting subclasses of $\mathcal{C}$.

Given $c \in \mathcal{C}$.

Condition 1: If $c(\delta) = z \in V$, then either $z$ is free or $z = t_{\delta'}$ for $\delta' \in \Delta$ where $\delta' \leq \delta$ or $\delta' = \delta \circ m$ for some $m \in \mathbb{N}$ (i.e. if a variable is bound, it must be so in an outer context).

Condition 2: If $c(\delta) = \omega$ for some $\delta \in \Delta$, $c(\delta') = \omega$ for any $\delta' \in \Delta$ with $\delta < \delta'$ (i.e. once a subexpression turns out to be bottom, any of its descendents must be bottom, too).

Condition 3: If $c(\delta) \neq \omega$, there exists an integer $k_{\delta}^C$ and a positive integer $N_{\delta}^C$ such that, for all $n > N_{\delta}^C$, $c(\delta \circ n) = t_{\delta \circ (n+k_{\delta}^C)}$ and $c(\delta \circ n \circ \delta') = t_{\delta \circ n \circ \delta'}$ for all $\delta' \in \Delta$ (i.e. $c$ is 'finitely wide').

Condition 4: Let $\text{Fr}(c) = \{z \mid z \in F, c(\delta) = z \text{ for some } \delta \in \Delta\}$. Then $\#(\text{Fr}(c)) < \infty$ (i.e. the number of the distinct variables which occur in $\{c(\delta) \mid \delta \in \Delta\}$ is finite).

Condition 5: There are partially computable functions $\phi_c : \Delta \rightarrow \mathbb{N}$ and $\psi_c : \Delta \rightarrow V$ such that $\phi_c(\delta) = N_{\delta}^C$ and $\psi_c(\delta) = z$ if $c(\delta) = z \in V$, $\phi_c(\delta)$ and $\psi_c(\delta)$ are undefined if $c(\delta) = \omega$ (i.e. $\{c(\delta) \mid c(\delta) \neq \omega, \delta \in \Delta\}$ is a recursively enumerable object and the width in Condition 3 is also partially computable).
6.2.1 Theorem. Each element of $C(\Lambda)$ satisfies Conditions 1-5.

Proof. Conditions 1 and 2 are obviously satisfied by the definition of $C$. Let $x$ be one of the $\lambda$-expression such that $C(x) = c$.

Condition 3: Since $C(x, \delta) \neq \omega$, $L(x, \delta) \neq \Omega$. Let

$$L(x, \delta) \xrightarrow{\alpha \beta} \lambda t_{\delta \circ 1} t_{\delta \circ 2} \cdots t_{\delta \circ p} z X_1 X_2 \cdots X_q$$

Now set $N^c_\delta = q$ and $k^c_\delta = p - q$.

Condition 4: Since $x$ is finite, $x$ can contain at most finite number of distinct free variables.

Condition 5: Obvious from the definition of $C$. □

The converse of Theorem 6.2.1 is true as demonstrated in Theorem 6.2.2.

$C_{\text{fin}}$ and $C_{\text{inf}}$ are subclasses of $C$ defined as follows:

$$C_{\text{fin}} = \{ c \mid c \in C, c \text{ satisfies Conditions 1-5} \}$$

$$C_{\text{inf}} = \{ c \mid c \in C, c \text{ satisfies Conditions 1-3} \}$$

We have the sequence: $C_{\text{fin}} \subseteq C_{\text{inf}} \subseteq C$. The smallest class $C_{\text{fin}}$ is, in fact, the same as $C(\Lambda)$ as proved in the following theorem.

6.2.2 Theorem. Let $c$ be in $C$. If $c$ satisfies Conditions 1-5, then there exists a $\lambda$-expression $x$ such that $C(x) = c$.

Proof. We give effective codings of $Z, \Lambda$ and $V$ into $\Lambda$ as:
We assume that $\text{En}(\mathbb{Z}), \text{En}(\Delta), \text{En}(T_\Delta)$ and $\text{En}(F) = F$ are mutually disjoint.

Given $c \in C_{\text{fin}}$, let $\Delta_c$ be the subset of $\Delta$ consisting of all $\delta$ such that $\phi_c(\delta)$ is defined. Obviously, $\Delta_c$ is recursively enumerable.

In the rest of the proof, we depend on the following fact due to Kleene:

"For each partial recursive function $\phi: \mathbb{N} \rightarrow \mathbb{N}$, there exists a $\lambda$-expression $\tilde{\phi} \in \Lambda$ such that

$$\tilde{\phi}\bar{m} \beta m \text{ if } \phi(n) = m.$$

$\tilde{\phi}\bar{m}$ has no head normal form if $\phi(n)$ is undefined.

where $\bar{m}, \bar{n}$ are the encodings of $m, n \in \mathbb{N}$ in $\Lambda$." (For the proof of the proposition above, see, for example, [2].)

We define $\pi_c \in \Lambda$ by:

$$\pi_c \delta \xrightarrow{\text{CNV}} \begin{cases} 
\text{a } \lambda\text{-expression without a head normal form} & \text{if } \delta \notin \Delta_c \\
\lambda x. x & \text{if } \delta \in \Delta_c
\end{cases}$$

A partially computable function $M_c: \Delta \rightarrow \mathbb{N}$ is defined by:

$$M_c(\delta) = \begin{cases} 
\text{undefined if } c(\delta) = \omega \\
k_\delta^c + \phi_c(\delta) & \text{if } c(\delta) \neq \omega
\end{cases}.$$
\( P \in \Lambda \) is defined by:

\[
P \overset{\text{CNV}}{\to} \delta i \quad \text{for } i \in \mathbb{N} \text{ and } \delta \in \Delta.
\]

Finally we define \( \Theta_c \in \Lambda \) by the following recursive equation:

\[
\Theta_c \overset{\text{CNV}}{\to} \pi_c \delta (f_{\delta M_c}(\delta) \phi_c(\delta) c(\delta) e)
\]

where \( f \in \Lambda \) is defined by:

\[
f_{\delta m} \overset{\text{CNV}}{\to} \begin{cases} 
\text{if } i = m \\
\lambda s.f_{\delta i+m}(N_{\delta i+1} e) \text{ otherwise}
\end{cases}
\]

where \( g \in \Lambda \) is given by

\[
g_{\delta j} \overset{\text{CNV}}{\to} \begin{cases} 
\text{if } j = n \\
g_{\delta j+m} \phi(\Theta_c(P_{\delta j} \delta) e) \text{ otherwise}
\end{cases}
\]

and \( N \in \Lambda \) is given by:

\[
N_{\delta i} \overset{\text{CNV}}{\to} \begin{cases} 
s \text{ if } z = t_{\delta i} \\
e \end{cases}
\]

Note that \( s \) at (**) is the same as the bound variable at (*).

Now we assert that \( C(\Theta_c \tilde{O} \overline{I}) = c \). To prove this, we show that there exists \( e_{\delta} \in \Lambda \) for each \( \delta \in \Delta_c \) such that:

1. \( \delta = 0 \):
   a. If \( c(0) = \omega \), then \( \Theta_c \tilde{O} \overline{I} \) is non-head normal.
   b. If \( c(0) = v \in V \), \( \phi_c(0) = q \) and \( q + k_c = p \),
      
      \[
      \Theta_c \tilde{O} \overline{I} \overset{\text{CNV}}{\to} \lambda s_1 s_2 \cdots s_{p} v(\Theta_c \tilde{e}_0)(\Theta_c \tilde{e}_0) \cdots (\Theta_c \tilde{e}_0).
      \]
(2) \( \delta = \delta' \odot \delta \neq 0 \)

(a) If \( c(\delta) = \omega \), then \( \Theta_{c} \tilde{e}_{\delta} \), is non-head normal.

