STANFORD ARTIFICIAL INTELLIGENCE PROJECT
MEMO AIM-168

STAN-CS-72-287

ADMISSIBILITY OF FIXED-POINT INDUCTION IN FIRST-ORDER LOGIC OF TYPED THEORIES

BY

SHIGERU IGARASHI

SUPPORTED BY
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
AND
ADVANCED RESEARCH PROJECTS AGENCY
ARPA ORDER NO. 457

MAY 1972

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

STANFORD ARTIFICIAL INTELLIGENCE PROJECT MAY 1972
MEMO AIM-168

COMPUTER SCIENCE DEPARTMENT
REPORT CS-287

ADMISSIBILITY OF FIXED-POINT INDUCTION IN FIRST-ORDER LOGIC OF TYPED THEORIES

by

Shigeru Igarashi*)

ABSTRACT: First-order logic is extended so as to deal with typed theories, especially that of continuous functions with fixed-point induction formalized by D. Scott. The translation of his formal system, or the λ calculus-oriented system derived and implemented by R. Milner, into this logic amounts to adding predicate calculus features to them.

In such a logic the fixed-point induction axioms are no longer valid, in general, so that we characterize formulas for which Scott-type induction is applicable, in terms of syntax which can be checked by machines automatically.

To be presented at the Symposium on Theoretical Programming, Novosibirsk, August 1972.

This research was supported in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense under Contract SD-183 and in part by the National Aeronautics and Space Administration under Contract NSR 05-020-500.

*) Address after 1 July 1972: Research Institute for Mathematical Sciences, Kyoto University, Sakyoku, Kyoto 606, Japan.

C O N T E N T S

1. Introduction 1

2. First-Order Logic of Typed Theories 2

2.1 Language 2
2.2 Interpretation 4
2.3 Truth functions associated with formulas 7
2.4 Logical axioms and rules 8

3. Weakly Continuous Functions 11

4. Admissibility of Fixed-Point Induction 15

5. Characterization of Predicates that admit Fixed-point Induction 21

6. Syntax of Formulas that admit Induction 29

6.1 Tables of inheritance of admissibility 29
6.2 Example of formula that admits induction 31

7. Translation of LCF into First-Order Logic of Typed Theories 32

7.1 Axiomatization 32
(Table of nonlogical axioms) 33
7.2 Adequacy 34
7.3 Example taken from proof of compiler correctness 35

Discussions 39

Admissibility of Fixed-Point Induction in First-Order Logic Of Typed Theories

by

Shigeru Igarashi

1 Introduction

D. Scott postulated a logic of typed functions combined with fixed-point induction[10]. R. Milner modified this logic into a formal system called LCF so as to handle λ-expressions conveniently, and implemented it in an interactive proof checker[6]. Since an early period of this implementation it has been thought that some predicate calculus-like facility may be needed for some or other reasons, so that in the machine version of LCF are included a kind of universal quantifier and implication, the latter being one level lower than the implication included in the original logic. These operators, however, can be used in quite a restricted manner, for they are only abbreviations of legitimate formulas in LCF. Especially implication cannot be nested.

The writer devised a formal means to carry out derivations of a predicate calculus whose objects were typed λ-expressions within LCF, which calculus included the universal quantifier as well as usual propositional operators but not the existential quantifier, which could not be replaced by negation and universal quantification since Gentzen's intuitionistic system was used as the basis. J. McCarthy[4] proposed to use the full classical predicate calculus as a super-structure of LCF, quantifiers ranging over LCF objects. He suggested also some generalization of such a system. The formal system discussed in the present paper is in the essentials along the last line. The main purpose of the present paper is to allow Scott-type fixed-point induction as much as possible in the intended logic.

This point will be explained more concretely. Suppose f and g are continuous partial functions. The predicate f≡g, where the equality means the "strong equality", i.e., if one side is undefined so is the other, is not continuous. But as in Scott's logic we can use fixed-point induction in order to prove this equality. Then what will happen to the following formula which we are going to allow in the intended logic?

$$\forall x(f(x)\equiv a \supset g(x)\equiv h(x)),$$

with the axiom

$f \equiv \mathrm{Min}\ \lambda f \lambda x J(f,x)$,

⇒ being implication in the classical sense, Min the minimal fixed-point of the function to which it is prefixed, and J(f,x) a term in LCF. It turns out that if all the functions involved in the expression J(f,x) are continuous, which condition is rather natural in order to consider its fixed-point, and the range of f is discrete, like a boolean function, then we can apply fixed-point induction without incurring inconsistency, even if g and h are non-continuous functions. In fact the continuity of g and h does not matter in this case, for fixed-point induction is not sound unless the above conditions are satisfied.

We shall give a syntactic characterization of the formulas for which fixed-point induction is sound, so that machines can check automatically whether or not a given formula admits application of the inference rule corresponding to fixed-point induction.

2 First-Order Logic of Typed Theories

We consider a kind of infinitely many-sorted first-order logic in the classical sense[12]. The objects are individuals in the usual sense together with functions of individuals or previously defined functions. Each type can be regarded as a sort. Only objects are typed, and we do not consider predicate variables. The intended formal system will be abbreviated as FLT. We shall partially follow Shoenfield's style[11].

2.1 Language

Types

A1. We presuppose that there are a number of types called the "base types". Some of the base types can be "ordered types". Types are denoted by α, β, etc., and the ordered types are postfixed by the letter "o", like αo, No relationships between α and αo are assumed if both α and αo happen to be base types. Types other than the base types are called the "function types".

A2. If α and β are types, so is $\alpha \to \beta$. Both $\alpha 1 \to \alpha 2 \to \ldots \to \alpha n \to \beta$ and $\alpha 1, \alpha 2, \ldots, \alpha n \to \beta$ are used as the abbreviations of $\alpha 1 \to (\alpha 2 \to (\ldots \to (\alpha n \to \beta), \ldots))$.

A3. If αo and βo are types, which must be ordered types, so is $(\alpha o \to \beta o)o$.

Because of this construction we can consistently abbreviate "o"s except the outmost one. For instance, $(\alpha o \to (\beta o \to (\beta o \to \beta o)o)o)o$ is abbreviated by $(\alpha \to \beta \to \beta \to \beta)o$.

Alphabet

The alphabet of the intended formal system consists of α-constants and α-variables for each type α, $(\alpha 1, \ldots, \alpha n)$-predicates, i.e., predicate constants, for each n-tuple $(\alpha 1, \ldots, \alpha n)$ of types $(n \geq 0)$, and the following logical symbols.

$= \; (\; , \;) \; \neg \; \forall \; \exists \; \mathrm{Min}$

If α is a base type, an α-constant or variable can be called an individual constant or variable. Otherwise, an α-constant or variable can be called a function constant or variable. It must be noted that functions of arbitrary finite order appear. An $(\alpha 1, \ldots, \alpha n)$-predicate is an n argument predicate in the usual sense, the i-th argument being of type αi for each i $(1 \leq i \leq n)$.

We shall use several defined symbols which are standard in logic as follows.

$\& \; \to \; \vee \; \equiv$

The symbol $\to$ stands for implication, and $\equiv$ for logical equivalence. Thus $\to$ means function in the text and implication in formulas.

Terms

B1. If a is an α-constant, then a is an α-term. If x is an α-variable, then x is an α-term.

B2. If t is an $\alpha \to \beta$-term and u is an α-term, then t(u) is a β-term. t(u) can be also written as (t u), and (t(u))(v) as t(u,v).

B3. If t is an $(\alpha o \to \alpha o)o$-term, then Min t is an αo-term.

B4. If t is an αo-term and αo is a function type, then t is an α-term.

Formulas

C1. If t and u are α-terms, then $t=u$ is a formula.

C2. If p is an $(\alpha 1, \ldots, \alpha n)$-predicate that is different from $=$, and ti is an αi-term for each i $(1 \leq i \leq n)$, then $p(t1, \ldots, tn)$ is a formula.

C3. If A is a formula, then $\neg A$ is a formula.

C4. If A and B are formulas, then $A \vee B$, $A \& B$, and $A \to B$ are formulas.

C5. If A is a formula and x is an α-variable, then $\forall x A$ and $\exists x A$ are formulas.

2.2 Interpretation

We choose a non-empty set $D[\alpha]$, or $D\alpha$, for each base type α as the domain of individuals of type α. If α is an ordered base type, we assume further that $D\alpha$ is an ordered set $(L, \leq)$ satisfying the following conditions.

(i) $(L, \leq)$ has the least element, i.e. inf L, which shall be denoted by C.

(ii) $(L, \leq)$ is an ω-inductively ordered set in that L is non-empty and every non-empty countable set X such that $X \subseteq L$ and X is linearly ordered has sup X in L.

That L is non-empty is a part of the standard definition of the inductively ordered set, which is automatically satisfied in this case. The symbol "ω" reads "omega" through out this paper. In some case, it can be read "aleph naught".

Suppose $D\alpha$ and $D\beta$ have been defined. We let $D[\alpha \to \beta]$ be the set of all the functions of $D\alpha$ into $D\beta$. If α and β are ordered type, we let $D[(\alpha \to \beta)o]$ be the set of all the ω-continuous functions belonging to $D[\alpha \to \beta]$ together with the order relation $\leq$ defined by

$$f \leq g \text{ iff } f(x) \leq g(x) \text{ for any } x \in D\alpha,$$

where the ω-continuity is defined as follows.

