ソフトウェア工学 77−10
（1991. 2. 7）

CASE指向抽象ソフトウェアモデルII
― データ構造の融合 ―

松本憲幸　湯原義彦

(株)東芝

プログラミング言語により具現化される以前の抽象段階において、ソフトウェアの構造や振る舞いの統一的な記述を想定した記号モデルについて説明する。

本論文では、リアルタイム構造化分析情報を統合した従来の素過程モデルの記述内容をその基本概念となる有限オートマトンとの対応関係から補正した後、情報の構造を表す制約を課した素過程を導入することにより、従来と親和性の高い形式でデータ構造の記述を導入し、その記述内容とジャクソン手法やワーニエ-オー手法によるデータ設計との対応を可能とする。

Abstract Software model for CASE (II)
― data structure introduction ―

Noriyoshi Matsumoto　Yoshihiko Yuhara

TOSHIBA Corporation

1 Toshiba-cho, Fuchu-Shi, Tokyo 183, Japan

An abstract software model is introduced to describe the primary software definition in Upper CASE domain. It extends our old symbolic model which synthesized the information of the real-time structured analysis to describe the data structure information. The model is primarily interpreted in the framework of the finite automata theory but has an additional term to define the process structure in according to the structured analysis and can be easily associated with the software design methodologies.

In this paper the nondeterministic finite automata interpretation of the model and the extension for the data structure description are presented.

1. はじめに

プログラムとして実体化される以前の抽象段階におけるソフトウェア設計には、構造化分析・構造化設計[1],[2],[3]などの手法が存在する。これらの手法で作成される各種特性図に分散した設計情報を統合しつつ、ダイアグラム表現から視覚的な曖昧さを除き、簡潔な記号イメージでの解析およびそのコンピュータサポートを想定したモデルを提案する。

ここでは、リアルタイム構造化分析(rtsa)情報を統合した従来の素過程モデル[4],[5]を、その基本概念となっている有限状態マシンの定義情報と解釈して非決定的有限オートマトン(NDFA)との対応を可能としてその記述範囲をより明確化し、さらに素過程モデルに対してデータ構造定義情報の融合を行う。

2. 素過程モデルの記述範囲

Hatleyのrtsaは決定的有限オートマトン(DFA)に基づくが、これはNDFAで多重遷移やλ(またはε)遷移を禁止することで記述可能であり、ここでは、より抽象度の高い段階からのサポートを想定してNDFAとの対応をとる。

2.1 素過程によるNDFA記述

素過程に含まれる情報は、NDFAの定義情報と比較した場合、次の点が欠落している。

・初期状態/終了状態の定義。
・入力データ(広義のアルファベット)の定義。

この中で、入力データ定義はデータ構造定義の問題として後述し、ここでは非決定的遷移の記述に関して説明する。

あるマシンAの状態遷移は、一連の入力列の入力開始及び入力終了の間で定義され、入力開始以前及び入力終了後Aは停止状態(ここではφで表す)となる。これを利用して初期/終了状態の指定を行い、状態遷移過程の非決定的解釈を導入することにより、NDFAを素過程の集合として記述することができる(例1)。

2.1.1 初期状態

初期状態Siは、最初の入力を受付ける時点でマシンAが占めるべき状態(群)で、これを停止状態から通常の状態への遷移をともなう素過程によって(2.1)の形式で表す。ここでは、入力列の先頭に入力開始の特殊な信号を仮定せず、λ遷移と見なしている。

<λ |A(φ →S)|>.・・(2.1)

または <|A(φ →S)|>.

この初期化過程が定義されている状態Sの群をマシンAの初期状態と見なせばよい。

2.1.2 終了状態

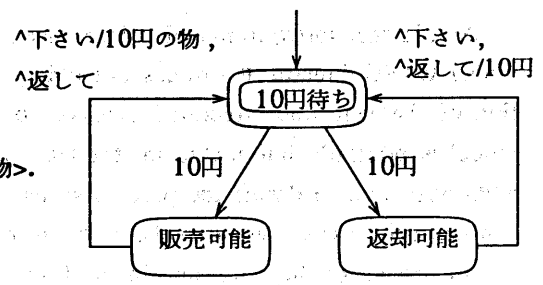
終了状態Sf は、入力文字列の終端 End-Of-String(EOS)を受け付けることが正当に定義された状態と見なすことができる。これを次の素

<λ | 販売機(φ →10円待ち) |>.
<EOS| 販売機(10円待ち→ φ) |>.