(b) If \( c(\delta) \in V \), \( \phi_{c}(\delta) = q \) and \( q + k_{\delta} = p \), then

\[
\Theta_{c} \tilde{e}_{\delta} \xrightarrow{\text{CNV}} \lambda r_{1} r_{2} \cdots r_{p} v(\Theta_{c} \delta \odot l_{0}e_{\delta})(\Theta_{c} \delta \odot 2e_{\delta}) \cdots (\Theta_{c} \delta \odot qe_{\delta})
\]

We only prove (1)-(a) and (b). (2)-(a) and (b) are proven similarly.

(1)-(a): Since \( c(0) = \omega \), \( 0 \notin A \) and so, \( \pi_{c} \tilde{0} \) is non-head normal. Thus,

\[
\Theta_{c} \tilde{0} \xrightarrow{\text{CNV}} \pi_{c} \tilde{0}(\cdots): \text{non-head normal}
\]

(1)-(b): Since \( c(0) \neq \omega \), \( \pi_{c} \tilde{0} \xrightarrow{\text{CNV}} I \)

\[
\Theta_{c} \tilde{0} \xrightarrow{\text{CNV}} I(\cdots)
\]

\[
\xrightarrow{\text{CNV}} f_{0}opq_{I}q_{I}
\]

\[
\xrightarrow{\text{CNV}} \lambda s_{1}. f_{0}tpq_{I}(N \tilde{0}T I)
\]

\[
\vdots
\]

\[
\xrightarrow{\text{CNV}} \lambda s_{1}s_{2} \cdots s_{p}. f_{0}ppq_{I}(N \tilde{0}p(N \tilde{0}p(\cdots(N \tilde{0}T I)) \cdots)
\]

(Set \( e_{0} = N \tilde{0}p(N \tilde{0}p(\cdots(N \tilde{0}T I)) \cdots). \))

\[
\xrightarrow{\text{CNV}} \lambda s_{1}s_{2} \cdots s_{p}. g \tilde{0}q_{I}v_{e_{0}}
\]

\[
\xrightarrow{\text{CNV}} \lambda s_{1}s_{2} \cdots s_{p}. g \tilde{0}Iq_{I}v_{e_{0}}(\Theta_{c}(p \tilde{0})e_{0})
\]

\[
\xrightarrow{\text{CNV}} \lambda s_{1}s_{2} \cdots s_{p}. g \tilde{0}Iq_{I}v_{e_{0}}(\Theta_{c} 0 \tilde{c}e_{0})
\]

\[
\vdots
\]

\[
\xrightarrow{\text{CNV}} \lambda s_{1}s_{2} \cdots s_{p}. g \tilde{0}q_{I}v_{e_{0}}(\Theta_{c} \tilde{c}e_{0})(\Theta_{c} \tilde{c}e_{0}) \cdots (\Theta_{c} \tilde{c}e_{0})
\]

This completes the proof for (1)-(b). \( \square \)
Probably $\theta: \mathbb{C}_{\text{fin}} \rightarrow \Lambda$ ($\theta: c \mapsto \Theta_c$) corresponds to the universal Turing machine.

6.2.3 Corollary. $\mathbb{C}_{\text{fin}} = C(\Lambda)$.

Now Theorem 4.3.2 can be stated as:

$$\mathbb{C}_{\text{fin}} = \Lambda/_{D_\infty}.$$

In the rest of this chapter, we shall mainly study $\mathbb{C}_{\text{inf}}$. 
§3. Infinite λ-expressions

In this section, we formalize the idea of infinitely-deep λ-expressions. We utilize the process of generating infinite programs given in §1 to define the infinite λ-expressions $\Lambda^\infty$.

6.3.1 Definition.

a. $\Lambda$ is the set of the expressions to be defined by:

1) A variable $v \in U$ alone is in $\Lambda$.

2) $\square \in \Lambda$.

3) If $\xi$, $\zeta$ are in $\Lambda$, so is $\xi (\zeta)$.

4) If $\eta$ is in $\Lambda$ and $v \in U$ is a variable, then $\lambda v. \eta$ is in $\Lambda$.

b. Let $\xi$, $\zeta \in \Lambda$. We say that $\zeta$ is a specification of $\xi$ if either $\xi = \zeta$ or $\zeta$ derives from $\xi$ by replacing some occurrences of $\square$ in $\xi$ with elements in $\Lambda$. (We write as $\zeta$ spec $\xi$.)

c. Given $\zeta \in \Lambda$, we define $\zeta^*$ in $\Lambda$ to be the λ-expression which is derived from $\zeta$ by replacing each $\square$ in $\zeta$ by $\Omega$.

d. $\Lambda^\infty$, infinite λ-expressions, is the set of all sequences

$$(\zeta_1, \zeta_2, \ldots, \zeta_n, \ldots)$$

where $\zeta_i \in \Lambda$ and $\zeta_{i+1}$ spec $\zeta_i$ for each $i = 1, 2, \ldots$, that is,

$$\Lambda^\infty = \{ \eta \mid \eta = (\zeta_1, \zeta_2, \ldots), \zeta_i \in \Lambda \text{ and } \zeta_{i+1} \text{ spec } \zeta_i \text{ for each } i = 1, 2, \ldots \}$$
Since $\Lambda$ can be regarded as a subset of $\Lambda^0$ by the obvious injection: $\Lambda \rightarrow \Lambda^0$, we can embed $\Lambda$ into $\Lambda^\infty$ as follows: Let $x \in \Lambda$, $i: x \rightarrow (x,x,x,x,...) \in \Lambda^\infty$ by $i: \Lambda \rightarrow \Lambda^\infty$, we regard as $\Lambda \subseteq \Lambda^\infty$. We define $\Lambda^\infty_\alpha$ to be the set of the infinite $\lambda$-expressions which do not contain any free variables, i.e.

$$\Lambda^\infty_\alpha = \{ \zeta | \zeta = (\zeta_1,\zeta_2,...,\zeta_n) \in \Lambda^\infty_\alpha \text{ where } \zeta_i \text{ has no free variables for each } i \}.$$

The restriction of $i$ to $\Lambda^\infty_\alpha$, $i|_{\Lambda^\infty_\alpha}$ gives the inclusion: $\Lambda^\infty_\alpha \subseteq \Lambda^\infty_\alpha$.

Given $\eta = (\zeta_1,\zeta_2,...,\zeta_n,...)$ in $\Lambda^\infty$, each $\zeta_i$ can be looked upon as a program which has some unspecified parts. $\Box$'s occurring in $\zeta_i$ are the unspecified parts. $\zeta_i+1$ is obtained by filling $\Box$ in $\zeta_i$ with another $\xi$ of $\Lambda^\infty$. This process eventually leads us to the infinite $\lambda$-expression $\eta$.

As we mapped $\Lambda$ into $D_\infty$ through $\mathcal{V}$, we are to define a semantic function $W_\infty$ of $\Lambda^\infty$ into $D_\infty$.

6.3.2 Definition (Semantics $W_\infty$ of $\Lambda^\infty$). $U$ is the set of all variables of discourse and $Env$ is the set of all functions: $U \rightarrow D_\infty$.