Definition. A "sequence" X in a set L is a function of the set of the positive integers into L, Xn denoting the n-th term X(n). X is written as (Xn) sometimes. A "monotone increasing" sequence X in (L, ≤) is a sequence in (L, ≤) such that

$$X1 \le X2 \le \ldots \le Xn \le \ldots \quad .$$

f is "ω-continuous" iff

$$f(\sup X) = \sup f(X),$$

for any monotone increasing sequence X in (L, ≤), where f(X) denotes the set {f(x)|x∈X}.

Remark. f is ω-continuous in this sense iff f(sup X) = sup f(X) for any countable directed set X⊂L. (See section 3.) This property will be called the ω-continuity, while a stronger definition of continuity is that f(sup X) = sup f(X) for any directed set X⊂L. f is said to be "monotone" iff f(x)≤f(y) whenever x≤y. The ω-continuity implies the monotonicity, which can be shown as follows[18].

Suppose x≤y. Let X1 be x and Xn be y for any n≥2, so that X is a monotone increasing sequence. By ω-continuity, f(sup X) = sup f(X). But sup X = y, and f(x) ≤ sup f(X). Therefore f(x) ≤ f(y).

By this construction Dαo can be shown to satisfy the conditions (i) and (ii), so that the inductive definition works. In fact, the function g: Dαo→Dβo such that

$$g(x) = 0 \quad \text{for any } x \in D\alpha o$$

is the least element of D[(αo→βo)o], and, for each asending chain (fn) in D[(αo→βo)o], the function h: Dαo→Dβo that maps each element x of Dαo onto sup(fn(x)) is sup(fn).

With each α-constant a in FLT is associated an element a* of Dα. With each (α1, ..., ,αn)-predicate p in FLT is associated an n-ary relation p* in Dα1× ... ×Dαn. Such a collection of Dα's will be denoted by D, and FLT(D) will denote the language obtained from FLT by adding a new α-constant, called a "name", for each element of Dα, for each α.

A term is "closed" if no variables occur free in it. Especially, a variable-free term is closed in this sense. We use this terminology becaouse we shall extend the syntax of terms later in order to axiomatize LCF, in which λxx is a closed term, though it is not variable-free. We define an α-individual *t for each closed α-term t by induction on terms.

D1. If t is an individual symbol, then t must be an α-constant since t is closed. We let *t be a*∈Dα.

D2. If t is u(v), then u must be a closed α→β-term and v a closed α-term, so that #u∈D[α→β] and #v∈Dα. We let #(u(v)) be #u(#v).

D3. If t is Min u, then u must be a closed (ασ→ασ)σ-term, so that #u is an ω-continuous function of type ασ→ασ. Let f denote #u. We let #t be inf(x|f(x)≤x) (with respect to the ordering of ασ), namely the least fixed point of f, which is shown to exist as follows[10].

Let f.n.x denote

f(f(,.., f(x),..,)) (f occurs n times),

for each n≥0. Especially, f.0.x is x. Then sup(f.n.0), or sup(f.n.0|0≤n<ω) strictly, is in fact inf(x|f(x)≤x). By ω-continuity,

$$
\begin{aligned}
f(sup(f.n.0)) &= sup(f(f.n.0)) \\
&= sup(f.(n+1).0) \\
&= sup(f.n.0|1\le n<\omega) \\
&\le sup(f.n.0),
\end{aligned}
$$

By monotonicity (see the above remark),

$$sup(f.n.0) \le f(sup(f.n.0)).$$

Thus

$$f(sup(f.n.0)) = sup(f.n.0),$$

Namely sup(f.n.0) is a fixed point of f. Let a be an element of Dασ such that f(a)≤a. Since 0≤a, f(0)≤f(a)≤a, by monotonicity. Then, by mathematical induction, f.n.0≤a for any n, so that sup(f.n.0)≤a. Thus sup(f.n.0)=inf(x|f(x)≤x).

D4. If t is a closed ασ-term and ασ is not a base type, then #t∈Dασ and Dασ⊂Dα, so that #t∈Dα.

A truth value is either T or F. T means "true" and F "false".

A formula is "closed" if no variables occur free in it. We define a truth value #A for each closed formula A in FLT(D) by induction on formulas. A[ ], or t[ ], denotes a formula, or a term, with voids, and A[x], or t[x], results of replacing them by x.

E1. If A is t=u, then t and u must be closed α-terms for a certain α, since A is closed. We let

#A=T iff #t=#u.

E2. If A is p(t1, ... ,tn) where p is different from =, we let

#A=T iff p#(t1, ... ,tn).

E3. If A is ¬B, then we let

*A=T iff *B=F.

E4. If A is BvC, then we let

*A=T iff *B=T or *C=T.

E5. If A is ∃xB[x] and x is an σ-variable, then B[a] is closed for each σ-name a. We let

*A=T iff *(B[a])=T for some σ-name a.

A "D-instance" of a formula A[x1, ... ,xn] of FLT is a closed formula of the form A[a1, ... ,an] in FLT(D), where ai is an σi-name if xi is an σi-variable (1≤i≤n). A formula A of FLT is "valid" in D if *A'=T for every D-instance A' of A. In particular, a closed formula A of FLT is valid iff *A=T.

2.3 Truth functions associated with formulas

To study the properties of formulas we shall consider truth functions, namely functions whose values are the truth values T and F, associated with formulas in the natural manner. For the convenience of the later description we use the following terminologies.

Let x be an σ-variable, and A[x] a formula in which at most x occurs free. Since A[a] is a closed fromula for each σ-name a, we can define a function f: Dσ→{T,F} that sends each a* onto the truth value *A[a]. f is called "the truth function determined by A and x in D", or, if there is no ambiguity, "the truth function determined by A"

Let A[x1, ... ,xn] be a formula in which at most variables x1, ... , xn, respectively of type σ1, ... , σn, occur free. A "(D,xi)-instance" of A[x1, ... ,xn] in FLT is a formula in FLT(D) of the form A[a1, ... ,a(i-1),xi,a(i+1), ... ,an] where a1, ... , an are names of type σ1, ... , σn. Thus at most xi occurs free in formulas that are (D,xi)-instances of a formula (1≤i≤n). Therefore each (D,xi)-instance determines a truth function.

A[x1, ... ,xn] also "determines" an n-ary truth function f: D[σ1]× ... ×D[σn] → {T, F} that sends each n-tuple (a1*, ... ,an*) onto *A[a1, ... ,an].

2.4 Logical axioms and rules

We shall accept the following axioms and rules for FLT.

Rule of substitution. In the below schemata of axioms or rules, arbitrary variables can be substituted in place of a, x, y, z, x1, y1, z1, ... , xn, yn, zn, and w, arbitrary terms in place of t, u, v, and s, an arbitrary n-ary predicate in place of p for each n, and an arbitrary formula in place of A, B, and C, subject to the restrictions that the results of substitutions should be well-formed formulas and that any free occurrence of variables should be kept free. On the induction axiom are imposed the additional restriction that only those formulas of the form A[ ] that "admit induction syntactically" are substituted in place of A[ ]. The effective definition of formulas that admit induction syntactically is given in section 6.1.

Logical axioms

propositional axiom.	$\neg A \vee A$.
identity axiom.	$x=x$.
equality axiom.	$x=y \rightarrow z=w \rightarrow x(z)=y(w)$.
	$x=y \rightarrow \mathrm{Min}\ x = \mathrm{Min}\ y$.
	$x1=y1 \rightarrow \ldots \rightarrow xn=yn \rightarrow p(x1, \ldots, xn) \rightarrow p(y1, \ldots, yn)$.
stationariness axiom.	$x(\mathrm{Min}\ x) = \mathrm{Min}\ x$.
induction axiom.	$A[0] \rightarrow \forall y(A[y] \rightarrow A[x(y)]) \rightarrow A[\mathrm{Min}\ x]$.

Rules of inference. We shall accept all the rules in Gentzen's system of Natural Deduction[1], or NJ, with the following modification of the quantifier-introduction and elimination rules. (a designates a variable in this section.)

$\forall$-introduction rule.

```
A[a]
------ <a>
∀xA[x]
```

$\forall$-elimination rule.

```
∀xA[x]
------
A[t]
```

∃-introduction rule, ∃-elimination rule,

$$\frac{A[t]}{\exists x A[x]} \qquad \frac{\exists x A[x] \quad \begin{matrix}(A[a])\\ C\end{matrix}}{C}\ \langle a\rangle$$

Restriction: In the ∀-elimination rule and the ∃-introduction rule, the eliminated or introduced bound variable, replacing x, must be of the same type as the corresponding term, replacing t. In the ∀-introduction rule and th ∃-elimination rule, the introduced or eliminated bound variable, replacig a, must be of the same type as the corresponding free variable (eigenvariable), replacing a.

⟨a⟩ indicates the restriction, in the original NJ, that the free variable substituted in place of a occurs only in the places explicitly designated by a. Thus, for instance, in the ∀-introduction rule the free variable replacing a must not occur in the formula designated by ∀xA[x], nor in any assumption formula of that formula.