<10円| 販売機(10円待ち→販売可能) |>.
<10円| 販売機(10円待ち→返却可能) |>.

<^下さい| 販売機(販売可能→10円待ち) |10円の物>.
<^下さい| 販売機(返却可能→10円待ち) |>.

<^返して| 販売機(販売可能→10円待ち) |>.
<^返して| 販売機(返却可能→10円待ち) |10円>.



例1. 素過程モデルによるNDFA記述と対応する状態遷移図

過程で記述する。

<EOS|A(S→ φ)|>.・・(2.2)

この終了過程が定義されている状態Sの群を終了状態と見なせばよい。ここで、EOSは逐次的な入力の終りではなく、活性化されたマシンの状態の放棄を表す。

2.1.3 非決定的な遷移解釈

NDFA解釈では、ある入力に対して複数の素過程が応答可能である。例えば、入力aに対して以下の遷移過程群が定義されているとする。

<a|A(S→ T1)|>, ... , <a|A(S→ Tn)|>

この時、Aが占める状態の1つにSが含まれる場合、状態は T1, ... ,Tn の複合状態に変化する(Aの最初の状態にS以外が含まれれば、そこからの遷移で生じる複合状態との和となる)。つまり、入力に応答する素過程が複数存在する場合、各過程は逐次的でなく並列的に解釈される。一方、応答する素過程が唯一の場合、NDFA解釈はDFAの決定的遷移と一致する。

2.2 素過程によるDFA記述

並行した状態過程を離散的に記述するNDFA形式の定義をそのまま右辺記述により構造化しても、並行した素過程間の因果関係が明確でない点で、これをDFA形式で統合記述する場合に比べて不完全と成り得る。一方、構造化分析ではデータ/コントロールフローにより情報の因果関係の記述が要求されるため、正確にrtsa情報を記述するにはNDFA形式による定義をDFA形式に変換する必要がある。

この変換で統合されたDFA形式の素過程の取り得る状態は、一般に最初に定義したNDFAの状態の組合せのうち、初期状態からの遷移で辿り着けるものとなる。この各状態をもとのNDFAの状態の集合({S1, .., Si}等)で表すと、例1のNDFA形式は例2のDFA形式で記述される。但し、初期化のλ遷移を除去すると初期状態が識別不能となるため残されている。

3. データ構造定義

素過程モデルは情報間の動的な結合関係を表すため、この記述範囲では情報の静的構造を表せない。故に、新たに「与えられた情報の静的構造を逐次的に辿る」という制約を加えた特殊な素過程として構文解析過程(パーサ)を導入し、従来の概念と親和性をもった形式でデータ構造を記述する。

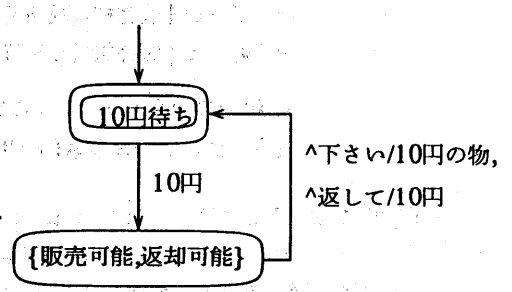
3.1 BNFパーサの導入

制約を加えたパーサの構造を、制約なしの従来の素過程の構造と識別して「素過程 ::= 複合過程 .」の形式で記述することにする。この時、制約を加えられたパーサとして、入力から自身の認識すべき構造を除去したものを出力するような素過程を考えると、情報 p の静的構造が qおよびrから構成される場合のパーサの処理は、

<x|parser-p|x-p> ::=
<x|parser-q|x-q><x-q|parser-r|x-q-r>.
ただし、|x-p> = |x-q-r> ・・・(3.1)

<λ|販売機(φ→10円待ち)|>.
<EOS|販売機(10円待ち→φ)|>.

<10円|販売機(10円待ち→{販売可能,返却可能})|>.
<^下さい|販売機({販売可能,返却可能}→10円待ち)|10円の物>.
<^返して|販売機({販売可能,返却可能}→10円待ち)|10円>.