Now, $W_\infty: EN \rightarrow (\Lambda^\infty + D_\infty)$ is the following map: Given $\eta = (\zeta_1,\zeta_2,...) \in \Lambda^\infty$ and $\rho \in EN$.

$$W_\infty[\eta]\rho = \bigcup_{i=1}^{\infty} W[\zeta_i] \rho$$

We should note that, in Definition 6.3.2,

$$W[\zeta_i] \rho \subseteq W[\zeta_{i+1}] \rho \quad \text{for each } i.$$
So, \( W_\omega[[\eta]] \rho \) is the least upper bound of a directed sequence in \( D_\omega \).

6.3.3 **Corollary.** Let \( x \) be in \( \Lambda \). Then

\[
W[[x]] \rho = W_\omega[[i(x)]] \rho
\]

\[\square\]

6.3.4 **Definition.** Given \( \xi \) and \( \zeta \) in \( \Lambda^\omega \), we define the **application of** \( \xi \) to \( \zeta \), \( \xi(\zeta) \in \Lambda^\omega \), as follows: Let

\[
\xi = (\xi_1, \xi_2, \xi_3, \ldots) \\
\zeta = (\zeta_1, \zeta_2, \zeta_3, \ldots)
\]

Then

\[
\xi(\zeta) = (\xi_1(\zeta_1), \xi_2(\zeta_2), \xi_3(\zeta_3), \ldots)
\]

It is easy to verify that

\[
W_\omega[[\xi(\zeta)] \rho = W_\omega[[\xi]] \rho (W_\omega[[\zeta]] \rho)
\]

for \( \xi, \zeta \) in \( \Lambda^\omega \).

Now, we are ready to consider the correspondence between \( \Lambda^\omega \) and \( C_{\text{inf}} \). In fact, the similar relation holds between \( \Lambda^\omega \) and \( C_{\text{inf}} \) to that between \( \Lambda \) and \( C_{\text{fin}} \).

The following lemma is necessary to prove part 3 of Theorem 6.3.6.

6.3.5 **Lemma.**

1. Let \( x \) and \( y \) be \( \lambda \)-expressions which satisfy the condition of Theorem 3.4.7-1. Then, by the theorem, given any \( a, b \in D_\omega \), we can choose \( e_1, e_2, \ldots, e_n \in \Lambda \) and an environment
\[ W \llbracket [x_1 e_2 \ldots e_n] \rrbracket \rho = a \]

and

\[ W \llbracket [y_1 e_2 \ldots e_n] \rrbracket \rho = b . \]

Here, if, for \( x_1 \) and \( y_1 \in A \), \( x \sqsubseteq x_1 \) and \( y \sqsubseteq y_1 \), then

\[ W \llbracket [x_1 e_2 \ldots e_n] \rrbracket \rho = a \]

and

\[ W \llbracket [y_1 e_2 \ldots e_n] \rrbracket \rho = b . \]

2. Let \( x \) and \( y \) be in \( A \). Assume that, for \( \delta \in \Delta \), \( x \) and \( y \) satisfy the condition of Theorem 3.4.7-2. Then, by the theorem, given any \( a, b \in D_\infty \), we can choose \( e_1, e_2, \ldots, e_n \in A \) and an environment \( \rho \) for which

\[ W \llbracket [x_1 e_2 \ldots e_n] \rrbracket \rho = a \]

and

\[ W \llbracket [y_1 e_2 \ldots e_n] \rrbracket \rho = \bot . \]

Here, if, for \( x_1, y_1 \in A \), \( x \sqsubseteq x_1 \), \( y \sqsubseteq y_1 \) and \( C(y_1, \delta) = \omega \), then

\[ W \llbracket [x_1 e_2 \ldots e_n] \rrbracket \rho = a \]

and

\[ W \llbracket [y_1 e_2 \ldots e_n] \rrbracket \rho = \bot . \]

**Proof.** We prove only part 1 of the lemma. By the assumption of the lemma, there exists \( \delta \in \Delta \) which satisfies the following:
(*) For any \( \delta' \) with \( \delta' < \delta \), \( \hat{C}(x,\delta') = \hat{C}(y,\delta') \) and

\[
\hat{C}(x,\delta) = (u,i) \\
\hat{C}(y,\delta) = (v,j)
\]

where \( (u,i) \neq (v,j) \).

Since \( \hat{C}(x) \leq \hat{C}(x_1) \) and \( \hat{C}(y) \leq \hat{C}(y_1) \),

\[
\hat{C}(x,\delta') = \hat{C}(x_1,\delta') \neq \omega \\
\hat{C}(y,\delta') = \hat{C}(y_1,\delta') \neq \omega
\]

for each \( \delta' \) with \( \delta' < \delta \).

So, (*) still holds if we replace \( x \) by \( x_1 \) and \( y \) by \( y_1 \).

As is seen in the proof of Lemma 3.4.5, the choice of \( e_1, e_2, \ldots, e_n \) and \( \rho \) depends upon only \( C(x,\delta') \) and \( C(y,\delta') \) for \( \delta' < \delta \), from which our assertion follows immediately. \( \square \)

6.3.6 Definition. \( C_\infty : \Lambda^\infty \to \mathbb{C} \) is the following map: Given \( \zeta = (\zeta_1, \zeta_2, \ldots) \in \Lambda^\infty \),

\[
C_\infty(\zeta,\delta) = \begin{cases} 
\omega & \text{if } C(\zeta^*_i,\delta) = \omega \text{ for all } i = 1, 2, \ldots \\
z \in V & \text{if } C(\zeta^*_i,\delta) = z \text{ for some } i.
\end{cases}
\]

\( C_\infty \) is well defined since \( \zeta^*_i \subseteq D_\infty \zeta^*_i \) and so \( C(\zeta^*_i) \leq C(\zeta^*_i) \).

6.3.7 Theorem.

1. \( C_\infty|_\Lambda = C \)

2. \( C_\infty(\Lambda^\infty) = \mathbb{C}_{\text{inf}} \)
3. For all $\xi, \zeta \in \Lambda^\omega$,

$$\xi \subseteq \zeta \text{ iff } C_\infty(\xi) \leq C_\infty(\zeta)$$

(where $\xi \subseteq \zeta$ means $\forall \rho \in \Lambda^\omega \exists \rho \subseteq \xi \subseteq \zeta$ for all $\rho \in \mathbb{N}$ and $\subseteq$ is the partial order over $\mathbb{C}$ as in Chapter 3).

So, $C_{\text{inf}} = \Lambda^\omega / D_{\infty}$.

Proof. It is obvious that $C_\infty|_\Lambda = \mathbb{C}$, and $C_\infty(\Lambda^\omega) \subseteq C_{\text{inf}}$.

To prove that $C_\infty$ is surjective, we need a similar concept to admissible $\Delta$-trees in Chapter 4.

Let $c \in C_{\text{inf}}$. We say that a $\Delta$-tree, $T$, is admissible to $c$ if $T$ satisfies the following:

If $\delta \in T$, then $\delta^0, \delta^2, \ldots, \delta^N$ are also in $T$ where $N^c$ is as in Condition 3 of $C_{\text{inf}}$.

Now we define $\alpha^\nu(c, T) \in \Lambda^\square$ for $c \in C_{\text{inf}}$, a $\Delta$-tree, $T$, admissible to $c$, and $n \in \mathbb{N}$ (similar to $\Lambda^\nu_p(x, T)$ in Chapter 4).

$$\alpha^\nu(c, T) = \alpha^0_c \text{ where } \alpha^\delta_c (\delta \in T) \text{ is defined by:}$$

1) If $|\delta| = n$, $\alpha^\delta_c = \square$.

2) If $|\delta| < n$ and $c(\delta) = \omega$, $\alpha^\delta_c = \Omega$.