As appears in the above rule we use ( ), in stead of [ ] in the original notation, to indicate the assumption formula which is not carried beyond the bar. Besides, we shall use A --→ B sometimes, as well as ( ), to denote that A is an assumption formula of B, and A1, ..., Am --→ B1, ..., Bn to denote a "sequent", in the sense of Gentzen's LK. For instance, the ∨-elimination rule can be expressed in the following ways, and we shall use all of them in the sequel for the convenience of description.

∨-elimination rule,

$$\frac{A\vee B \quad \begin{matrix}(A)\\ C\end{matrix} \quad \begin{matrix}(B)\\ C\end{matrix}}{C}$$

Infer C from A∨B, A--→C, and B--→C.

Infer P --→ C from P --→ A∨B, A,P --→ C, and B,P --→ C.

An inference rule of the last form, i.e. a rule to infer a sequent from other sequents is called a "relativized" inference rule. A sequent of the form A1, ... ,Am --→ B1, ... ,Bn is "valid in D" iff the formula A1&...&Am → B1∨...∨Bn is valid in D. A relativized inference rule is "sound" iff the consequence of the rule is valid in D (as sequent) whenever all of its premises are valid in D, for any D.

We can treat the logical axioms in the form of inference rules. We list them in the generalized forms for the practical derivation. These rules are derived rules actually.

propositional rule.

$$\frac{}{\neg A \vee A}$$

identity rule.

$$\frac{}{t = t}$$

equality rule.

$$\frac{t = u \qquad A[t]}{A[u]}$$

stationariness rule.

$$\frac{}{t(\mathrm{Min}\ t) = \mathrm{Min}\ t}$$

induction rule.

$$\frac{A[0] \qquad A[a] \to A[t(a)]}{A[\mathrm{Min}_n\ t]} \langle a \rangle$$

<a> indicates the same restriction as described above. Thus the variable substituted in place of a must not occur free in A[Min t], nor in A[0], nor in any assumption formula of A[Min t].

Apparently the induction axiom, or rule, is not acceptable unless some adequate restriction, like the one indicated in the rule of substitution, is imposed on it. First, in order to instantiate this axiom by a name b, substituting b in place of x, ωb must be ω-continuous so that Scott-type fixed point induction makes sense, which restriction is satisfied in the present formalism, for Min b is not a well-formed term otherwise. Second, even if Min b represents an ω-continuous function of an appropriate type, there exist many formulas which make this axiom not valid. The main purpose of this paper is to characterize those formulas for which the induction axiom is valid, so that they admit the application of this rule.

3 Weakly Continuous Functions

The validity of the induction axiom reflects the properties of truth functions associated with formulas of FLT. The first such property will be called the weak continuity. It must be noted that most of the truth functions determined by formulas of FLT are not continuous, and we are going to establish some criteria for such non-continuous predicates to make the induction axiom valid. The weak continuity can be defined for functions, so that we discuss this property in general.

Through out this section, L denotes an ω-inductively ordered set with the least element 0 (see section 2.2), and L' a complete lattice. Namely, L' is an ordered set such that inf X and sup X exist for any subset X of L'. 0 and I shall denote the least element of L', or inf L', and the greatest element of L', or sup L', respectively.

Let X be a sequence in L'. We consider the monotone increasing sequence Y defined by $Yn = \inf(Xm | m \geq n)$, and the monotone decreasing sequence Z defined by $Zn = \sup(Xm | m \geq n)$, which are well-defined by completeness. Then, by completeness again, sup Y and inf Z exist, which are called "liminf X" and "limsup X" respectively.

3.1 Definition. A sequence X in a complete lattice L' is "convergent" iff

$$\liminf X = \limsup X.$$

In such a case we define lim X by

$$\begin{aligned} \lim X &= \liminf X \\ &= \limsup X. \end{aligned}$$

A sequence X in an ordered set is a "quasi-ascending chain" iff it is an ascending chain or there exists a number M s.t.

$$X1 < X2 < \ldots < XM = X(M+1) = \ldots = X(M+n) = \ldots \quad .$$

In the latter case X is said to be "semi-finite".

3.2 Proposition. Let f be a function s.t. $f: L \to L'$. f(X), i.e. the sequence (f(Xn)), is convergent for any semi-finite X, and

$$\lim f(X) = f(\sup X).$$

Proof. Apparently

$$\lim f(X) = f(X^M) \text{ and } X^M = \sup X,$$

where M satisfies the condition of definition 3.2.

3.3 Proposition. Let f be a function s.t. $f: L \to L'$. f is ω-continuous iff

$$f(\sup X) = \sup f(X),$$

for any countable directed set X s.t. $X \subseteq L$.

Proof. The sufficiency is trivial. We prove the necessity. Let X be a countable directed set s.t. $X \subseteq L$. Then we can choose a quasi-ascending chain Y s.t. $Y \subseteq X$ and Y is cofinal in X so that

$$\sup Y = \sup X.$$

Suppose f is ω-continuous. Then, by ω-continuity,

$$f(\sup Y) = \sup f(Y).$$

But

$$\sup f(Y) \leq \sup f(X),$$

since

$$Y \subset X.$$

Thus

$$\begin{aligned} f(\sup X) &= f(\sup Y) \\ &= \sup f(Y) \\ &\leq \sup f(X). \end{aligned}$$

By monotonicity (see the remark in section 2.2),

$$f(x) \leq f(\sup X) \qquad \text{for any } x \in X,$$

since

$$x \leq \sup X,$$

so that

$$\sup f(X) \leq f(\sup X).$$

Therefore

$$f(\sup X) = \sup f(X).$$

3.4 Definition. Let L be an ω-inductively ordered set, and L' a complete lattice. $f: L \to L'$ is "weakly continuous" iff

$$f(\sup X) = \lim f(X),$$

for every ascending chain X in L. (This relationship implies that $\lim f(X)$ exists, for the left hand side always exists.)

3.5 Proposition. f is weakly continuous iff

$$f(\sup X) = \lim f(X)$$

for any quasi-ascending chain X.

Proof. Apparent from proposition 3.2.

3.6 Theorem. f is ω-continuous iff f is weakly continuous and monotone.

Proof. necessity: Suppose f is ω-continuous. Then f is monotone, so that for any ascending chain X

$$f(X1) \le f(X2) \le \ldots \quad .$$

Therefore

$$\sup f(X) = \lim f(X).$$

By ω-continuity,

$$f(\sup X) = \sup f(X),$$

so that

$$f(\sup X) = \lim f(X).$$

sufficiency: Let X be an quasi-ascending chain. We have to show

$$f(\sup X) = \sup f(X).$$

By weak continuity,

$$f(\sup X) = \lim f(X),$$

and, by monotonicity,

$$\lim f(X) = \sup f(X),$$

so that

$$f(\sup X) = \sup f(X).$$

3.7 Theorem. f is weakly continuous iff for any ω-continuous function g: L→L the following relationship holds.

$$f(\mathrm{Min}\ g) = \lim f(g.n.0),$$

where Min g denotes the least fixed point of g, i.e. inf(x|g(x)=x), which can be expressed as sup(g.n.0) (see section 2.2.)

We need the following lemma in order to prove this theorem.

3.8 Lemma. Let X be a quasi-asending chain in L. Then there exists an ω-continuous function f: L→L s.t.

$$f.n.0 = Xn \qquad \text{for any } n.$$

Proof of lemma. The following construction suffices.

$$f(x) = \begin{cases} X1, & x=U; \\ X(n+1), & x\neq \emptyset \text{ and } x\le Xl \text{ does not hold for any } l \\ & \text{s.t. } l\le n-1, \text{ and } x\le Xn \text{ holds } (n\ge 1); \\ \sup X, & x\le Xn \text{ does not hold for any } n. \end{cases}$$

(This construction was given by R. Milner.)

Proof of theorem 3.7. necessity: Suppose g is --continuous, then

$$\mathrm{Min}\ g = \sup(g,n,0).$$

(g,n,U) is a quasi-ascending chain, so that by weak continuity

$$f(\mathrm{Min}\ g) = \lim f(g,n,0).$$

sufficiency: Let X be a quasi-ascending chain in L. Then by lemma 3.8 there exists an --continuous function g s.t.

$$g,n,0 = Xn.$$

Assume

$$f(\mathrm{Min}\ g) = \lim f(g,n,0).$$

We note that

$$\mathrm{Min}\ g = \sup X$$

and

$$\lim f(g,n,0) = \lim f(X).$$

Therefore

$$f(\sup X) = \lim f(X).$$

3.9 Theorem. f is weakly continuous iff

$$\limsup f(X) = f(\sup X)$$

for every ascending chain X in L.

Proof. The necessity is trivial, so that we prove the sufficiency. Let X be an ascending chain in L. We prove that

$$\liminf f(X) = \limsup f(X)$$

follows the latter condition of the theorem. Let a and b denote liminf f(X) and limsup f(X), respectively. We prove a=b. We can choose a subsequence Y of X s.t.

$$\lim f(Y) = a,$$

since a is liminf f(X). Then, by definition,

$$\limsup f(Y) = a,$$

We note that Y is also an ascending chain in L, so that

$$\limsup f(Y) = f(\sup Y)$$

by the supposition of the theorem. Since Y is cofinal in X,

$$\sup Y = \sup X,$$

so that

$$f(\sup Y) = f(\sup X).$$

But

$$f(\sup X) = b$$

again by the supposition of the theorem. Thus

$$\limsup f(Y) = b.$$

Namely,

$$a = b.$$

4 Admissibility of Fixed-Point Induction

We shall discuss properties of predicates. For the convenience of mathematical description we introduce the ordering of truth values such as

$$F \le T.$$

This ordering is outside our logic, and it must be noted that the concept of weak continuity of predicates as well as that of admissibility of induction introduced below can be stated without referring to this ordering (see 4.6 below), though it makes some arguments more understandable.

