数理解析研究所講究録
第 189 巻 1973 年 2-6

TWO FORMAL SYSTEMS FOR PROVING ASSERTIONS ABOUT PROGRAMS

Shigeru Igarashi (Kyoto University)

1. First-order logic of typed theories.

<u>1.1 Types</u>. α , β, γ .... denote types. <u>Ordered types</u> are denoted by αo, βo, etc.

a) We presuppose that there are finite number of <u>base types</u>.
b) α , β are types -----→ $\alpha\to\beta$ is a type.
c) $\alpha o \to \beta o$ is a type -----→ $(\alpha o \to \beta o)o$ is a type.

<u>1.2 Alphabet</u>.

α -constants
α -variables (for each α)

($\alpha 1$, ... , αn)-predicates
logical symbols:

$$(\quad , \quad) \quad \mathrm{Min} \quad = \quad \vee \quad \exists \quad \neg$$

<u>1.3 Terms</u>.

a) a is an α-constant -----→ a is an α -term.
b) x is an α-variable -----→ x is an α -term.
c) t is an $\alpha\to\beta$-term, u is an α-term -----→ t(u) is a β-term.
d) t is an ($\alpha o \to \alpha o$)-term -----→ Min t is an αo-term.

<u>1.4 Interpretation</u>.

Definitions.

$\aleph_0$-<u>inductively ordered set</u>. L is nonempty. Any countable linearly ordered subset (nonempty) X of L has sup X in L.

f: L → L' is <u>continuous</u> iff

$$f(\sup X) = \sup f(X) \qquad (1)$$

for any monotone increasing sequence X in L.

a) α is a base type that is not ordered -----→ $D\alpha$ is a non-empty set as the <u>domain of individuals</u>.
b) αo is an ordered base type -----→ $D\,\alpha o$ is an 0-inductively ordered set with the least element O.
c) $D(\alpha\to\beta)$ is the set of functions of $D\alpha$ into $D\beta$.
d) $D(\alpha o \to \beta o)o = \{ f \mid f:\ \text{continuous},\ f \in D(\alpha^0\to\beta^0) \}$.

t(u) denotes the <u>application</u> of t to u.

$$\mathrm{Min}\ f = \sup\{ f(O),\ ff(O),\ fff(O),\ \dots \} \quad . \qquad (2)$$

Logical axioms.

propositional axiom. $A \vee A.$

identity axiom. $x=x.$

equality axiom.

$$x=y \to \mathrm{Min}\ x = \mathrm{Min}\ y.$$
$$x=y \to z(x) = z(y).$$
$$x1=y1 \to \dots \to xn=yn \to p(x1, \dots, xn) \to p(y1, \dots, yn).$$

extensionality axiom. $x=y \equiv \forall z(x(z) = y(z)).$

stationariness axiom. $x(\mathrm{Min}\ x) = \mathrm{Min}\ x.$

induction axiom (fixed-point induction).

$$A[O] \to \forall y(A[y] \to A[x(y)]) \to A[\mathrm{Min}\ x].$$

2. Admissibility of fixed-point induction.

<u>Truth functions</u> are functions into the two element <u>complete lattice</u>.

<u>2.1 Hierarchy of admissibility</u>.

I. a.i.w.
II. a.i.s. (f(sup X)=limsup f(X))
III. weakly continuous. (f(sup X)=liminf f(X) = limsup f(X))
IV. continuous.
V. constant.

<u>2.2 Inheritance tables</u>.

$A \vee B$

A \ B	a.i.w.	a.i.s.	w.cont.	const.
a.i.w.	x*)	x	x	x
a.i.s.	x	a.i.s.	a.i.s.	a.i.s.
w.cont.	x	a.i.s.	w.cont.	w.cont.
const.	x	a.i.s.	w.cont.	const.

*) becomes a.i.w. in case of <u>conjunction</u>.

<u>2.3 Elementary formulas</u>.

Theorems. <u>Scott's awffs</u> of the form $t \leq u$ are a.i.s.
If $D\ \alpha o$ is <u>discrete (upward)</u> (No ascending chains that interpolate two elements of $D\ \alpha o$ exist.), then Scott's awffs of type αo are weakly continuous.

For A to be weakly continuous it is necessary and sufficient that A and $\neg A$ are a.i.s. (ETC.)

3. Formal system representing assertions for ALGOL-like statements.**)

3.1 Statements.

a) q is an (m,n)ary procedure symbol,
xl, ... ,xm are variables,
tl, ... , tn are terms in L(T)

$\longrightarrow$ q(xl, ... ,xm;tl, ... ,tn) is an atomic statement.

b) A, B are statements $\longrightarrow$ A;B is a statement.

c) A, B are statements, F is a quantifier-free formula in L(T) $\longrightarrow$ if F then A else B is a statement.