3) If $|\delta| < n$ and $c(\delta) = z$,

$$\alpha^\delta_c = \lambda t^{\delta_0} t^{\delta^1} \cdots t^{\delta_{\gamma_T(\delta)}} \cdot z^{\delta^0} a^{\delta_1} a^{\delta_2} \cdots a^{\delta_{\gamma_T(\delta)}}$$

where $k^c_\delta$ is as in Condition 3 of $C_{\text{inf}}$. 
Now it is easy to see that
\[ a^n(c,T) \in A_\square \text{ and } a^{n+1}(c,T) \in \text{spec } a^n(c,T) \]
for \( n = 0,1,2,\ldots \), and, letting \( \alpha = (a^0(c,T),a^1(c,T),a^2(c,T),\ldots) \), \( \alpha \in \Lambda^\infty \) and \( C_\infty(\alpha) = c \).

To prove the last part of the theorem, let us assume that
\[ C_\infty(\xi) \nsucceq C_\infty(\zeta) \]
for given \( \xi = (\xi_1,\xi_2,\ldots) \) and \( \zeta = (\zeta_1,\zeta_2,\ldots) \)
in \( \Lambda^\infty \). By the definition of \( C_\infty \), there exist \( i, j \in \mathbb{N} \) and \( \delta \in \Delta \) for which

- either \( C(\xi_i^*,\delta) = u \) and \( C(\zeta_j^*,\delta) = v \) with \( u \neq v \)
or \( C(\xi_i^*,\delta) \neq \omega \) and \( C(\zeta_j^*,\delta) = \omega \) for all \( k \in \mathbb{N} \).

Since \( \xi_k^* \subseteq \xi_{k+1}^* \) and \( \zeta_k^* \subseteq \zeta_{k+1}^* \) for each \( k \), by Lemma 6.3.5,
there exist \( e_1,e_2,\ldots,e_n \in \Lambda \) and an environment \( \rho \) for which there exists \( K > 0 \) such that for all \( m > K \),
\[
W [[\xi_m^*e_1e_2\ldots e_n]] \rho = K \\
W [[\zeta_m^*e_1e_2\ldots e_n]] \rho = H
\]

or
\[
W [[\xi_m^*e_1e_2\ldots e_n]] \rho = K \\
W [[\zeta_m^*e_1e_2\ldots e_n]] \rho = 1
\]

Since \( W_\infty [[\xi]] \rho = \bigcup_{k=1}^{\infty} W [[\xi_k^*]] \rho \) and \( W_\infty [[\zeta]] \rho = \bigcup_{k=1}^{\infty} W [[\zeta_k^*]] \rho \), by the definition of \( W_\infty \), we conclude that \( \xi \notin \infty \).

On the other hand, let us assume that \( C_\infty(\xi) \leq C_\infty(\zeta) \). We take a \( \Delta \)-tree, \( T \), which is admissible to both \( C_\infty(\xi) \) and \( C_\infty(\zeta) \).
We define $\beta(n,i)$ for $n = \xi$ or $\zeta$ and $i > 0$ as follows:

$\beta(n,i) = \beta^0(n,i)$ where $\beta^\delta(n,i)$ ($\delta \in T$) is defined by:

1) If $|\delta| = i$, then $\beta^\delta(n,i) = \Omega$.
2) If $|\delta| < i$ and $C(n^*,\delta) = \omega$, then $\beta^\delta(n,i) = \Omega$.
3) If $|\delta| < i$ and $C(n^*,\delta) = (z,k)$, then

$$\beta^\delta(n,i) = \lambda t_{\delta_1} t_{\delta_2} \cdots t_{\delta_k} (\gamma_T(\delta)+k).$$

In the first place, $\beta(\xi,i) \subseteq \xi^*_{\infty}$ and $\beta(\zeta,i) \subseteq \zeta^*_{\infty}$.

In the second place, since

$$\xi^*_i = \bigcup_{n=0}^{\infty} A_p^n(\xi^*_i,T)$$

and

$$\zeta^*_i = \bigcup_{n=0}^{\infty} A_p^n(\zeta^*_i,T),$$

and

$$A_p^n(\xi^*_i,T) \subseteq \beta(\xi,j)$$

for $j > \max(i,n)$

and

$$A_p^n(\zeta^*_i,T) \subseteq \beta(\zeta,j)$$

we conclude that

$$(*) \quad \zeta = \bigcup_{i=1}^{\infty} \beta(\xi,i) \text{ and } \xi = \bigcup_{i=1}^{\infty} \beta(\zeta,i).$$
Since $C_\infty(\xi) \leq C_\infty(\zeta)$, by the definition of $C_\infty$, it follows that, for any $p > 0$, there exists a sufficiently large $Q > 0$ such that

$$C(\beta(\xi, p)) \leq C(\beta(\zeta, Q)),$$

that is,

$$(**) \quad \beta(\xi, p) \subseteq \beta(\zeta, Q) \quad \text{D}$$

From (*) and (**), $\xi \subseteq \zeta$ is immediate. $\square$

6.3.8 Definition. Given a complete lattice $D$, let $E$ be a subset of $D$.

The directed completion of $E$ in $D$ is the set

$$\{a \mid \text{there is a directed set } F \subseteq E \text{ such that } a = \cup F\}.$$

6.3.9 Theorem. $\{W_\infty(\llbracket \xi \rrbracket) \mid \xi \in \Lambda^\infty_c\} \subseteq D_\infty$ is the directed completion of $\{W(\llbracket x \rrbracket) \mid x \in \Lambda^\infty_c\} \subseteq D_\infty$. Thus $\{W_\infty(\llbracket \xi \rrbracket) \mid \xi \in \Lambda^\infty_c\}$ is a directed complete subset of $D_\infty$.

(Note that the elements of $\Lambda^\infty_c$ and $\Lambda_c$ do not have any free variables. So, their values in $D_\infty$ do not depend on the environment $\rho \in EN$.)

Proof. The argument for the proof is similar to Theorem 6.3.7. $\square$

Theorem 6.3.9 shows that the relation between $\Lambda^\infty_c$ and $\Lambda_c$ is similar to that between the real numbers and the rational numbers.
Each real number is defined as the limit of a non-decreasing sequence of rational numbers. In this way, we may well regard $\Lambda^\infty$ as the generalization of $\Lambda$.

6.3.10 Corollary. The cardinality of $D_\infty$ is strictly larger than denumerable if $D_0 \neq \{I\}$.

Proof. Obviously, $C_\infty(\Lambda_c^\infty) (\subseteq C_{\text{inf}})$ has a cardinality strictly larger than denumerable. Since $\xi$ and $\zeta$ in $\Lambda_c^\infty$ are mapped to different elements in $D_\infty$ if $C_\infty(\xi) \neq C_\infty(\zeta)$, $D_\infty$ must have a cardinality strictly larger than denumerable. $\square$
§4. **Lattice Structure of \( C_{\text{fin}} \) and \( C_{\text{inf}} \)**

In the previous section, we generalized \( \Lambda \) to \( \Lambda^\infty \). Here we shall show that the lattice structure in \( \Lambda \) and \( \Lambda^\infty \) induced by the \( D_\infty \) partial order is equivalent to the lattice structure of \( C_{\text{inf}} \) which is a directed-complete partially ordered set. In addition, we shall examine the structure of \( C_{\text{inf}} \) and \( C_{\text{fin}} \).