Since we considered total predicates when we interpreted formulas, the concept of monotonicity or ω-continuity has little importance as long as we assume T and F are not comparable with each other. For, then, the only monotone or continuous predicates are the identically true predicate and the identically false one. We shall use, however, the concepts of monotonicity and continuity of predicates with respect to the above ordering. These concepts are mainly related to the existential quantifier.

4.1 Definition. Let TO denote the complete two element lattice. Namely TO consists of 0 and I, while $0 \leq I$. (TO can be regarded as a T0-space whose open sets are $\emptyset$=( ), (I), and (0,I), which is also a continuous lattice, as discussed by D. Scott.) We shall use this lattice to represent the truth values, 0 and I corresponding to F, i.e. false, and T, i.e. true, respectively, so that

$$F \leq T.$$

4.2 Definition. A "truth function" on L is a function s.t.

$$L \to TO.$$

a) A truth function f "admits induction weakly" iff

$$f(g,n,0) = T \text{ for every } n\ (n \geq 0) \text{ implies } f(\text{Min } g) = T.$$

Especially, f(x) admits induction weakly if f(0) = F.

b) A truth function f on L "admits induction strongly" iff

$$\lim f(g,n,0) = T \text{ implies } f(\text{Min } g) = T.$$

4.3 Proposition. Let X denote an ascending chain in L, and f a truth function on L.

a) f admits induction weakly iff

$$f(Xn) = T \text{ for every } n\ (0 \leq n) \text{ implies } f(\sup X) = T,$$

for any X.

b) f admits induction strongly if f admits induction weakly and f(0) = T.

c) The following conditions are equivalent to each other.

(i) f admits induction strongly.

(ii) $\lim f(X) \leq f(\sup X)$ for any ascending chain X for which $\lim f(X)$ exists.

(iii) $\limsup f(X) \leq f(\sup X)$ for any ascending chain X.

Proof. a) similar to the proof of theorem 3.7 using lemma 3.8.

b) Suppose

$$\lim f(X) = T.$$

Then

$$f(Xn) = T \quad \text{for almost every } n, \text{ (see 4.6)}$$

so that we can choose a quasi-ascending subchain Ym of X s.t.

$Y1 = O$ and $f(Ym) = T$ for every m.

By weak admissibility,

$f(\sup Y) = T.$

By cofinality,

$f(\sup X) = T.$

c) We prove that (i) implies (iii), the rest being left to the reader. Suppose

$\limsup f(X) \leq f(\sup X).$

If $\limsup f(X) = F$, then $\lim f(X) = F$, so that

$\lim f(X) \leq f(\sup X).$

Suppose

$\limsup f(X) = T,$

Then we can choose an ascending subchain Y of X s.t.

$\lim f(Y) = T.$

By cofinality of Y in X,

$\sup Y = \sup X,$

and, by strong admissibility,

$f(\sup Y) = T.$

Thus

$\limsup f(X) = f(\sup X).$

4.5 Theorem. Of the following conditions the upper ones are implied by the lower ones.

(i) f admits induction weakly.
(ii) f admits induction strongly.
(iii) f is weakly continuous.
(iv) f is ω-continuous.

Proof. We shall see that (iii) implies (ii), the rest having been proved. Suppose f is weakly continuous, and lim f(X) exists. Then

$\lim f(X) = f(\sup X),$

by weak continuity.

4.6 Remark. As noted in the beginning, the concepts of admissibility of induction and weak continuity of truth functions are independent of the ordering of truth values, for we can regard the relationship

$\lim f(g,n,0) = T$

simply as stating

$f(g,n,0) = T$ for almost every n,

because of the finiteness (thence discreteness, see 5.4) of TO. Thus these conditions can be restated as follows.

a) A truth function f admits induction weakly iff

$f(g,n,0) = T$ for every n ($n \geq 0$) implies $f(\mathrm{Min}\ g) = T$.

b) f admits induction strongly iff

$f(g,n,0)$ almost every n implies $f(\mathrm{Min}\ g) = T$.

c) f is weakly continuous iff

$f(g,n,0) = f(\mathrm{Min}\ g)$ almost every n.

4.7 Definition. Let x be an ω-variable, and A a formula in which at most x occurs free. A "admits induction weakly w.r.t. x in D" iff the truth function determined by A and x in D admits induction weakly. If A is an arbitrary formula, A "admits induction weakly w.r.t. x in D" iff every (D,x)-instance of A admits induction weakly.

A "admits induction weakly w.r.t. x" iff A admits induction weakly in any D.

We define the concepts that A "admits induction strongly w.r.t. x (in D)" and that A is "weakly continuous in x (in D)" similarly.

4.8 Theorem. The induction axiom $A[0] \rightarrow \forall y(A[y] \rightarrow A[x(y)]) \rightarrow A[\mathrm{Min}\ x]$ is valid iff A admits induction weakly w.r.t. x.

Proof. We prove the sufficiency first. Sufficiency: Let D be any collection of Dω's, $B[0] \rightarrow \forall y(B[y] \rightarrow B[a(y)]) \rightarrow B[\mathrm{Min}\ a]$ be a D-instance of the induction axiom, so that at most y occurs free in $B[y]$. Let $F(x)$ denote the truth function determined by $B[x]$, and $f(x)$ the function determined by $a(x)$, i.e., $\neq a$. $B[x]$ admits induction weakly in D because of the assumption that $A[x]$ does. Thus

$F(f,n,0)$ for any $n \geq 0$ implies $F(\mathrm{Min}\ f)$, while Min f is *(Min a). Assume the truth values of B[O] and ∀y(B[y]→B[a(y)]) are both T. Then

F(U)=T,

and

F(b)=T implies F(f(b))=T for any b.

Therefore we have

F(f,n,O)=T every n≥0,

so that, by weak admissibility of F(x),

F(Min f)=T.

Therefore the truth value of B[Min a] is T. Thus B[O]→∀y(B[y]→B[a(y)])→B[Min a] is valid in D. Hence the induction axiom is valid in D.

necessity: We use the same notations as above. By definition of validity any D-instance of the axiom must be valid. Therefore if the truth values of B[O] and ∀y(B[y]→B[a(y)]) are both T, i.e., F(f,n,O)=T every n≥0, the truth value of B[Min a] is T. Namely F(Min f)=T.

4.9 Definition. A[x] "admits relativized induction w.r.t. x" iff A[x] makes the induction rule sound. Namely, the rule obtained from the schema of induction rule by substituting A[x] in place of the meta-variable A is sound.

4.10 Theorem. A[x] admits relativized induction if A[x] admits induction weakly.

Proof. We have to prove the soundness of the following rule.

$$\frac{P \dashrightarrow A[O] \qquad A[a],\ P \dashrightarrow A[t(a)]}{P \dashrightarrow A[\mathrm{Min}_n\ t]}\ \langle a \rangle$$

Let C denote C1 & ... & Cm where P is C1, ... , Cm. The rule is sound iff (C→A[O])→(C→∀y(A[y]→A[t(y)]))→(C→A[Min t]) is valid by definition. Therefore we shall prove for any D, every D-instance of this formula, say (E→B[O])→(E→∀y(B[y]→B[b(y)]))→(E→B[Min b]), is valid, where b is an arbitrary variable-free term of the same type as t, i.e. (αo→αo)o for some αo. Obviously we have only to prove for the case that b is a name. For, if b is not a name, there is some name b' s.t. *b=*b', and the validity can be established easily using this fact and the case that b is a name. Let F(x) be the truth function determined by B[x], and f be *b. Then the following condition is sufficient.

If ≡E≡T, F(O)≡T, and, ≡E≡T and F(c)≡T imply F(f(c))≡T for any c∈D∞, then F(Min f)≡T.

Min f is ≡(Min D) by definition. Assume the premise of the above condition. Then by induction,

F(f,n,O)≡T every n≥0.

By the assumption that A[x] admits induction weakly, so does B[x], so that $F(_x)$ admits induction weakly. Therefore

F(Min f)≡T.

4.11 Theorem. A[x] admits lcf induction if A[x] admits relativized induction, where by lcf induction is meant the relativised rule in LCF to infer A[y] from y≡Min x, A[O], and A[a] → A[x(a)].

