MORE NOTES ON APPLICATION OF COMPLETE POSETS TO RED LANGUAGES

Barry K Rosen
Computer Sciences Dept.
IBM Watson Research Center

10/28/74

We have a set Σ of operator symbols, and a dyadic operator ~~ap~~ is distinguished. The 8/1/74 notes dealt with the set E of finite expressions built up from these symbols (some of which are niladic) and the discrete complete poset $D = E \cup \{\perp\}$. At the end we noted that a fancier complete poset would be nice.

Algebraic notions used here are as in Goguen and Thatcher. For simplicity we assume there is only one sort of expression. When it is desired to have operators with restrictions like "the operands must be numerical" then we could use a many sorted algebra, with several sorts of expressions explicitly considered. This beats hell out of using partial functions to build up expressions. See Goguen and Thatcher.

~~For~~ Instead of D we consider the complete poset CT_Σ used by Goguen and Thatcher. Members of CT_Σ are (possibly infinite) trees with nodes marked by members of Σ. Precisely: a Σ-tree is a partial function from strings of nonnegative integers to Σ. Letting ω be the set of nonnegative integers and ω^* be the set of finite strings of them (empty string allowed), we require that a Σ-tree t satisfy, for all u in ω^* and all i in ω,

t defined at ui implies t defined at u;

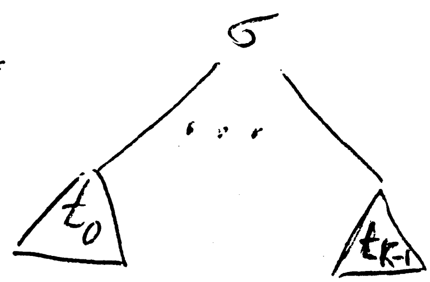
t defined at ui implies $i < K$ where the symbol tu in Σ is K-adic.

The set CT_Σ of all Σ-trees becomes a complete poset when ordered by

$s \le t$ iff t is an extension of s;

$\perp = $ (the empty partial fcn from ω^* into Σ).

It also ~~too~~ becomes a Σ-algebra in the natural way: if σ is K-adic and $t_0, \ldots, t_{K-1}$ are in CT_Σ, then

$\sigma(t_0, \ldots, t_{K-1}) = $ 

As a map $\sigma : (CT_\Sigma)^K \longrightarrow CT_\Sigma$, each σ in Σ is also continuous.

Some interesting subsets of CT_Σ:

. $\underset{\sim}{C}$ = (all t in CT_Σ such that $tu \ne \underline{ap}$ for all u in ω^*);

$\underset{\sim}{F}$ = (all t in CT_Σ with tu defined for just finitely many u);

$\underset{\sim}{M}$ = (all t in CT_Σ such that $t \lesssim s$ implies $t = s$ for all s in CT_Σ).

These are the ~~constat~~ constant, finite, and maximal members of CT_Σ.

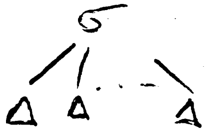
The set ~~CA~~ $\underset{\sim}{F} \cap \underset{\sim}{M}$ is a subalgebra of CT_Σ and is isomorphic to the old algebra ~~B~~ E. Thus we can think of CT_Σ as an elaboration of D in which we no longer insist ~~upo~~ upon $\sigma(\ldots,\perp,\ldots) = \perp$. That CT_Σ is an especially well motivated elaboration is shown by the many nice properties established by Goguen and Thatcher. The various "initiality results" allow many arguements analogous to structural inductions to be used for infinite as well as finite ~~experie~~ trees.

In order to introduce the μ function without the discreteness crutch used before, we notice that syntactic decompositions and tests on Σ-trees can be performed with continuous functions.

Let Σ^+ be the discrete complete poset $\Sigma \cup \{\text{NOOP}\}$, with $\text{NOOP} \leq \sigma$, all σ in Σ, and $\sigma \leq \tau$ iff $\sigma = \tau$. Selecting out the operator at the root of a tree is a continuous map:

$$\text{OPER} : CT_\Sigma \longrightarrow \Sigma^+$$

$$\text{OPER}(\perp) = \text{NOOP};$$

OPER() = σ.

Selecting the k-th argument (counting from 0) to the root of a tree is also a continuous map:

$$ARG_k : CT_\Sigma \longrightarrow CT_\Sigma ;$$

$$ARG_k\left(\begin{array}{c} \sigma \\ \diagup \cdots \diagdown \\ t_0 \qquad t_{K-1} \end{array} \right) = t_k \quad \text{if } k < K ;$$

$$ARG_k(s) = \perp \quad \text{otherwise.}$$

(We could equally well let ERROR be niladic in Σ and set (for σ in Σ)

$$ARG_k\left(\begin{array}{c} \sigma \\ \diagup \ \diagdown \\ t_0 \cdots t_{J-1} \end{array} \right) = \text{ERROR for } J \leq k.)$$

A trickier decomposition ARGLIST maps trees to finite strings of trees. Let SCT_Σ be the set CT_Σ^* together with a special element NOLIST which is NOT the empty string. We get a complete poset by saying that all x, y in CT_Σ^* have

$$x \le y \quad \text{iff} \quad x, y \text{ have the same length and } x_i \le y_i \text{ for all } i$$

and that

$$\mathrm{NOLIST} \le x, \text{ all } x \text{ in } CT_\Sigma^*.$$

The empty string (and any other string with only maximal trees in it) is maximal in SCT_Σ. Now we have

$$\mathrm{ARGLIST} : CT_\Sigma \to SCT_\Sigma;$$

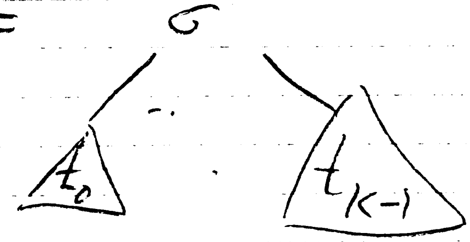
$$\mathrm{ARGLIST}\left(\begin{array}{c}\sigma\\ / \dots \backslash\\ t_0 \qquad t_{k-1}\end{array}\right) = (t_0, \dots, t_{k-1}), \text{ a ~~st~~ string } k \text{ long in } CT_\Sigma^*;$$

$$\mathrm{ARGLIST}(\bot) = \mathrm{NOLIST}.$$

Our last syntactic operation builds up rather than tears down:

$\cancel{\Sigma}$ $MAKE: \Sigma^{+} \times SCT_{\Sigma} \longrightarrow CT_{\Sigma}$;

$MAKE(\sigma, (t_0, \ldots, t_{K-1})) =$



if σ is K-adic in Σ

$MAKE(\alpha, z) = \perp$ otherwise.

(For $\alpha \neq NOOP$ and $z \neq NOLIST$ but of the wrong length for α, we could use $MAKE(\alpha, z) = ERROR$ instead.)

The construction which passes from CT_{Σ} to SCT_{Σ} also lets us pass from ~~Σ~~ $[CT_{\Sigma} \to CT_{\Sigma}]$ to $[SCT_{\Sigma} \to SCT_{\Sigma}]$ by the usual extension of scalar functions to handle vector arguments: Given $F: CT_{\Sigma} \to CT_{\Sigma}$ continuous, we get $F^{*}: SCT_{\Sigma} \longrightarrow SCT_{\Sigma}$ continuous by letting

$$F^* \; NOLIST = NOLIST;$$

$$F^*(t_0,\dots,t_{K-1}) = (Ft_0,\dots,Ft_{K-1}).$$

The mapping from F to F^* is also continuous.

The conditional maps <u>cond</u> from p. 5 of the 8/1/74 notes and the evaluation maps from p. 2 of the 8/1/74 notes are also continuous. Because compositions of continuous maps are continuous and a function of several variables is continuous in each iff continuous in all (<u>ie.</u> p. 3 of 8/1/74 notes), we know that any

$$\rho_* : CT_\Sigma \longrightarrow [CT_\Sigma \rightarrow CT_\Sigma]$$

continuous leads to a continuous functional

$$\underset{\sim}{T} : ([CT_\Sigma \rightarrow CT_\Sigma] \times [CT_\Sigma \rightarrow CT_\Sigma]) \longrightarrow [CT_\Sigma \rightarrow CT_\Sigma]$$

with

$\underset{\sim}{T}(F, G)\, t =$

$$\underline{\text{cond}}(\ \text{OPER}(t) = \underline{ap},\ G[(\rho_* F(\text{ARG}_0 t))\ F(\text{ARG}_1 t)],\ \text{MAKE}(\text{OPER}(t),\ F^* \text{ARGLIST}\ t)).$$

Compare $\underset{\sim}{T}$ with T on p. 9 of the 8/1/74 notes!

Let μ_* be the least fixpoint of the map from F to $\underset{\sim}{T}(F, F)$, so that

$$\mu_* : CT_\Sigma \longrightarrow CT_\Sigma \text{ continuous}$$

and μ_* has the main properties desired of a meaning function.

To generalize the elementary properties of μ on pp 14-15 of the 8/1/74 notes, we proceed as follows.

LEMMA 1: All t in CT_Σ have $\mu_* t$ in $\underset{\sim}{C}$.

PROOF: $\mu_* = \text{lub}\, \{\mu_n \mid n = 0, 1, \ldots\}$ where

μ_0 is $\perp$ in $[CT_\Sigma \to CT_\Sigma]$ and

$\mu_{n+1} = \underset{\sim}{T}(\mu_n, \mu_n)$. By induction on n, each μ_n has $\mu_n t$ in $\underset{\sim}{C}$ for all t. The lub of a set of trees in $\underset{\sim}{C}$ which has a lub in CT_Σ will also be in $\underset{\sim}{C}$, so

$$\mu_* t = \text{lub}\{\mu_n t \mid n = 0, 1, \ldots\}$$

and is in $\underset{\sim}{C}$. ∎

LEMMA 2: All t in $\underset{\sim}{C}$ have $\mu_* t = t$.

PROOF: By structural induction, all t in $\underset{\sim}{C} \cap \underset{\sim}{F}$ have $\mu_* t = t$. For any t in CT_Σ, let t_n be the restriction of t to strings of length n or less in ω^*. Then $t_n \in \underset{\sim}{C} \cap \underset{\sim}{F}$ if $t \in \underset{\sim}{C}$, and

$$t = \text{lub}\{t_n \mid n = 0, 1, \ldots\}$$

Therefore

$$\mu_* t = \text{lub}\{\mu_* t_n \mid n = 0, 1, \ldots\} = t \text{ , all } t \text{ in } \underset{\sim}{C}. \ ∎$$

LEMMA 3. $\mu_*\mu_* = \mu_*$ & $\underset{\sim}{C} = \{t \mid \mu_* t = t\}$.

PROOF: By Lemmas 1 and 2. ∎

BUT HOW ARE μ_* VALUES TO BE COMPUTED IN THE REAL WORLD? So far we only required that

$$\rho_*: CT_\Sigma \longrightarrow [CT_\Sigma \rightarrow CT_\Sigma]$$

be continuous. We have not yet considered how ρ_* could be effectively determined by its values on members of $\underset{\sim}{C}$. We want to stay fairly close to the old ~~style~~ style ρ but to let ρHEAD and the like be handled as well.

✗ MORE TO COME ✗