6.4.1 **Proposition.** \( C_{\text{fin}} \) and \( C_{\text{inf}} \) are partially ordered sets with the order \( \subseteq \) defined by: For \( a, b \in C_{\text{inf}} \), \( a \subseteq b \) if and only if, for all \( \delta \in \Delta \),

\[
\text{either} \quad a(\delta) = \omega \\
\text{or} \quad a(\delta) = b(\delta) .
\]

6.4.2 **Proposition.** \( C_{\text{fin}} \) and \( C_{\text{inf}} \) lower semi-lattices. More precisely, we define \( a \cap b \in C_{\text{inf}} \) (\( C_{\text{fin}} \)), \( a, b \in C_{\text{inf}} \) (\( C_{\text{fin}} \)) as follows: Let \( m_\delta = \max(N_\delta^a, N_\delta^b) + 1 \) where \( N_\delta^a \) and \( N_\delta^b \) are as in Condition 3 of \( C_{\text{inf}} \). Define \( c = a \cap b \in C_{\text{inf}} \) (\( C_{\text{fin}} \)) by:

1) \( c(0) = \begin{cases} 
   a(0) & \text{if } a(0) \neq \omega \text{ and } a(m_0) = b(m_0) \\
   \omega & \text{otherwise.}
\end{cases} \)

2) Let \( \delta = \delta' \circ \alpha \) and suppose that \( c(\delta') \) is already defined.

   a) If \( c(\delta') = \omega \) then \( c(\delta) = \omega \).

   b) If \( c(\delta') \neq \omega \) then

\[
c(\delta) = \begin{cases} 
   a(\delta) & \text{if } a(\delta) = b(\delta) \neq \omega \text{ and } a(\delta \circ \delta\alpha) = b(\delta \circ \delta\alpha) \\
   \omega & \text{otherwise.}
\end{cases}
\]
Proof. If \( a, b \in \mathcal{C}_{\text{inf}} \), \( c \in \mathcal{C}_{\text{inf}} \) since, obviously, \( c \) as defined above satisfies Conditions 1 to 3 of \( \mathcal{C}_{\text{inf}} \). In case \( a, b \in \mathcal{C}_{\text{fin}} \), we only have to show that \( c \) satisfies Conditions 4 and 5 of \( \mathcal{C}_{\text{fin}} \) to prove that \( c \in \mathcal{C}_{\text{fin}} \). Since \( c \) cannot contain any free variable that does not appear in \( a \) or \( b \), \( c \) satisfies Condition 4. On the other hand, \( c(\delta) \) for each \( \delta \in \Delta \) is computed in the following way: If \( \psi_a(\delta) \) and \( \psi_b(\delta) \) are both defined and \( \psi_a(\delta) = \psi_b(\delta) \) and if \( \psi_a(\delta \cdot (\max(\phi_a(\delta), \phi_b(\delta)) + 1)) = \psi_b(\delta \cdot (\max(\phi_a(\delta), \phi_b(\delta)) + 1)) \) then \( c(\delta) = \psi_a(\delta) \). Otherwise \( c(\delta) = \omega \).

This statement guarantees that there exists a partially computable function \( \psi_c : \Delta \rightarrow \mathbb{V} \) that satisfies Condition 5. Also \( \phi_c : \Delta \rightarrow \mathbb{N} \) is defined by:

\[
\phi_c(\delta) = \begin{cases} 
\text{undefined} & \text{if } \psi_c \text{ is undefined} \\
\max(\phi_a(\delta), \phi_b(\delta)) & \text{otherwise.}
\end{cases}
\]

On the other hand it is easy to see that there is no \( d \in \mathcal{C}_{\text{inf}} \) such that \( c \not\subseteq d \) and both \( d \subseteq a \) and \( d \subseteq b \). \( \Box \)

6.4.3 Corollary. Given \( x, y \in \Lambda \), there exists a \( \lambda \)-expression \( z \in \Lambda \) such that

\[
C(z) = C(x) \cap C(y)
\]

Proof. Immediate from Proposition 6.4.2 and Theorems 6.2.1 and 6.2.2. \( \Box \)

Given \( x, y, z \in \Lambda \), if \( C(z) = C(x) \cap C(y) \), \( z \subseteq x \cap y \) by Theorem 4.3.2.
However it is not generally the case that $z = x \cap y$.

6.4.4 Counterexample. Let

$$X = \lambda xyz. x \Omega z$$
$$Y = \lambda xyz. x \Omega y$$
$$Z = \lambda xyz. x \Omega \Omega$$.

Obviously, $C(Z) = C(X) \cap C(Y)$, but when $D_\infty$ is continuous,*

$$(u \cap v)(w) = u(w) \cap v(w)$$
for $u, v, w \in D_\infty$, and so,

$$Z(\lambda ab.a \cup \lambda ab.b) \rightarrow \Omega$$
$$X(\lambda ab.a \cup \lambda ab.b) \rightarrow I$$
$$Y(\lambda ab.a \cup \lambda ab.b) \rightarrow I$$.

So

$$Z \notin X \cap Y$$.

6.4.5 Proposition. Any directed subset of $C_{\inf}$ has its least upper bound in $C_{\inf}$. So $C_{\inf}$ is directed-complete.

Proof. Let $D$ be any directed set of $C_{\inf}$. $d \in C_{\inf}$ as defined below gives the least upper bound of $D$:

$$d(\delta) = \begin{cases} 
\omega & \text{if } c(\delta) = \omega \text{ for all } c \in D \\
z & \text{if } c(\delta) = z \text{ for some } c \in D
\end{cases}.$$ 

d is well defined by Theorem 6.3.7-3 since $D$ is directed. It is easy to see that $d$ satisfies Conditions 1-3 of $C_{\inf}$. □

The following proposition shows that the lattice topology of $\Lambda$ and $\Lambda^\infty$ induced by $D_\infty$ partial order is, in fact, equivalent to the lattice topology of $C_{\inf}$.

6.4.6 Theorem. For $\xi \in \Lambda^\infty(\Lambda)$ and a directed set $D \subseteq \Lambda^\infty(\Lambda)$, $\xi = \bigcup_D$ if and only if $C(\xi) = \bigcup\{C(\zeta) \mid \zeta \in D\}$ in $C_{\text{inf}}(C_{\text{fin}})$.

Proof. For the case of $D \subseteq \Lambda$, we proved in Theorem 4.4.4. Since the $D_\infty$-value of each element of $\Lambda^\infty$ is defined as the limit of a directed sequence of members of $\Lambda$, this result is extended to the case of $D \subseteq \Lambda^\infty$ in a straightforward manner. \(\square\)

It would be interesting to ask if we could remove the condition of directedness from Theorems 4.4.4 and 6.4.6. The answer is 'no' by the following argument.

6.4.7 Definition. 1) For $a, b \in C_{\text{inf}}$, we say $a$ and $b$ are compatible if there is no $\delta \in \Delta$ such that

$$a(\delta) \neq \omega \quad \text{and} \quad b(\delta) \neq \omega$$

2) For $S \subseteq C_{\text{inf}}$, $S$ is said to be compatible if any two elements of $S$ are compatible.