Proof. We see that lcf induction rule is a derived rule.

```
y≡Min x, P ⟶ A[O]          A[a], y≡Min x, P ⟶ A[x(a)]
------------------------------------------------------- <a> induction
              y≡Min x, P ⟶ A[Min x]
-------------------------------------------------------      equality
              y≡Min x, P ⟶ A[y]
```

4.12 Corollary. A[x] admits lcf induction if A[x] admits induction weakly.

5 Characterization of Predicates that admit Fixed-Point Induction

We study what kind of formulas admit induction. For the readability of proofs we shall discuss them in terms of truth functions that have one argument designated by x. The theorems below can be so applied to every instance of formula that the results will be regarded as statements about formulas in general by definition (see 4.7.) For this purpose logical combinators below should be understood as functions or functionals whose values are T or F. For instance $\forall y F(x,y)$ denotes the truth function determined by $\forall y A[x,y]$ where $F(x,y)$ is the truth function determined by $A[x,y]$ in D. The relation $\leq$ is not a logical symbol of FLT, but it will be used as a predicate later on in connection with LCF.

5.1 Theorem. The relationship $f(x) \leq g(x)$ admits induction strongly if $f(x)$ and $g(x)$ are ω-continuous.

Proof. Let $F(x)$ denote the corresponding truth function, i.e.,

$$F(x) = \begin{cases} T & f(x) \leq g(x); \\ F & \text{otherwise.} \end{cases}$$

Let X be an ascending chain in L. Suppose

$$\lim F(X) = T,$$

so that

$$f(X_n) \leq g(X_n) \quad \text{for almost every } n.$$

Then, by monotonicity of g,

$$f(X_n) \leq g(\sup X) \quad \text{for almost every } n.$$

Therefore we can choose an ascending subchain Y of X s.t.

$$f(Y_m) \leq g(\sup X) \quad \text{for every } m.$$

Thus

$$\sup f(Y) \leq g(\sup X).$$

But, by ω-continuity of f,

$$f(\sup Y) = \sup f(Y),$$

so that

$$f(\sup Y) \leq g(\sup X).$$

By cofinality of Y in X,

$$\sup Y = \sup X,$$

Thus

$f(\sup X) \leq g(\sup X)$,

i.e.,

$F(\sup X) = T$.

5.2 Remark. $f(x) \leq g(x)$, f and g being continuous, is not always weakly continuous, (the fact that it is not ω-continuous being well-known.) Let N' be the natural numbers with the infinity ω (omega) ordered in the usual sense. Define f, g: N'→N' by

$f(x) = x+1$,

$g(x) = x$.

Let X be s.t.

$X_n = n$ each n s.t. $1 \leq n < \omega$.

Then

$F(X_n) = F$ each n,

but

$F(\sup X) = T$.

5.3 Theorem. Let f be an ω-continuous function into a discrete lattice L', c an element of L. Then the relationship

$f(x)=c$

is weakly continuous.

Proof. Let X be an ascending chain in the domain of f. By theorem 5.1, $f(X_n)=c$ almost every n implies $f(\sup X)=c$. Suppose

$f(X_n)\neq c$ almost every n.

We have to prove

$f(\sup X)\neq c$

Let Y_n denote $f(X_n)$ for each n. By monotonicity of f,

$a=Y_1 \leq \ldots Y_n \leq Y(n+1) \leq \ldots \leq b$,

where b denotes $\sup Y$. Y must have at least an accumulating point, for, otherwise, we could choose an ascending chain Z that is a subset of Y s.t.

$a<Z_1< \ldots <Z_n<Z(n+1)< \ldots <b$,

which contradicts the discreteness of L'. By monotonicity such an accumulating point is unique and will be denoted by d. Thus

Yn=d   almost every n.

By the supposition

c≠d.

By monotonicity again, d is the maximum element of Y, so that

d=sup f(X).

By ω-continuity,

f(sup X)=sup f(X)
=d
≠c.

Thus

f(sup X)≠c.

5.4 Remark. In the above proof we used "discreteness" to mean there is no ascending chain s.t.

a<X1<X2< ... <Xn< ... <b,

for any a and b.

5.5 Theorem. a) F(x)vG(x) admits induction strongly if F(x) and G(x) do.
b) F(x)vG(x) is weakly continuous if F(x) and G(x) are.

Proof. a) Suppose

limsup F(X)vG(X) = T.   (Cf. 4.3.c(iii))

Then either

limsup F(X) = T

or

limsup G(X) = T,

so that either

F(sup X) = T

or

G(sup X) = T

by strong admissibility. Thus

F(sup X)vG(sup X) = T.

b) By weak continuity,

F(sup X) = F(Xn) = a   for almost every n

and

G(sup X) = G(Xn) = b   for almost every n, for some a and b. Therefore

$$F(X_n) \vee G(X_n) = a \vee b \qquad \text{for almost every } n.$$

At the same time,

$$F(\sup X) \vee G(\sup X) = a \vee b.$$

5.6 Remark. a) $F(x) \vee G(x)$ does not necessarily admit induction weakly even if $F(x)$ and $G(x)$ do. We consider N' (see remark 5.2) again. Let

$$F(x) = \begin{cases} T & x=0; \\ F & 0<x\le\infty; \end{cases}$$

and

$$G(x) = \begin{cases} T & 0<x<\infty; \\ F & x=0 \text{ or } x=\infty. \end{cases}$$

Then F and G admit induction weakly, and

$$F(n) \vee G(n) = T \quad \text{for every } n\ge 0.$$

But

$$F(\infty) \vee G(\infty) = F.$$

b) $F(x) \vee G(x)$ does not necessarily admit induction weakly even if one of $F(x)$ and $G(x)$ is weakly continuous and the other admits induction weakly. For, in fact, $F(x)$ in the above example is weakly continuous.

5.7 Theorem. a) $F(x) \& G(x)$ admits induction weakly if $F(x)$ and $G(x)$ do.
b) $F(x) \& G(x)$ admits induction strongly if $F(x)$ and $G(x)$ do.
c) $F(x) \& G(x)$ is weakly continuous if $F(x)$ and $G(x)$ are.

Proof: left to the reader.

5.8 Theorem. $\neg F(x)$ is weakly continuous if $F(x)$ is.

Proof. Let a denote the truth value $F(\sup X)$. By weak continuity,

$$F(X_n) = a \qquad \text{for almost every } n.$$

Let b denote the truth value $\neg a$. Then

$$\neg F(X_n) = b \qquad \text{for almost every } n.$$

Besides,

$$\neg F(\sup X) = b.$$

5.9 Remark. a) $\neg F(x)$ does not necessarily admit induction weakly even if $F(x)$ admits induction strongly. Let $F(x)$ be the truth function determined by $x\le\infty$ & $\infty\le x$, which is equivalent to $x=\infty$, in N'. Then $F(x)$ admits induction strongly because of theorems 5.1 and 5.7(b).

Let Xn be the n-th natural number for each n. Then

$$\neg F(Xn) = T \qquad \text{for } n \ge 0.$$

But

$$\neg F(\sup X) = \neg F(\infty) = F.$$

Thus $\neg F(x)$ does not admit induction weakly.

b) By the above argument, the negation of a formula of LCF does not admit induction weakly in general.

5.10 Theorem. If $F(x)$ and $\neg F(x)$ both admit induction strongly then $F(x)$ is weakly continuous.

Proof. We prove that $F(X)$ is convergent for any ascending chain X. The case that

$$\text{limsup } F(X) = F$$

is trivial. Suppose

$$\text{limsup } F(X) = T,$$

We prove

$$\text{liminf } F(X) = T$$

by contradiction. Assume

$$\text{liminf } F(X) = F,$$

so that

$$\text{limsup } \neg F(X) = T.$$

By strong admissibility,

$$\neg F(\sup X) = T,$$

i.e.,

$$F(\sup X) = F.$$

Thus $F(x)$ does not admit induction strongly, which is a contradiction.

5.11 Theorem. a) $F(x) \supset G(x)$ admits induction strongly if $F(x)$ is weakly continuous and $G(x)$ admits induction strongly.

b) $F(x) \supset G(x)$ is weakly continuous if $F(x)$ and $G(x)$ are.

Proof. $F(x) \supset G(x)$ is a tautology of $\neg F(x) \vee G(x)$, so that theorems 5.8 and 5.5 suffice.

5.12 Remark. $F(x) \supset G(x)$ does not necessarily admit induction weakly even if $F(x)$ admits induction strongly and $G(x)$ is ∞-continuous. Let $G(x)$ be F, i.e., the identically false truth function. Then $F(x) \supset G(x)$ is a tautology of $\neg F(x)$. Consider the example of remark 5.9.

Hereafter y is used to indicate the argument instead of x.

5.13 Theorem. a) $\forall x F(x,y)$ admits induction weakly w.r.t. y if $F(x,y)$ does.
b) $\forall x F(x,y)$ admits induction strongly w.r.t. y if $F(x,y)$ does.