例2. 素過程モデルによるDFA記述(例1の決定的バージョン。但し、不正行為の遷移過程は割愛している)

ここで、次の単純な2つの規約を設定すると、この記述形式は一般性を失うことなくBackus-Naur Form(BNF)と一致する。

・ある名称の情報は同一名のパーサで解析する。(parser-p → p 等の名称変更を行う)

・すべてのパーサは、入力から「その名称により表される情報構造」を除いて出力する形式を取り、この時入出力記述を省略可能とする。(<p> = <入力 |p| 入力-p>が成り立つ)

また、並列的に結合された素過程過程によって表現されるパーサ(<p> ::= <q>;<r>.)の処理に「排他的並列性」の制約を課し、これを「<p> := <q> | <r>.」と表すと、導入したパーサを完全にBNFで置き換えられる。

BNFパーサと素過程モデルの関係は、context free言語(cfl)と有限オートマトン(FA)の関係から考えると一見誤って見えるが、これは素過程モデルとFAとではBNF解析方法に次の相違があることに起因している。

・FAはBNF構文を自分自身の内部状態変化で記憶するが、素過程モデルは構文を記憶した右辺(FAに無い記憶構造)から各非終端記号に対応するパーサ呼出しによって構文を解釈する。

もっとも、モデルのBNF定義が制約を課された特殊な素過程に対応することは、コンピュータサポート時の解釈に関するもので、実際のデータ定義に際して意識する必要はない(例3)。

BNFパーサの導入に際して、モデルではパーサの存在に次の意味付けを行う。

・同一名のパーサ定義をもつ情報はその文法に従った言語を値とする。(非終端記号に相当)

・同一名のパーサをもたない情報は、単にその名称を値とする。(終端記号に相当)

これにより従来定義不能であったFA解釈における入出力アルファベットΣ={a, b, c }等は、

$$<\Sigma> ::= a \mid b \mid c . \quad \cdots (3.2)$$

という表現で定義可能となる。

3.2 BNFパーサの記述範囲と形式

BNFは記述内容を context free 構造に限定するが、ジャクソンやワーニエ-オーなどの各種設計法[6]は、あるデータの構造が文脈に依存して変化するような形式はとっておらず、この記述能力の範囲内で対応を取ることができる。

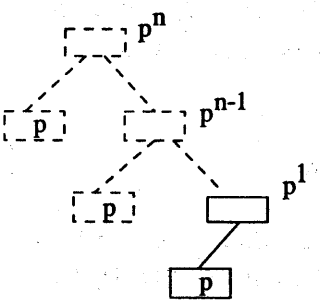
一方、BNFでは繰り返し構造を記述する場合に再帰構造(及びこれを体現する余計な非終端記号)を必要とし、この点で階層構造を不格好にす

```
< 10円 | 販売機(10円待ち→{販売可能, 返却可能}) | >.
< ^下さい | 販売機({販売可能, 返却可能}→10円待ち) | 10円の物>.
< ^返して | 販売機({販売可能, 返却可能}→10円待ち) | 10円>.

< 10円の物 > ::= ねこ | こま | まり | りす | すずめ | めがね .
< ^下さい > ::= 要求( <10円の物> ).
```

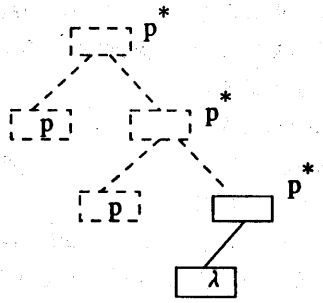
- 「10円の物」としては、「ねこ, こま, まり, りす, すずめ, めがね」のいずれかのシンボルが値となる。
- 「^下さい」というコントロールは、実際には「要求(ねこ)」,「要求(こま)」等の形式をとる。(実はこれは情報を運ぶという点でデータフローである)

例3. 入出力情報の定義を含んだモデル記述



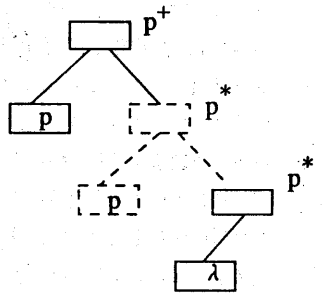
<p>^{1} ::= <p>.
<p>^{n} ::= <p> <p>$^{n-1}$.
n= 2, 3, ...

