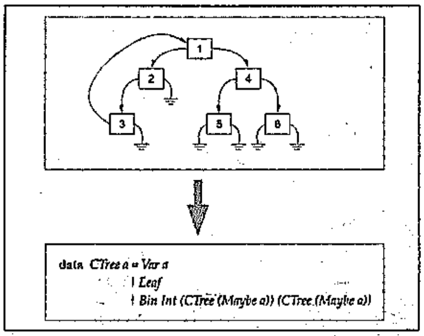
群大工学部 研究室から ㉜

ソフトウエアを科学する

理論計算機科学

正しく、よりよいものを作るため 現代にこそ必要な「原理」の研究



サイクルを持つような木構造データとその型表現。一見複雑な構造の簡明な表現が見つかれば、ソフトウエアの構築に役立ちます

情報工学科 情報数理工学講座第3研究室

私たちは現在、コンピューターが社会のどこにでもあって、コンピューターがあることさえも特別に意識しなくてもよい社会に生きています。多くの人がパソコンを持ち、パソコンのソフトウエアを使って仕事をしています。しかし、そのソフトウエアで行われている多種多様な計算がどのような原理に基づいて行われているのか、あなたは知っているでしょうか？

いま私たちが使っているコンピューターの計算の原理の発見は1930年代から40年代にかけてなされました。この発見は今私たちが知っている形のコンピューターが出現する以前のことです。当時活躍したゲーデル、チューリング、フォン・ノイマンといった天才的な科学者たちは、私たちが漠然ととらえている計算のすべてを矛盾なく行える厳密な体系を構築するとともに、計算の限界を明らかにすることに興味を持っていました。そのような理論的研究から現代のコンピューターが誕生したのです。

私の研究は、この30年代からの流れを継いだソフトウエアの基本原理を研究する理論計算機科学という分野です。しかし現代十分に強力になったコンピューターに対して、そのような古めかしい「原理」の研究などがまだ必要なのでしょうか？

実は今こそ、このようなソフトウエアの原理の研究が必要なのです。こう言うと驚かれるかもしれませんが、どんな先進的なソフトウエア開発者も、また大学で研究するコンピューターの専門家でも、ソフトウエアというものが「何なのか」いまだ正確には理解できていないのです。

もちろんソフトウエアを「つくる」ことはできます。しかし、どのようにつくれば一番良いのか、つくったものが正しいものなのか、その判断基準は何か。そんな大事な問題に対して現在は「経験的にまあよさそう」といった段階の答えしか分かっていません。ソフトウエアにかかわる人々はそのような不満足な土台の上でなんとか力を尽くしてやりくりをしているのです。幾度となく報道される証券会社や銀行のシステムの不具合でわかるように、最も正確でなければならないソフトウエアですらいまだに間違いが起こってしまうことがこの状況をよく表しています。

さまざまな角度から「計算のモデル」解釈し

では、正しく、よりよいソフトウエアのためにはどのような原理が必要なのでしょうか。私はこの問題に対して数理論理学、形式的体系、圏論的代数という数理的な道具を用いて「計算のモデル」をさまざまな角度から解釈し直し、より正しい原理の発見を目指して研究をしています。この観点からこれまでに、新しい論理型プログラミング言語、意味モデルに基づく計算規則の正しさの判定技術、サイクルを持つデータの「型」表現といったものを研究しました。最近はこれらの基礎となる計算モデルの入門書なども書いたりしました。

私の研究方法は、机上で理論を構築し、それをコンピューター上で実現しながらその有効性を確かめ、結果を理論にフィードバックし、より完全なものに近づけるものです。理論計算機科学は、伝統的な学問と新しい情報技術が混在する刺激的な分野だと思っています。

浜名誠助手

〈私が執筆しました〉

浜名誠 助手

【プロフィル】東京都出身。1971年1月生まれ。98年筑波大学工学研究科電子情報工学専攻博士課程修了、工学博士。京都大学で日本学術振興会特別研究員、英国エジンバラ大学計算機科学基礎研究所客員研究員を経て、2001年から群馬大学工学部情報工学科助手。

ソフトウエアを動作させる計算の数学的モデルをわかりやすく解説した近著

```
foldT :: (forall a. a -> f a) ->
         (forall a . f a) ->
         (forall a. Int -> f (Maybe a) -> f (Maybe a) -> f a) ->
         CTree a -> f a
foldT v l n (PtrT x)        = v x
foldT v l n Leaf            = l
foldT v l n (Node x t1 t2) = n x (foldT v l n t1) (foldT v l n t2)

foldTm :: (forall a. Maybe a -> f a) ->
          (forall a . f a) ->
          (forall a. Int -> f (Maybe a) -> f (Maybe a) -> f a) ->
          CTree (Maybe a) -> f a
foldTm v l n (PtrT x)        = v x
foldTm v l n Leaf            = l
foldTm v l n (Node x t1 t2) = n x (foldTm v l n t1) (foldTm v l n t2)

ctree1 = Node 1 Leaf (Node 2 Leaf Leaf)
ctree2 = Node 1 (PtrT Nothing) Leaf
ctree3 = Node 1 (Node 2 (PtrT Nothing) Leaf) (Node 3 Leaf (PtrT Nothing))
ctree4 = Node 1 (Node 2 (Node 3 (PtrT Nothing) Leaf)
                        Leaf)
                (Node 4 (Node 5 Leaf Leaf)
                        (Node 6 Leaf Leaf))
```

研究には現在注目を集めているHaskellという先進的なプログラミング言語を用います

Theorem 21. *For a CRS $(\mathcal{R}, Z)$, $(M_\Sigma Z, \rightsquigarrow^+_{\mathcal{R}})$ is a free $(\Sigma, \mathcal{R})$-monoid over Z, i.e. for any admissible assignment ϕ from Z into a $(\Sigma, \mathcal{R})$-monoid $(A, >_A)$, there exists a unique Σ-monoid map ϕ^* that is monotone and makes the right diagram commute in $\mathbf{Set}^{\mathbb{F}}$, where $\eta_Z : z^l \longmapsto z[1, \ldots, l]$.*

$$\begin{tikzcd} Z \arrow[r, "\eta_Z"] \arrow[dr, "\phi"'] & M_\Sigma Z \arrow[d, "\phi^*"] \\ & A \end{tikzcd}$$

Proof. Let $\phi : Z \longrightarrow A$ be an admissible assignment into a $(\Sigma, \mathcal{R})$-monoid $(A, >_A)$. Since $M_\Sigma Z$ is a free Σ-monoid [Ham04], ϕ^* is a unique Σ-monoid morphism that makes the above diagram commute in $\mathrm{Set}^{\mathbb{F}}$. So, the remaining task is to show ϕ^* is monotone. This is proved by induction on the structure of inference of $\rightsquigarrow_{\mathcal{R}}$ and the length of $\rightsquigarrow^+_{\mathcal{R}}$. The case for instantiation of a rewrite rule, we use $(\phi^* \circ \theta)^* = \phi^* \circ \theta^*$, which is proved by induction on meta-terms. The crucial case is to show ϕ^* preserves the relation of $z[\ldots, s, \ldots] \rightsquigarrow_{\mathcal{R}} z[\ldots, t, \ldots]$. This holds because we have assumed that ϕ is admissible. $\square$

Theorem 22. *A CRS $(\mathcal{R}, Z)$ is meta-terminating if and only if there is a well-founded $(\Sigma, \mathcal{R})$-monoid.*

構築した理論の一部