Proof. a) Suppose

$$\forall x F(x,0) = T$$

and

$$\forall x F(x,Yn) = T \quad \text{for every } n.$$

Then

$$F(a,0) = T$$

and

$$F(a,Yn) = T \quad \text{for every } n.$$

for any a. Therefore, by weak admissibility,

$$F(a,\sup Y) = T \quad \text{for any } a.$$

Thus

$$\forall x F(x,\sup Y) = T.$$

b) Suppose

$$\limsup \forall x F(x,y) = T.$$

Then

$$\limsup F(a,Yn) = T \qquad \text{each } a.$$

By strong admissibility,

$$F(a,\sup Y) = T \qquad \text{each } a.$$

Thus

$$\forall x F(x,\sup Y) = T.$$

5.14 Remark. a) $\forall x F(x,y)$ is not necessarily weakly continuous even if $F(x,y)$ is. Let F be s.t.

$$F(x,y) = \begin{cases} T & x<\infty \text{ and } x<y, \text{ or } x=\infty; \\ F & \text{otherwise.} \end{cases}$$

Then F is weakly continuous in y, for

$$\lim F(a,Yn) = F(a,\infty) = T \qquad \text{each } a<\infty,$$

and

$$F(\infty,Yn) = F(\infty,\infty) = T \quad \text{for every } n,$$

for any ascending chain Yn in N'. Moreover,

$$\forall x F(x,Yn) = F \quad \text{for every } n,$$

so that

$$\lim \forall x F(x,Y) = F.$$

But

$$\forall xF(x,\sup Y) = \forall xF(x,\omega) = T.$$

b) $\forall xF(x,y)$ is not necessarily weakly continuous even if $F(x,y)$ is ω-continuous in y, for $F(x,y)$ defined above is ω-continuous in y, because it is not only weakly continuous but also monotone (cf. theorem 3.6.)

5.15 Theorem. a) $\exists xF(x,y)$ admits induction strongly if $F(x,y)$ is monotone in y.
b) $\exists xF(x,y)$ is monotone and weakly continuous (and therefore ω-continuous. See theorem 3.6) if $F(x,y)$ is.

Proof. a) Suppose

$$\lim \exists xF(x,Y) = T,$$

so that for some a and M

$$F(a,YM) = T.$$

By monotonicity,

$$F(a,\sup Y) = T.$$

Thus

$$\exists xF(x,\sup Y) = T.$$

b) We prove

$$\exists xF(x,\sup Y) = \liminf \exists xF(x,Y) = \limsup \exists xF(x,Y)$$

for each ascending chain Y by case analysis. (i) Suppose

$$\limsup \exists xF(x,Y) = T,$$

so that

$$F(a,YM) = T \qquad \text{for some a and M.}$$

By monotonicity,

$$F(a,Yn) = T \qquad M \le n,$$

so that

$$\exists xF(x,Yn) = T \qquad M \le n,$$

i.e.,

$$\lim \exists_x F(x,Y) = T.$$

Also by monotonicity, $F(a,YM) = T$ implies

$$F(a,\sup Y) = T,$$

so that

$$\exists xF(x,\sup Y) = T.$$

(ii) Suppose

$$\limsup \exists xF(x,Y) = F,$$

i.e.,

$$\lim \exists xF(x,Y) = F.$$

Then there exists M(a) for each a, s.t.

$$F(a,YM(a)),$$

so that, by monotonicity,

$$F(a,Yn) = F \qquad \text{for any } a \text{ and } n,$$

(otherwise $F(b,Ym) = T$, $M \leq m$ for some b and M, which implies $\lim \exists x F(x,Y) = T$.) Thus

$$\lim F(a,Y) = F \qquad \text{for any } a,$$

so that by weak continuity,

$$F(a,\sup Y) = F \qquad \text{for any } a,$$

Therefore

$$\exists x F(x,\sup Y) = F.$$

5,16 Remark. a) $\exists x F(x,y)$ is not necessarily weakly continuous even if $F(x,y)$ is monotone (and therefore admits induction strongly by theorem 5,15) in y. Let $F(x,y)$ be

$$\forall z(z<\infty \rightarrow z<y).$$

Then $F(x,y)$ is monotone in y, and

$$F(x,n) = F \qquad \text{for every } n \text{ and any } x,$$

so that

$$\exists x F(x,n) = F \qquad \text{for every } n.$$

But

$$\exists x F(x,\infty) = T,$$

because

$$F(x,\infty) = T,$$

b) $\exists x F(x,y)$ is not necessarily weakly continuous even if $F(x,y)$ is monotone and admit induction strongly. Let

$$G(x,y) = \begin{cases} T & y \leq x < \infty; \\ F & \text{otherwise.} \end{cases}$$

Namely,

$$G(x,y) \equiv \neg F(x,y),$$

$F(x,y)$ being the truth function described in remark 5,14 so that $G(x,y)$ is weakly continuous in y by theorem 5.8. But

$$\exists x G(x,n) = T \qquad \text{every } n,$$

and

$$\exists x G(x,\infty) = F.$$

6 Syntax of Formulas that admit Induction

6.1 Tables of inheritance of adm sibility

We summarize the inheritance of admissibility of induction in the tables so that they can be checked by machines easily.

These tables shall be regarded as a part of the postulates of FLT for technical (logical) reasons. Since the weak admissibility of induction is an informal concept that is not effective, we cannot accept a formal system described in terms of that concept, although we would like to use the induction axiom, or rule, for every formula that admits induction weakly. Instead we regard these tables as an inductive definition, and hence an effective definition, of formulas that "admit induction syntactically". Namely we call a formula A[ ] to admit induction syntactically iff A[x] is concluded to admit induction weakly w.r.t. x using only these tables, the primitive cases listed in I1 serving as the base step of inductive definition.

We add the following definition for practical purposes.

Definition. A formula A is said to be "constant w.r.t. x" iff #A does not depend on x. A term t is an "lcf term" iff all the constants and variables occurring in t are of continuous types. A formula of the form t≤u where t and u are lcf terms is called an "lcf awff".

Obviously a sufficient condition for A to be constant w.r.t. x is that x does not occur free in A. Proofs concerning the inheritance of admissibility related to this condition are left to the reader.

I1. The following conditions are hierarchical in the sense that the lower are the stronger conditions.

	(primitive cases)
A admits induction weakly.	
A admits induction strongly.	t≤u (t and u are lcf terms)
A is weakly continuous.	t=0, t=TRUE, t=FALSE (t is an lcf term)
A is constant.	x does not occur free in A.

(i) A admits relativized induction and lcf induction w.r.t. x if A admits induction weakly w.r.t. x.

(ii) A is ω-continuous iff A is weakly continuous and monotone.
(iii) A admits induction weakly w.r.t. x if A is monotone w.r.t. x.

12. Table for &, v, and →.

If A and B satisfy the conditions stated in the first column and the first row, respectively, then A&B, AvB, and A→B satisfy the conditions shown in the corresponding places.

A \ B	op	adm, weak,	adm, str.	weak. cont,	const,
adm, weak,	&	adm, weak,	adm, weak,	adm, weak,	adm, weak,
	v	x	x	x	adm, weak,
	→	x	x	x	adm, weak,
adm, str.	&	adm, weak,	adm, str.	adm, str.	adm, str.
	v	x	adm, str.	adm, str.	adm, str.
	→	x	x	x	adm, str.
weak. cont.	&	adm, weak,	adm, str.	weak. cont,	weak, cont,
	v	x	adm, str.	weak. cont,	weak, cont,
	→	x	adm, str.	weak. cont,	weak, cont,
const,	&	adm, weak,	adm, str.	weak. cont,	const,
	v	adm, weak,	adm, str.	weak. cont,	const,
	→	adm, weak,	adm, str.	weak. cont,	const,

13. Table for ¬, ∀, and ∃.

All the conditions are w.r.t. x.
If x and y are identical then ∀yA and ∃yA are constant w.r.t. x.

A	¬A	∀yA	∃yA	
			in general	A: monotone
adm, weak.	x	adm, weak,	x	adm, str.
adm. str.	x	adm, str.	x	adm, str.
weak, cont.	weak. cont,	adm, str.	x	weak, cont,
const.	const,	const.	const,	const,

6.2 Example of formula that admits induction

$\forall xF(x,y)$ is weakly continuous if $F(x,y)$ is anti-monotone and admits induction strongly w.r.t. y.

For, $\forall xF(x,y)$ is a tautology of $\neg\exists x\neg F(x,y)$. Suppose $F(x,y)$ is anti-monotone and admits induction strongly w.r.t. y. $\neg F(x,y)$ is monotone, so that $\neg F(x,y)$ admits induction strongly by theorem 5.4a. Then $\neg F(x,y)$ is weakly continuous by theorem 4.8, so that $\exists xF(x,y)$ is weakly continuous by theorem 5.4b. Thus $\neg\exists xF(x,y)$ is weakly continuous by theorem 4.6. (See tables of 6.1)

We can check this result by a direct proof as follows.

Proof. Case i) Suppose

$\limsup \forall xF(x,Y) = F$, i.e., $\lim \forall xF(x,Y) = F$.

Then there exists M s.t.

$\forall xF(x,YM) = F$,

so that there is some a s.t.

$F(a,YM) = F$.

By anti-monotonicity,

$F(a,\sup Y) = F$,

so that

$\forall xF(x,\sup Y) = F$.

Case ii) Suppose

$\limsup \forall xF(x,Y) = T$.

Then,

$\limsup F(a,Y) = T$ each a,

so that

$F(a,Yn) = T$ for every n, each a,

i.e.,

$\lim F(a,Y) = T$.

(Otherwise $\limsup F(a,Y) = F$ by anti-monotonicity.) By strong admissibility,

$F(a,\sup Y) = T$ each a,

so that

$\forall xF(x,\sup Y) = T$.

7 Translation of LCF into First-Order Logic of Typed Theories

7.1 Axiomatization

In order to axiomatize LCF, first we need to extend the syntax of terms so as to include λ-expressions as follows.