6.4.8 Corollary. For $D \subseteq C_{\text{inf}}$, if $D$ is directed then $D$ is compatible. \(\square\)

By Theorem 4.3.2 and Theorem 6.3.7-3, if $C(\xi) = \bigcup\{C(\zeta) \mid \zeta \in S\}$ for $\xi \in \Lambda^\infty$ and $S \subseteq \Lambda^\infty$ such that $\{C(\zeta) \mid \zeta \in S\} \subseteq C_{\text{inf}}$ is compatible, then $\bigcup_S \subseteq \xi$. However it is not always the case that $\bigcup_S = \xi$.\(\text{D}_{\infty}\)
6.4.9 Counterexample. Let

\[ X = \lambda z x y . z \Omega y \]
\[ Y = \lambda z x y . z x \Omega \]
\[ Z = \lambda z x y . z x y \]

Obviously, \( X \) and \( Y \) are compatible and \( C(Z) = C(X) \cup C(Y) \), but

\( Z(\lambda a b . b a) \xrightarrow{\beta} \Omega \)
\( X(\lambda a b . b a) \xrightarrow{\beta} \Omega \)
\( Y(\lambda a b . b a) \xrightarrow{\beta} \Omega \)

so

\[ X \cup Y \subsetneq Z \]

We note that \( C_{\text{fin}} \) and \( C_{\text{inf}} \) have the least element \( \tilde{\Omega} \) which satisfies

\[ \tilde{\Omega}(\delta) = \omega \quad \text{for all} \quad \delta \in \Delta \]

6.4.10 Proposition. For all elements \( a \in C_{\text{inf}} (C_{\text{fin}}) \) except \( \tilde{\Omega} \), there exists an interesting directed set \( D \subseteq C_{\text{inf}} (C_{\text{fin}}) \) whose least upper bound is \( a \), i.e. \( x \in a \) for all \( x \in D \) and \( a = \cup D \).

Proof. Similar to Theorem 4.4.6. \( \square \)

6.4.11 Proposition. Given \( a, b \in C_{\text{inf}} (C_{\text{fin}}) \) such that \( a \nsubseteq b \), then there exists \( c \in C_{\text{inf}} (C_{\text{fin}}) \) such that

\[ a \nsubseteq c \nsubseteq b \]
Proof. Similar to Theorem 4.4.7. □

Here we state the negative result that $C_{\text{inf}}$ is not continuous.* The following example shows that $C_{\text{inf}}$ is not continuous.

Let

\[ a_n = C(\lambda x y z. z (F^n(\Omega)y)x \]

\[ b = C(\lambda x y z. z y \Omega) \]

\[ c = C(\lambda x y z. z y x) \]

where $F = \lambda f x y.x(fy)$. Since $I = J = \cup_{n=0}^{\infty} F^n(\Omega)$,

\[ c = \cup_{n=0}^{\infty} a_n. \]

Also, $b \subseteq c$, but it is not the case that $b \subseteq a_n$ for any $n \geq 1$.
(The topological order, $\prec$, on $C_{\text{inf}}$ is trivial in a sense that $a \prec b$ for $a, b \in C_{\text{inf}}$ if and only if $a = \bot$.)

6.4.12 Proposition. Given $a, b \in C_{\text{inf}}(C_{\text{fin}})$, we define $a(b)$, application of $a$ to $b$, as follows:

\[ a(b) = C_{\infty}(\xi(\zeta)) \]

where $\xi \in C_{\infty}^{-1}(a)$ and $\zeta \in C_{\infty}^{-1}(b)$. Then $\phi_a : C_{\text{inf}} \to C_{\text{inf}}$

($: C_{\text{fin}} \to C_{\text{fin}}$) for each $a \in C_{\text{inf}}(C_{\text{fin}})$, defined by $\phi_a(b) = a(b)$,

*This fact was suggested with the example to the author by Christopher Wadsworth.
is continuous and $C_{\text{inf}}(C_{\text{fin}})$ is extensional with respect to
the application, i.e. given any $a, b \in C_{\text{inf}}(C_{\text{fin}})$ if
$a(c) = b(c)$ for all $c \in C_{\text{inf}}(C_{\text{fin}})$ then $a = b$.

**Proof.** The continuity of $\phi_a$ is immediate from the definition and the fact that the application of $\lambda$-expressions is continuous in $D_\omega$. The extensionality can be shown in a similar way to Theorem 4.4.8. \qed

We now note that $C_{\text{inf}}$ can become a complete lattice by adding $T$ (top) to $C_{\text{inf}}$. Namely, we define $a \cup b \in \{T\} \cup C_{\text{inf}}$ for $a, b \in \{T\} \cup C_{\text{inf}}$ as follows:

1) If $a = T$, $b = T$ or $a$ and $b$ are not compatible then $a \cup b = T$.

2) If $a$ and $b$ are compatible, determine $c = a \cup b$ by:

$$c(\delta) = \begin{cases} a(\delta) & \text{if } b(\delta) = \omega \\ b(\delta) & \text{if } a(\delta) = \omega \\ \omega & \text{if } a(\delta) = b(\delta) = \omega \end{cases}$$

**6.4.13 Corollary.** $\{T\} \cup C_{\text{inf}}$ is a complete lattice. \qed

However, Counterexample 6.4.9 shows that $\cup S$ reflects the reality only if $S \subseteq C_{\text{inf}}$ is directed. Also, the definition of $\cup$ above is artificially too strong. For example, take $\lambda x.x$ and $\lambda x.xx$. Since they are not compatible, by the definition above $\lambda x.x \cup \lambda x.xx = T$, but since

$$\{\forall (\lambda x.xx) \cup \forall (\lambda x.x) (\forall (\lambda x.xx)) = \forall (\lambda x.x) \} \cup \forall (\lambda x.xx)(\lambda x.xx))$$

$$= \forall (\lambda x.xx) \neq T$$
So \( W \llbracket \lambda x.xx \rrbracket \cup W \llbracket \lambda x.x \rrbracket \neq T \). However, can any interesting theory be built if we allow \( \cup \) of mutual elements such as \( \lambda x.xx \) and \( \lambda x.x \) not be \( T \)? I leave this question open here.
CHAPTER 7
SUMMARY, CONCLUSIONS, PROSPECTS

Summarizing the results obtained in the previous chapters, we show that only some abstract properties of $D_\infty$ are needed to deduce these results. This fact makes it possible to give an axiomatization of the extensional model theory of $\lambda$-calculus. Some prospects for future research are given.
§1. Summary

We start this chapter with the summary of the results in Chapters 3, 4 and 6. The following diagram is illustrative:

7.1.1 Diagram.

\[
\begin{array}{ccc}
\Lambda & \subseteq & \Lambda^\infty \\
\downarrow W & & \downarrow W^\infty \\
C & \iff & C^\infty \\
\downarrow d & & \downarrow d^\infty \\
C_{\text{fin}} & \subseteq & C_{\text{inf}}
\end{array}
\]

where \(d^\infty\) and \(d\) are defined as follows:

For \(c \in C_{\text{inf}}, \rho \in EN,\)
\[d^\infty [c] \rho = V_{\infty} [\xi] \rho \text{ for } \xi \in \Lambda^\infty \text{ such that } c = C_{\infty}(\xi).
\]

For \(c \in C_{\text{fin}}, \rho \in EN,\)
\[d [c] \rho = V [z] \rho \text{ for } z \in \Lambda \text{ such that } C(z) = c.
\]

These definitions of \(d\) and \(d^\infty\) are valid, since, by Theorems 4.3.2 and 6.3.7-3,
\[
\xi = \eta \text{ iff } C_{\infty}(\xi) = C_{\infty}(\eta) \text{ at } D^\infty.
\]