3.2 Assertions (wffs).

i) F { A } G. F, G are formulas in L(T), A is a statement.

ii) p(xl, ... ,xm;yl, ... ,yn) proc A.

iii) Formulas in L(T).

3.3 Axioms.

primitive procedures.

assignment axiom. R(f) { x ← f } R(x).

invariant axiom. R{ q(xl, ... ,xm;tl, ... ,tn) } R.

xl, ... , xm do not occur free in R.

defining axioms for procedures. Wffs of the form (ii) of 3.2.

logical axioms. Theorems belonging to the theory T.

3.4 Inference rules.

logical rules.

$$\frac{P \rightarrow Q \qquad Q\{A\}R}{P\{A\}R} \quad (1) \qquad\qquad \frac{P\{A\}R \qquad R \rightarrow S}{P\{A\}S} \quad (2)$$

$$\frac{P\{A\}R \qquad Q\{A\}S}{P \overset{\vee}{\wedge} Q\{A\}R \overset{\vee}{\wedge} S} \quad (3)$$

**) This is an exposition of the study by London, Luckham, and Igarashi.

substitution rules.

$$\frac{P(x)\ \{q(x;t(x))\}\ R(x)}{P(z)\ \{q(z;t(z))\}\ R(z).} \tag{4}$$

z denotes distinct variables which do not occur free in P(x), t(x), or in R(x).

$$\frac{P(y)\ \{\ q(x;t(y))\}\ R(y)}{P(u)\ \{\ q(x;t(u))\}\ R(u).} \tag{5}$$

x does not occur free in u.

recursion rule.

$$\frac{r(x;y)\ \underline{\text{proc}}\ K(r) \qquad \begin{matrix} [\ P\ \{q(x;y)\}\ R\ ] \\ P\ \{K(q)\ \}\ R \end{matrix}}{P\ \{\ r(x;y)\}\ R} . \tag{6}$$

q is a free procedure variable that does not occur free in any of the upper formulas except those places that are explicitly so indicated.

rules for constructors.

$$\frac{P\{\ A\}\ Q \qquad Q\ \{\ B\}\ R}{P\{\ A;B\}\ R} . \tag{7}$$

$$\frac{P\&F\{\ A\}\ R \qquad P\&\neg F\ \{B.\ \}\ R}{P\{\ \text{if}\ F\ \text{then}\ A\ \text{else}\ B\}R.} \tag{8}$$

3.5 Relatively sound rules.

$$\frac{P\&F\{A\ \}P}{P\{\ \text{while}\ F\ \text{do}\ A\ \}\ P\&\neg F.} \tag{9}$$

The following rule is a derived rule relative to (9).

$$\frac{P\&F\ \{\ A\}\ P \qquad P\&\neg F \to Q}{P\{\ \text{while}\ F\ \text{do}\ A\}\ Q} . \tag{10}$$

3.6 "Verification conditions"

A sufficient set of formulas in L(T) to prove $P\{A\}R$ is called a set of verification conditions for $P\{A\}R$. There is an algorithms to get this set from any given goal to be proved, which is a kind of <u>backward derivation</u> and similar to <u>parsing</u> procedures. A practical version of this algorithm has been implemented for PDP-10 of the Artificial Intelligence Project, Stanford University, by London, and has turned out to be extremely useful.

E.g.,

$$\mathrm{AH}(x\ \ f,\ R) = \mathrm{Subst}(R,\ x,\ f).$$

$$\mathrm{AH}(\text{if } F \text{ then } A \text{ else } B,\ R) = (F \rightarrow \mathrm{AH}(A,\ R))\ \&\ (\neg F \rightarrow \mathrm{AH}(B,R)).$$

$$\left.\begin{array}{l}\mathrm{AH}(A;B,\ R) = \\[1em] \mathrm{AH}\,p^*(\text{while } F \text{ do } B,\ R) = \end{array}\right\}\ \text{left to the reader.}$$

$$\mathrm{AH}(q(z;t),\ R) = \mathrm{Pre}(q)\ \&\ \forall x(\mathrm{Subst}(\mathrm{Res}(q),y,t) \rightarrow R).$$

(Cf. the <u>rule of adaptation</u> (Hoare))

$$\mathrm{VC}(P,\ A,\ R) = P \rightarrow \mathrm{AH}(A,\ R).$$

3.7 Consistency and strengthening the interpretation for proving termination.

These problems are being successfully studied. We have a consistency proof up to the recursion rule, and also a formal system for proving strong correctness (involving termination).