B5. If t is an αo-term and x is a βo-variable, then λxt is a (βo→αo)o-term. Any occurrence of x in λxt is not free.

The corresponding interpretation is as follows.

D5. If t is λxu[x] and x is a βo-variable, u[a] must be a closed αo-term for each βo-name a so that $*(u[a]) \in D\alpha o$, for some αo. We let *t be the function which sends each $*a \in D\beta o$ onto $*(u[a])$. Such a function is known to be continuous[10, 7].

Remark. The proof of continuity of the functions represented by λ-expressions, namely the terms involving the operator λ, requires induction on the structure of terms. The case that sup Dα's do not exist in general has been treated by R. Milner.

We introduce an ordered base type denoted by Bo, three Bo-constants Ω, TRUE, and FALSE, and, a (Bo→αo→αo→αo)o-constant ⊃ and an (α,α)-predicate ≤ for each α.

D[Bo] consists of three elements, TRUE* and FALSE* being incomparable. Hereafter we use the same symbol to denote a Bo-constant and the truth value represented by it.

⊃(t,u,v), namely ((⊃(t))(u))(v) reads "if t then u else v" and is written as t⊃u,v usually. We let a⊃b,c be Ω, b, and c, if a is Ω, TRUE, and FALSE, respectively, for each a∈Bo, b∈Dαo, and c∈Dαo. This function is continuous[10].

x≤y represents the order relation discussed in the previous sections, mathematically. Intuitively, however, x≤y means that y is "defined" more than or as much as x. x=Ω read "x is undefined." If x and y are functions, this means y is an extension of x as function.

We give the following non-logical axioms. An arbitrary term with voids can be substituted in place of t[ ], provided that the variable designated by x does not occur free in that term. t[x] and t[y] denote the terms obtained from it by substituting arbitrary variables designated by x and y, respectively, in place of its voids.

Nonlogical axioms

reflexivity.	$x \leq x$.	
antisymmetry.	$x \leq y$ & $y \leq x \rightarrow x \equiv y$.	
	$x \equiv y \rightarrow x \leq y$.	
transitivity.	$x \leq y$ & $y \leq z \rightarrow x \leq z$.	
extensionality.	$\forall z(x(z) \leq y(z)) \rightarrow x \leq y$.	
	$x \leq y \rightarrow x(z) \leq y(z)$.	
monotonicity.	$x \leq y \rightarrow z(x) \leq z(y)$	z must be an eo-variable.
minimal elements.	$0 \leq x$.	
	$0(x) \leq 0$.	
truth values.	$x \equiv 0 \lor x \equiv TRUE \lor x \equiv FALSE$	x must be a Bo-variable.
	$\neg 0 \equiv TRUE$.	
	$\neg 0 \equiv FALSE$.	
	$\neg TRUE \equiv FALSE$.	
conditionals.	$0 \supset x, y \equiv 0$.	
	$TRUE \supset x, y \equiv x$.	
	$FALSE \supset x, y \equiv y$.	
λ-conversion.	$(\lambda x\, t[x])(y) \equiv t[y]$.	

7.2 Adequacy

We need to see that all the inference rules in LCF can be adequately expressed in the present calculus in the form of theorems or derived rules, which means that we do not lose anything by changing the logic. In other words, we are dealing with an extension of LCF in that we can prove a theorem A in the new calculus if A is a theorem in LCF, and, moreover, we can use any rule of LCF in the present calculus. We have only to examine those rules that are neither of the nature of propositional calculus nor expressed as one of the logical or nonlogical axioms.

J1. abstraction rule (LCF).

$$\frac{t[a] \le u[a]}{\lambda x t[x] \le \lambda x u[x]}\ \langle a\rangle$$

Derivation.

$$\begin{array}{lll}
\dfrac{\dfrac{\dfrac{t[a] \le u[a]}{\lambda x t[x](a) \le \lambda x u[x](a)}}{\forall y((\lambda x t[x])(y) \le (\lambda x u[x])(y))}\ \langle a\rangle}{\lambda x t[x] \le \lambda x u[x]} & & \begin{array}{l}\lambda\text{-conversion (and equality)}\\ \forall\text{-introduction}\\ \text{extensionality}\end{array}
\end{array}$$

J2. function rule (LCF).

$$\frac{}{\lambda x y(x) \equiv y}$$

Derivation.

$$\begin{array}{lll}
\dfrac{\dfrac{(\lambda x y(x))(z) \equiv y(z)}{\forall z((\lambda x y(x))(z) \equiv y(z))}\ \langle z\rangle}{\lambda x y(x) \equiv y} & & \begin{array}{l}\lambda\text{-conversion}\\ \forall\text{-introduction}\\ \text{extensionality}\end{array}
\end{array}$$

J3. cases rule (LCF).

$$\frac{\begin{array}{ccc}(t\equiv\Omega) & (t\equiv\mathrm{TRUE}) & (t\equiv\mathrm{FALSE})\\ A & A & A\end{array}}{A}$$

Derivation.

```
                             (t=0)    (t=TRUE)    (t=FALSE)
t=0 v t=TRUE v t=FALSE         A         A           A
-----------------------------------------------------------  v-elimination
                             A                                      (twice)
```

J4. Induction rule (LCF). It suffices to show that any conjunction of lcf awffs admits induction syntactically in the sense of section 6.1, for LCF is a formal system that carries out relativized deduction for these sentences. Each lcf awff admits induction strongly w.r.t. any variable (table I1, 6.1) subject to the type comformity. So does any conjunction of them (table I2, 6.1).

7.3 Example taken from proof of compiler correctness

The following example is taken from an FLT-like proof of McCarthy-Painter's theorem[5]. The proof of this theorem in LCF is discussed in [8] and [13].

We presuppose there are three types called language1, language2, and the meaning space. These need not be base types. In particular the meaning space can be the type (states)→(states). Namely the meaning space is the set of partial functions of (states) into itself. A conceptual compiler carries out a translation of language1 into language2, an expression x in language1 being mapped onto obj(x). We need not assume continuity of the meaning space and function obj for the present argument, which is, however, not an important point. We use the following constants, each of them being either an individual constant or a function in the usual sense. The asterisked constants are assumed to have been given appropriate axioms.

constant		type	comment
isconst	*	(language1→Bo)o	isconst(9)=TRUE.
isvar	*	(language1→do)o	isvar(a)=TRUE.
isexp		(language1→do)o	isexp((8+a)+(9+b))=TRUE.
arg1	*	(language1→language1)o	arg1((8+a)+(9+b))=8+a.
arg2	*	(language1→language1)o	arg2(8+a)=a.
obj	*	language1→language2	
mean1	*	language1→meaning space	
mean2	*	language2→meaning space	

We use a (language1,language2)-predicate Correct(x,y) to mean y is a correct object program for expression x. Correct(x,y) is not continuous in general, because it is usually defined by an axiom like

(Ax.1) $\forall x \forall y(\mathrm{Correct}(x,y) \equiv \mathrm{mean1}(x)=\mathrm{mean2}(y))$, (*)

The function isexp is defined by the following axiom.

(Ax.2) $\mathrm{isexp}=\mathrm{Min}\ \lambda f\ \lambda x(\mathrm{isconst}(x)\supset \mathrm{TRUE}, (\mathrm{isvar}(x)\supset \mathrm{TRUE},$
$(f(\mathrm{arg1}(x))\supset(f(\mathrm{arg2}(x))\supset \mathrm{TRUE}, \mathrm{FALSE}), \mathrm{FALSE})))$.

The theorem we want to prove is

(1) $\forall x(\mathrm{isexp}(x)=\mathrm{TRUE} \rightarrow \mathrm{Correct}(x,\mathrm{obj}(x)))$.

Correct(x,obj(x)) is, however, not sufficient as an induction hypothesis in general, so that we prove first a formula of the form

(2) $\forall x(\mathrm{isexp}(x)=\mathrm{TRUE} \rightarrow A)$,

usually, where A is the conjunction of a certain generalization of Correct(x,obj(x)) and additional conditions peculiar to each compiling algorithm. More concretely, we shall consider a compiler which works with a counter, n, indicating that the addresses whose mnemonic names are TS(1), ... , TS(n) are occupied as temporary storages. We define the following constants, the last three related to the loading or allocation. The set of integers, or addresses, is a base type. varsno(x) is the number of distinct variables occurring in x. varno(z,x) denotes some numbering of such variables.

constant		type	comment
compl	*	(language1,integers)→language2	
TS		integers→integers	
varno	*	(language1,language1)→integers	varno(a,(8+a)+(9+b))=1.
varsno	*	language1→integers	varsno((8+a)+(9+b))=2.
loc		(language1,language1)→integers	

In this case, obj(x) is defind by the following axiom.

(Ax.3) $\forall x(\mathrm{obj}(x)=\mathrm{compl}(x,\emptyset))$.

A typical form of A is

(3) $\forall n(n\geq\emptyset \rightarrow \mathrm{Correct}(x,\mathrm{compl}(x,n))\ \&\ \mathrm{Unaffected}(x,n,\mathrm{compl}(x,n)))$.

where Unaffected is a (language1,integers,language2)-predicate s.t. Unaffected(x,n,y) means the object program y does not destroy the contents of the storages corresponding to the program variables occurring in the source program x or any of TS(1), ... , TS(n).