We list the results we have reached:
1) \(C\) and \(C^\infty\) are surjective.
2) \(d\) and \(d^\infty\) are injective.
3) \(C = C^\infty|\Lambda\) and \(d = d^\infty|C_{\text{fin}}\).
4) The diagram is commutative, i.e. \(W = d \circ C\) and \(W^\infty = d^\infty \circ C^\infty\).
5) \(d^\infty: C_{\text{inf}} \rightarrow [EN \rightarrow D^\infty]\) is a continuous map.
   (So \(d^\infty\) is monotonic, too.)
Let us prove 5). We regard \([EN \rightarrow D_\infty]\) as a lattice by the partial order induced by \(D_\infty\), i.e., given \(\alpha, \beta \in [EN \rightarrow D_\infty]\), \(\alpha \preceq \beta\) if and only if \(\alpha(p) \subseteq \beta(p)\) for all \(p \in EN\). Given any directed set \(\mathcal{D} \subseteq \mathcal{C}_\text{inf}\), let \(\bar{\mathcal{D}} = \{\xi \in \Lambda^\infty \mid C_\omega(\xi) \in \mathcal{D}\}\). Then \(\bar{\mathcal{D}}\) is directed in \(D_\infty\). Let \(\eta = \bigcup \mathcal{D} \in \Lambda^\infty\). By Theorem 6.4.6, \(C_\omega(\eta) = \bigcup \{C_\omega(\xi) \mid \xi \in \bar{\mathcal{D}}\}\). Now, given any \(p \in EN\),

\[
d_\omega [\bigcup \mathcal{D}] \rho = d_\omega [\bigcup \{C_\omega(\eta)\}] \rho
\]

\[
= \bigvee_\omega [\bigcup \mathcal{D}] \rho \quad \text{by the definition of } d_\omega
\]

\[
= \bigvee_\omega [\bigcup \bar{\mathcal{D}}] \rho
\]

\[
= \bigcup \{\bigvee_\omega [\xi] \rho \mid \xi \in \bar{\mathcal{D}}\}
\]

\[
= \bigcup \{d_\omega [\xi] \rho \mid \xi \in \bar{\mathcal{D}}\} \quad \text{by the definition of } d_\omega.
\]

This proves that \(d_\omega\) is continuous. \(\Box\)

Since \(d_\omega\) is 1 to 1, continuous, we can say that \(\mathcal{C}_\text{inf}\) give the lattice structure of \(\Lambda\) and \(\Lambda^\infty\) induced by \(D_\infty\). Since \(\mathcal{C}_\text{inf}\) can be defined naturally from \(\Lambda\), independent of \(D_\infty\), the lattice structure of \(\mathcal{C}_\text{inf}\) can be said to be the inherent structure of \(\Lambda\).
§2. **Universality of $\mathcal{C}_{\inf}$ over $\Lambda$**

We may ask what properties of $D_\infty$ are essential to have the theory summarized in the previous section. Namely, for what kind of model of the $\lambda$-calculus, could we draw Diagram 7.1.1 such that the properties 1) to 5) may hold? This speculation will probably lead us to a more general theory of $\lambda$-calculus models.

To make the description of this section as self-contained as possible, we start with the definitions of the $\Delta$-trees and their admissibility in Chapter 4.

7.2.1 **Definition.** An infinite subset $\mathcal{T}$ of the pedigree $\Delta$ is said to be a $\Delta$-tree if

1) $0 \in \Delta$

2) If $\delta \in \mathcal{T}$, then there exists $N \in \mathbb{N}$ such that $\delta^0, \delta^1, \ldots, \delta^N \in \mathcal{T}$ and $\delta^k \notin \mathcal{T}$ for all $k > N$.

For a $\Delta$-tree $\mathcal{T}$ and $\delta \in \mathcal{T}$, define $\gamma_{\mathcal{T}}(\delta)$ to be $N$ in (2), i.e.

$$\gamma_{\mathcal{T}}(\delta) = \#\{\delta' \in \mathcal{T} | \delta' = \delta^m \text{ for some } m \in \mathbb{N}\}.$$

We redefine the notion of admissibility as follows:

7.2.2 **Definition.** Given a $\Delta$-tree $\mathcal{T}$ and $c \in \mathcal{C}_{\inf}$, $\mathcal{T}$ is said to be admissible to $c$ if, for all $\delta \in \mathcal{T}$, $\gamma_{\mathcal{T}}(\delta) \geq N^C_\delta$ where $N^C_\delta$ is as in Condition 3 of $\mathcal{C}_{\inf}$.

7.2.3 **Definition (Structural Approximation).** Given $c \in \mathcal{C}_{\inf}$ and a $\Delta$-tree $\mathcal{T}$ which is admissible to $c$, we define $A^n_p(c, \mathcal{T}) \in \Lambda^p$ in the following way: $A^n_p(c, \mathcal{T}) = A^0_p(c, \mathcal{T}, n)$ where $A^\delta(c, \mathcal{T}, n)$ is defined for each $\delta \in \mathcal{T}$ inductively as:
1) If $|\delta| < n$,
   (i) if $c(\delta) = \omega$ then $A^\delta(c,T,n) = \Omega$
   (ii) if $c(\delta) = z$, then
   \[
   A^\delta(c,T,n) = \lambda t_0 t_1 t_2 \cdots t_{\delta_0(\gamma_T(\delta)+k_0^C)} A^\delta_1(c,T,n) A^\delta_2(c,T,n) \cdots A^\delta_\gamma_T(\delta)(c,T,n)
   \]

where $k_0^C$ is as in Condition 3 of $C_{inf}$.

2) If $|\delta| = n$, then $A^\delta(c,T,n) = \Box$.

Lemma 4.2.2 is rewritten as follows:

7.2.4 Lemma. Given $x \in A$ and a $\Delta$-tree $T$, if $T$ is admissible to $C(x)$, then

\[
A^\infty_x = \bigcup_{n=1}^{\infty} (A^n_p(C(x),T))^*
\]

where $*: A \rightarrow A$ is as in Definition 6.3.1.

Obviously $A^{n+1}_p(c,T) \supseteq A^n_p(c,T)$ for $c \in C_{inf}$ and a $\Delta$-tree $T$ admissible to $c$. So $\xi_c = (A^1_p(c,T),A^2_p(c,T),\ldots)$ is in $A^\infty$. This $\xi_c$ gives a decoding of $c$ in $A^\infty$, i.e. $c = C_\infty(\xi_c)$.

7.2.5 Definition. We say that a domain $D$ is a reasonable extensional model for the $\lambda$-calculus if $D$ satisfies the following conditions:

Axiom 1. $D$ is a directed-complete partially ordered set with the least element $\bot = \cap D$ and $D \not= \{\bot\}$. 
**Axiom 2.** There is the following pair of maps \((\phi, \psi)\) that are bijective and continuous

\[
\begin{array}{ccc}
D & \xrightarrow{\phi} & [D \times D] \\
\psi & \downarrow & \\
D & \xleftarrow{\psi} & [D \times D]
\end{array}
\]

**Axiom 3.** We map \(\Lambda\) into \(D\) by the semantic function \(W\) as follows: Let \(EN = (U \times D)\) for the set of variables \(U\).

\(W: \Lambda \to [EN \times D]\) is defined as:

1. For \(v \in U\) and \(\rho \in EN\), \(W[v] \rho = \rho(v)\).
2. For \(x, y \in \Lambda\), \(\rho \in EN\), \(W[x(y)] \rho = \phi(W[x] \rho) (W[y] \rho)\).
3. For \(v \in U\), \(x \in \Lambda\) and \(\rho \in EN\),

\[
W[\lambda v. x] \rho = \psi(\lambda \beta \in D: W[x] \rho[v/\beta])
\]

where

\[
\rho[v/\beta](u) = \begin{cases} 
\rho(u) & \text{if } u \neq v \\
\beta & \text{if } u = v
\end{cases}
\]

Then the following two properties hold:

a) For each \(x \in \Lambda\), \(W[x] \rho = \bot\) for all \(\rho \in EN\) if \(x\) has no head normal form.

b) For each \(x \in \Lambda\) and a \(\Delta\)-tree \(T\) which is admissible to \(C(x)\),

\[
W[x] \rho = \cup \{W[A^n_p(C(x), T)] \rho | n \in \mathbb{N}\}
\]

for each \(\rho \in EN\).

