7G-8

T式による関数型プログラム開発

飯島正，　岡田謙一，　横山光男，　北川節
（慶応義塾大学）

1．はじめに

図形的なプログラム表現を利用するプログラム設計支援の一方式に、命令型（imparative）プログラムに直結する木構造制御流れ図を用いた段階的詳細化の支援があり、既に実用化の域に達している[2][3]。しかし、木構造制御流れ図は、処理の時系列という命令型スタイルを受け継いでいるため、データの流れ・処理と処理の間の関係が不明確となりがちであり、仕様の変更の波及する範囲の把握も困難である。しかも、プログラムの導出履歴を複数の抽象レベルとして呈示する能力が要求されているにもかかわらず、その構造がプログラム言語レベルの構造化制御構文（連接・反復・選択）に基づくブロック構造に対応しているため、柔軟性に欠ける傾向にある。本論文は、これらの問題を克服する一方策として関数型（functional/applicative）プログラミング・スタイルに着目し、図的表現：T式[1]を利用した関数型プログラム開発法について報告するものである。

関数型プログラム開発法は、特に目新しい手法を含んでいるわけではない。関数型プログラミング・スタイルの枠組みの中で自然に取り扱うことのできる、既存の各種プログラム開発技法を総称して関数型プログラム開発法と呼んでいる。そのため、それが関数型プログラミングと相性が良いのは当然といえるが、構造化された命令型プログラミングへの応用も比較的容易である。基本方針は、段階的詳細化による機能分割の際に、パラメータ化を伴うことによってデータの依存関係の明確化を図り、入出力表明を明らかにしておくことによってモジュール間の不整合の発生を抑えることである。

2．関数型プログラムのための図的表現：T式

T式（Tree structured Symbolic expression；木構造化記号表現）とは、関数型プログラムの可読性・直感性と操作性の向上を目的として、文字を単位とするプログラム表現、例えば、S式（Symbolic expression；記号表現）やM式（Meta expression）にとって代わるべく提案された図的プログラム表現法である[1]。その構造はλ－記法に対応している（図1）。T式は、"適用"によって形成される関数型プログラムの基本構造、すなわち、関数（非終端ノード）とデータ（葉）から成る多分木構造をそのまま図面として表示する。これによって、従来のインデント記法以上の高い可読性が得られ、多値関数等の表記にも有効性を示した。また、現在開発中のT式エディタを用いることで高い操作性（編集時に構造の破壊を招かず、また編集単位である部分木の指定が容易）が得られる。仕様の記述法としてのT式は、"λ－抽象"によって静的な機能の階層関係を表現することができ、更に、並行最内代入規則（parallel-innermost rule）を仮定し作用的な見方をすると、"適用"によって動的なデータの依存関係を表すことができる（図2；図中、aはbとcに依存し、cはdとeに依存する。よって、bとcおよびdとeは、それぞれ並行して評価することができる。但し、aの評価は、その引数であるbとcの値な確定後に始められる。つまり、bとcおよびdとeの評価は、それぞれそれらに適用する関数aとcの評価に先行して、並列にすすめられる。）。このような仮定は、実用上、特に問題が無く、プログラム設計者にとって自然に受け入れられるものである。

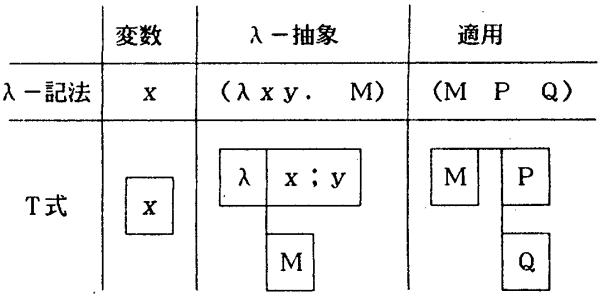

図1．λ－記法とT式の対応
（x，y：変数；　M，P，Q：λ－式，T式）

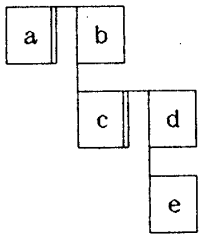

図2．データ依存関係

Functional/Applicative program development methods using T-expression
Tadashi Iijima, Ken-ichi Okada, Teruo Yokoyama, Misao Kitagawa
KEIO University

3．関数型プログラム開発法

本来、プログラム設計・開発技法はプログラム言語に依存するものではないが、関数型言語の利用者社会にはあまり普及せず、そこでは代わりにソフトウェア・ツール（これは、規範・指針ではなく、便利な道具に過ぎない）の開発がすすめられてきた。その理由として、命令型言語のために開発されてきた設計・開発技法の多くが、関数型プログラミング・スタイル自身に内包されているために、問題仕様分析・形式化の規範としてのそれらが不必要であったことが考えられる。むしろ、それらの設計・開発技法は、1)モジュール性・局所性（１入口／１出口）、2)参照の透明性（モジュール間インターフェースの明解性）、3)階層モジュール構造による複数抽象レベルの混在、といった関数型プログラムのもつ特性を命令型プログラミングに付与しようとするものとさえいえる。