----

*) The reader may recall that $\equiv$ means logical equivalence, while = equality in the strong sense, that is, $\equiv$ in LCF.

If we make the addresses absolute by the below axiom, which corresponds to a particular loading obviously, the object program becomes as fig. 1 below. Occur(z,x) reads z occurs in x.

(Ax. 4) $\forall z \forall x(\text{isvar}(z)=\text{TRUE} \to \text{Occur}(z,x) \to \text{loc}(z,x)=\text{varno}(z,x))$,

$\forall x \forall n(\text{loc}(\text{TS}(n))=\text{varsno}(x)+n)$.

```
    compl((8+a)+(9+b),n)                                memory map

 (instruction)          (mnemonics)              +----------------------+
 -------------          -----------              |0      |accumulator   |
   LI     8                                      |----------------------|
   ADD    1               a                      |1      |a             |
   STO    n+3             TS(n+1)                |2      |b             |
   LI     9                                      |----------------------|
   ADD    2               b                      |3      |TS(1)         |
   STO    n+4             TS(n+2)                        .....   .....
   LI     n+3             TS(n+1)                |n+2    |TS(n)         |
   ADD    n+4             TS(n+2)                |----------------------|
                                                 |n+3    |TS(n+1)       |
                                                 |n+4    |TS(n+2)       |
 Let n=0 to get obj((8+a)+(9+b)).                |----------------------|
```

fig. 1 Example of objcet program
and memory map

Let A[x] denote (3) hereafter. We note that neither isexp nor n occurs free in A[x]. Then, the formula (2) admits iof induction w.r.t. "isexp" as follows.

isexp(x)=TRUE	weak. cont. w.r.t. isexp;
A(x)	const. w.r.t. isexp;
isexp(x)=TRUE → A[x]	weak. cont. w.r.t. isexp;
∀x(isexp(x)=TRUE → A[x])	adm. str. w.r.t. isexp.
	(See tables in section 6.1.)

Thus we can infer (2) from (4) and (5) below.

(4) $\forall x(O(x)=\mathrm{TRUE} \rightarrow A[x])$.

(5) $\forall x(f(x)=\mathrm{TRUE} \rightarrow A[x]) \longrightarrow$
$$\begin{aligned}&\forall x((\mathrm{isconst}(x)\supset \mathrm{TRUE},(\mathrm{isvar}(x)\supset \mathrm{TRUE},\\ &\qquad (f(\mathrm{arg1}(x))\supset(f(\mathrm{arg2}(x))\supset \mathrm{TRUE},\\ &\qquad\qquad \mathrm{FALSE}).\mathrm{FALSE})))=\mathrm{TRUE} \rightarrow A[x]).\end{aligned}$$

We can improve the readability by the following consideration. Let p be an $(\alpha\rightarrow Bo)$-term. Then we let p and $\bar{p}$ stand for the formulas $p=\mathrm{TRUE}$ and $p=\mathrm{FALSE}$, respectively. This causes no comfusion because of the syntax we employed. Obviously

$$p \vee \bar{p}$$

is not valid, while $p\vee\neg p$ is. We notice the relationship

$$(p\supset q,r) \equiv p\&q \vee \bar{p}\&r, \qquad (*)$$

which is provable in FLT, since this formula is an abbreviation of

$$(p\supset q,r)=\mathrm{TRUE} \equiv p=\mathrm{TRUE} \& q=\mathrm{TRUE} \vee p=\mathrm{FALSE} \& r=\mathrm{TRUE}.$$

Thus we can rewrite (4) and (5) as follows.

(4′) $\forall x(O(x) \rightarrow A[x])$.

(5′) $\forall x(f(x) \rightarrow A[x]) \longrightarrow$
$$\begin{aligned}&\forall x(\mathrm{isconst}(x)\vee \mathrm{isvar}(x)\vee \overline{\mathrm{isconst}}(x)\&\overline{\mathrm{isvar}}(x)\\ &\qquad \& f(\mathrm{arg1}(x))\& f(\mathrm{arg2}(x)) \rightarrow A[x]).\end{aligned}$$

It must be noted that there are some substitutes in LCF for formulas like (1)-(4), though these formulas are not allowed as legitimate formulas in it and the interpretation becomes different. By the deduction theorem in first-order logic we can also express the sentence (5′) by a formula of FLT, replacing $\longrightarrow$ by $\rightarrow$ and binding f by universal quantifier, obtaining

(5″) $\forall f(\forall x(f(x) \rightarrow A[x]) \rightarrow$
$$\begin{aligned}&\forall x(\mathrm{iscontst}(x) \vee \mathrm{isvar}(x) \vee \overline{\mathrm{isconst}}(x) \&\\ &\overline{\mathrm{isvar}}(x) \& f(\mathrm{arg1}(x)) \& f(\mathrm{arg2}(x)) \rightarrow A[x])).\end{aligned}$$

For such a formula there seem to be no natural substitutes in the form of LCF formulas.

----
*) It is a little interesting, and also useful, that this old relationship still holds in a calculus that includes the undefined truth value. See, e.g., [2].

Discussions

The writer has been motivated toward the study described in this paper through an attempt to translate his formal system representing the equivalence of Algol-like statement[2, 3] into LCF. For that purpose having some predicate calculus-like facility seems to be essential, for we need to express implication between strong equivalence in the form of formula.

From the writer's point of view, the following are among the possible advantages of having some predicate calculus-like things within logic for computable functions.

1. (human engineering) In not a few cases, the conventional logical operators make the writing and understanding of descriptions easier. Besides, many people are familiar with expressions and derivation in predicate calculus, especially, of first-order.

2. (underlying theories) In the practical field of application of such a logic, for instance proving correctness of compilers, we have to handle underlying theories whose representations in predicate calculus seem to be natural, like elementary set theory. We do not care if some of the sets involved in our proof are not computable or continuous, even if they might be in fact computable. There are also theories of equivalence and correctness of programs which are related to predicate calculus.

3. (meta-theorems) There will be many facts about the objects of LCF that can be stated only in the form of meta-theorems of LCF, while significant portion of them could be stated as theorems in an extended logic. Then handling derived rules and applying already proved theorems will become more convenient.

Obviously these desirable properties will not be obtained before considerable experiments. Moreover there must be some compromise. For instance, if we use entire classical predicate calculus as in the present paper, we are out of the LCF-like world that consists of solely continuous functions, losing some neatness of the formalism and relative simplicity of implementation. Employing second or higher order predicate calculus might give us more complexity as well as power.

It must be noted that J. McCarthy[4] suggested that in some generalization of Scott's logic using predicate calculus we should be able to prove the continuity of functions. It seems that FLT is capable of doing that in spite of the limitation that no predicate variables are allowed, for we have quantifiers ranging over typed sets in effect. A fixed-point induction based mainly on monotonicity within second-order predicate calculus has been discussed by D. Park[9].

Acknowledgements

The writer acknowledges J. McCarthy, R. Milner, R. Weyhrauch, and R. London for stimulating discussions, valuable suggestions, and helpful comments on an earlier draft.

References

[1] Gentzen, G., Untersuchungen uber das logische Schliessen, Mathematische Zeitschrift, 39 (1934-5).

[2] Igarashi, S., An axiomatic approach to the equivalence problems of algorithms with applications, Reports of Computer Centre, University of Tokyo, 1, No. 1 (1968). Also distributed as: Publications of Research Institute for Mathematical Sciences, Kyoto University, 8, Nos. 33, 34, Kyoto (1969). (Appeard first as: Ph. D. Thesis, University of Tokyo, 1964.)

[3] Igarashi, S., Semantics of Algol-like statements, Symposium on Semantics of Algorithmic Languages, Engeler, E. (ed.), Lecture Notes in Mathematics, 188, Springer-Verlag (1971).

[4] McCarthy, J., On adding quantifiers to LCF, private communication, Stanford (1972).

[5] McCarthy, J. & Painter, J., Correctness of a compiler for arithmetic expressions, Proceedings of a Symposium in Applied Mathematics, 19, Schwartz, J. T. (ed.), American Mathematical Society (1967).

[6] Milner, R., Implementation and applications of Scott's logic for computable functions, Proceedings of a Conference on Proving Assetertions about Programs, New Mexico State University, SIGPLAN Notices 7 (1972).

[7] Milner, R., private communication, Stanford (1972).

[8] Milner, R. & Weyhrauch, R., Proving compiler correctness in a mechanized logic, Machine Intelligence 7, Michie, D. (ed.), Edinburgh, Edinburgh University Press (1972, to appear).

[9] Park, D., Fixpoint induction and proofs of program properties, Machine Intelligence, 5, Meltzer, B. & Michie, D. (eds.), Edinburgh, Edinburgh University Press (1970).

[10] Scott, D., private communication, Oxford (1969).

[11] Snoenfield, J. R., Mathematical Logic, Addison-Wesley Publ. Co. (1967).

[12] Wang, H., Logic of many-sorted theories, Journal of Symolic Logic, 17, No. 2 (1952).

[13] Weyhrauch, R. & Milner, R., Program semantics and correctness in a mechanized logic, Proceedings of USA-Japan Computer Conference, Tokyo, (1972, to appear).