**7.2.6 Theorem (Universality of \(C_{\inf}\) over \(\Lambda\)).** If \(D\) is any reasonable extensional model for \(\Lambda\), then Diagram 7.1.1 is valid even if we replace \(D_\infty\) by \(D\).
Proof. This theorem asserts that all the theory developed in Chapters 4, 5 and 6 depend on only the properties of \( D \) in Definition 7.2.5. The proof is done by a careful inspection on what properties of \( D_\infty \) are used to prove the validity of the diagram. \( \Box \)

Here we note that Wadsworth's theorem on reduced approximants also holds on \( D \) as defined in 7.2.5.

7.2.7 Proposition. Let \( A(x) \) be the set of all reduced approximants of \( x \in A \). Then in a reasonable extensional model \( D \),

\[
x = \bigcup_{D} A(x)
\]

Proof. Let \( T \) be a \( \Delta \)-tree admissible to \( C(x) \). We show that given \( A^n_p(C(x), T) \) for any \( n \in \mathbb{N} \) there is \( \varepsilon \in A(x) \) such that

\[
A^n_p(C(x), T) \subseteq \varepsilon
\]

i) If \( x \) has no head normal form, then \( x = \bot \) and \( A(x) = \{ \Omega \} \), so \( x = \bigcup_{D} A(x) \).

ii) If \( x \) has a head normal form, let us consider

\[ A^n_p(C(x), T) \]. By Corollary 4.1.13, there is \( T^n(x) \) such that

\[
x \xrightarrow{\text{CNV}} T^n(x)
\]

and \( A^n_p(C(x, T)) \) is obtained from \( T^n(x) \) by replacing each \( L(x, \delta) \) in \( T^n(x) \) by \( \Omega \) for each \( \delta \in T \) such that \( |\delta| = n \).

Let \( x \xrightarrow{\beta} x' \xrightarrow{\text{CNV}} T^n(x) \) be the sequence of reductions so that
x' \rightarrow T^n(x) does not contain any \( \beta \)-reductions. Since \( \alpha, \eta, \Omega \)-conversions do not increase \( \beta \)-redexes, this resolution is possible. Then let \( \varepsilon \) be the direct approximant of \( x' \). Then

\[ A^n_p(C(x), T) \subseteq \varepsilon \]

because all the \( \beta \)-redexes in \( x' \) are in \( T^n(x) \) and they are in the parts of \( T^n(x) \) which have no corresponding part in \( A^n_p(C(x), T) \) except \( \Omega \). Since \( x = \bigcup_{D(n=0)} A^n_p(C(x), T) \), \( x = UA(x) \).

However Axiom 3b cannot be deduced from Axiom 3a or \( x = UA(x) \) or the combination of both. One may ask if we could reduce Axiom 3b to a simpler condition. This does not seem to be possible if we consider Park's pathological model [10]. Park showed that if a different \( (\phi, \psi) \) is adopted to construct \( D_\infty \), then \( Y \neq \lambda f. \cup_{D_\infty} f^n(\bot) \) for the Curry's pathological combinator \( Y \).

As Corollary 4.2.3, Axiom 3b implies \( Y = \lambda f. \cup_{D_\infty} f^n(\bot) \). This implies that Axiom 3b is not true in Park's model. This indicates that this axiom cannot be deduced from such a simpler condition as the continuity of \( D_\infty \). (We note that the proof of Axiom 3b in \( D_\infty \) depends on the type construction of \( D_\infty \).) Probably, Axiom 3b must be proved for each model \( D \) directly from its construction.

By Proposition 7.2.7, we can conclude that, if \( D \) satisfies Axioms 1-3, all the results on \( D_\infty \) vs. \( \Lambda \) due to Wadsworth [21,22] are valid on \( D \) vs. \( \Lambda \). (So, \( I = J \), \( Y = \lambda f. \cup_{D_\infty} f^n(\bot) \), etc.)
§3. Prospects

We have completely ignored the other models of the $\lambda$-calculus. $E_\infty$ in [21] and $P_\omega$ in [17] are examples of non-extensional models, on which the formulation of the infinite normal form developed in this thesis is no longer valid.

It would be possible to formulate the infinite normal forms on these non-extensional models. However, it does not seem possible to give such a clean theory as is possible on $D_\infty$. Many interesting algebraic properties of $C_{\inf}$ are possible due to the extensionality of $D_\infty$.

Another point to note is the problems of the compatibility and inconsistency among the $\lambda$-expressions. We say that two $\lambda$-expressions $x, y$ are inconsistent if there exists $\delta \in \Delta$ such that $C(x, \delta) \neq \omega$, $C(y, \delta) \neq \omega$ and $C(x, \delta) \neq C(y, \delta)$. Some of the problems caused by the introduction of $\cup$ into the $\lambda$-calculus were discussed in §4, Chapter 6.

I am not sure at this point whether or not we could develop an interesting theory by introducing the $\cup$ operation into the $\lambda$-calculus.
Lastly in this section, we focus our attention on $\Lambda^\infty$. One may ask what $\Lambda^\infty$ is in actuality. We can say that $\Lambda^\infty$ is arbitrary if we confine ourselves to the function over the integers. For example, if we consider the flat space of:

$$0, 1, 2, 3, ...$$

$N = \cdots \downarrow$

$\Lambda^\infty$ gives all continuous functions from $N$ to $N$. * Let us consider the following sequence of mappings over $N$. Let

\[ f_0: N \to N \text{ be } f(n) = \begin{cases} a_0 & \text{if } n = 0 \\ \bot & \text{if } n > 0 \end{cases} \]

\[ f_1: N \to N \text{ be } f(n) = \begin{cases} a_0 & \text{if } n = 0 \\ a_1 & \text{if } n = 1 \\ \bot & \text{if } n > 0 \end{cases} \]

\[ \vdots \]

\[ f_p: N \to N \text{ be } f(n) = \begin{cases} a_0 & \text{if } n = 0 \\ a_1 & \text{if } n = 1 \\ \vdots & \text{if } n = i \leq p \\ \bot & \text{if } n > p \end{cases} \]

\[ \vdots \]

Then $f_0 \subseteq f_1 \subseteq \cdots \subseteq f_n \subseteq \cdots$. Since the choice of $a_i$ is arbitrary, $f = \bigcup_{i=0}^{\infty} f_i$ can be the arbitrary continuous function $N \to N$.

This shows that to give the smooth property to $\Lambda$, we must

---

*This fact was suggested to the author by Manuel Blum.*
inevitably include a rather non-computable structure. This is similar to extending the rational number to the real number. The real numbers contain all the arbitrary transcendental numbers, but we cannot discuss any problem in the elementary calculus excluding these objects. Another more hopeful view is that $\Lambda^\infty$ may give a certain significant proper subset of the continuous functions if the domain's lattice structure is more complex and has the continuous cardinality. For example, $R \equiv \{[a,b] | a \leq b \text{ are real}\} \cup \{\emptyset\}$ with the partial order $\alpha < \beta$ if $\beta \leq \alpha$ and $\bot = \emptyset \in R$. (If $\text{LAMDA}^\infty$ is generated from $\text{LAMDA} [17]$ in the same manner as $\Lambda^\infty$ is generated from $\Lambda$, it will probably be the case that $P_\omega = \text{LAMDA}^\infty$, for any recursively enumerable set is in $\text{LAMDA}$ and any set of integers is the limit of an increasing sequence of recursively enumerable sets.)
REFERENCES