これらの特性に支持されるプログラム設計法に機能分割（functional decomposition）と入出力仕様記述がある。機能分割は更に、手続き抽象化である段階的詳細化（stepwise refinement）とデータ抽象化の二通りに分けられる。前者は要求からのトップダウン方式、後者はデータ型からのボトムアップ方式といえる。

段階的詳細化は、既に木構造制御流れ図による命令型スタイルの支援がすすめられている。Ｔ式による関数型スタイルの支援でも、”λ－抽象”化に従って次第に詳細化されていき、その過程がＴ式上に記録される点は、基本的には同じである。しかし、Ｔ式の場合には、更に、”適用”によって、データの依存関係も記録される（図２参照）。これらの機能の階層関係とデータの依存関係の記録は、仕様の変更が影響する範囲を特定するのに有効であり、設計時の思考のフィードバックのためばかりでなく、プログラムのデバッグ時・再利用時に利用される。また、命令型スタイルの場合には、要求する結果から出発して入力データへ至る方向（前進過程）のみが許されるのに対し、関数型スタイルではパラメータ化によって、入力データを変形していって結果へ至る方向（後退過程）も許されており、両者を混在させることもできる。この両方向性は、定理の証明（仮定と結論）や迷路の脱出（入口と出口）といった典型的な問題解決にも活用される。思考の道具としては、これらの両方向共に再試行（バックトラック）が容易にできるような支援が必要であると考えられる。データ抽象化に対するＴ式並びに関数型プログラミング・スタイルの貢献度は、Ｔ式によってデータ型の間の関係を整理して示すことができる程度であり、比較的低い。逆に、抽象データ型の蓄積が関数型スタイルにもたらす利点は大きい。適切な抽象データ型が用意されていることで、実現部の詳細記述が省け、関数型プログラムの持つラピッドプロトタイピングあるいは実行可能仕様（executable specification）といった側面が強調されてくるからである。

入出力仕様記述が関数型プログラミングに適している事も当然といえる。命令型プログラミングで処理と処理の間の関係を示す表明は、関数型プログラミングでは関数の入出力表明となる。Ｔ式では、全ての”λ－抽象”に入出力表明を与えることができる。関数の入力表明と引数の出力表明の間の矛盾の有無は自動的にチェックされる。

4．命令型プログラミングへの影響

命令型プログラムは、処理の時系列である。これをＴ式の関数型プログラミング・スタイルで扱うために、goto文やループからの脱出命令を持たない命令型言語を仮定する。すると、反復は帰納的定義で、選択は条件式で記述できる。そして、連接（ａ；ｂ；ｃ）は、”λ－抽象”と”適用”によって表すことができるが、図３に示す通り一意には決まらない。これは、命令型スタイルでは、データの依存関係の情報が失われてしまうためである。逆に、連接を図３のどれか適切なもので表現しておけば、その情報は保存される。設計時に、木構造制御流れ図の代わりにＴ式を用いることによって、この効果が得られる。

5．おわりに

現在、プログラム設計を支援するＴ式エディタを試作中である。今後は、プログラム開発の全過程におけるインターフェースとしてのＴ式の可能性を追求していくつもりでいる。Ｔ式が基礎とするλ－記法は、開発のためのインターフェースとしては有効であるが、実行や効率化変形・正当性の検証の自動化には不利なので、まず、それらに有利な”変数無しプログラム”（ＦＰ，コンビネータ）へのＴ式の枠内での変換系の開発を予定している。また、ここでは取り上げなかったが、命令型プログラミングにとって標準的なプログラム開発法としてジャクソン法がある。これは、入出力データ構造に沿って機能分割する手法ということができ、データ構造を意識しながら前述の前進・後退過程を繰り返す事によって実現できると思われる。より積極的に支援する方法を検討中である。

参考文献

[1]飯島，岡田，横山，北川：”Ｔ式：多分木構造による関数型プログラムの図的表現”，情報処理学会第３２回（昭和６１年前期）全国大会，pp.537-538
[2]二村：プログラム設計法ＰＡＤ／ＰＡＭ，情報処理，Vol.24,No.11,pp.1237-1246(1984)
[3]花田，佐藤，松本，長野：コンパクト・チャートを用いたプログラム設計法，情報処理学会論文誌，Vol.22,No.1,pp.44-50(1981)
[4]井田，田中：関数型言語の計算モデル，コンピュータソフトウェア（日本ソフトウェア科学会誌），Vol.3,No.2,pp.2-18(1986)
[5]紫合：ソフトウェア設計法について，コンピュータソフトウェア（日本ソフトウェア科学会誌），Vol.1,No.2,pp.55-68(1984)

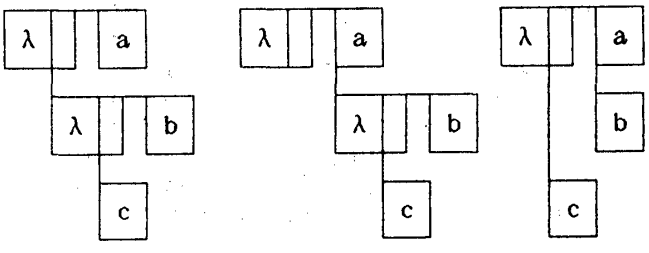

図３．連接（ａ；ｂ；ｃ）のＴ式による記述例