(4.1) n 回繰り返し



<p>* ::= λ | <p> <p>*.

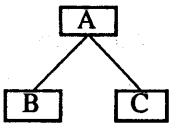
(4.2) 0 回以上繰り返し



<p>$^{+}$ ::= <p> <p>*.

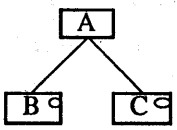
(4.3) 1 回以上繰り返し

例4 繰り返し構造の標準記述形式



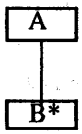
<A> ::= <B> <C>.

(5.1) 順次



<A> ::= <B> | <C>

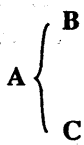
(5.2) 選択



<A> ::= <B>*.

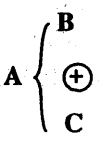
(5.3) 繰り返し

例5 ジャクソン手法のデータ構造とBNF記述の対応



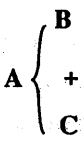
<A> ::= <B> <C>.

(6.1) 順次



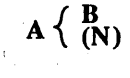
<A> ::= <B> | <C>.

(6.2a) 排他的選択

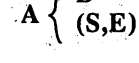


<A> ::= <B> | <C> |
<B><C>.

(6.2b) 包含的選択



<A> ::= <B>^{N}.



<A> ::= <B>$^{(S,E)}$.
<S> ::= 0 | 1.

(6.3) 繰り返し

例6 ワーニエ-オー手法のデータ構造とBNF記述の対応

る。ここでは、パーサPに対する繰り返し構造の標準記述形式(例4)を用意し、その詳細構造表現を不要とすることにより、見かけ上問題を回避する(ジャクソン設計法等おける繰り返し構造の*記号表現と同じで効果と考えれば良い)。

例5にジャクソン手法で用いられるデータ構造とBNFの対応を示す。また、ワーニエ-オー手法では、繰り返しの範囲指定が用いられる。これはBNFで以下のようになる。

$$<p>^{(n,m)} ::= <p>^{n} |<p>^{n+1}| \cdot \cdot <p>^{m}. \quad \cdot \cdot (3.3)$$

この列挙形の定義を毎回記述することは、定義情報の見通しを悪くするため、ワーニエ-オーの定義情報を記述する場合、これも特に標準形式として用意すべきであろう(例6)。

4. おわりに

ここでは、BNFパーサにより素過程モデルの入出力情報のデータ構造の定義を導入し、これによってNDFAおよびFA相当の情報定義を可能とした。

但し、導入されたのはデータに関する「文法」であり、さらにその値(各BNFパーサが獲得した値)の取扱いを検討する必要がある。

また、モデルには変数の概念が導入されておらず、定義されたデータ構造は、

$$x = a^{自然数}\ b^{自然数}\ c^{自然数} \quad \cdot \cdot (4.1)$$

というレベルのもので、数学的な解釈での、

$$x = a^{n}\ b^{n}\ c^{n} \quad (n=1,2,..) \quad \cdot \cdot (4.2)$$

等のデータ構造を表現することが出来ない。

さらに、記述された階層関係は Yordon[7] のいうWhole-Part 構造であり、オブジェクト指向分析等に見られるような情報のクラス分類の記述は定義されていない(つまりGen−Spec構造は記述できない)ため概念的な部分の情報が欠落しており、データ定義およびその解釈が効率的であるとは言えない。

これらの問題等に関しては、処理系の実現方式を含めて、現在さらに検討を行っている。

[参考文献]

[1] Tom DeMarco, Structured Analysis and System Specification, YORDON, Inc., 1979.

[2] Derek J. Haley, Strategies for Real-Time System Specification, Dorset House Publishing Co., Inc., 1986.

[3] Edward Yordon, Managing the Structured Techniques, Prentice-Hall, Inc., 1986.

[4] 松本, CASE指向抽象ソフトウェアモデル, 情報処理学会 ソフトウェア工学研究会報告 71−9, 1990.

[5] 松本 他, CASE指向ソフトウェアモデル, 情報処理学会第39回全国大会予稿集, p1379−1380, 1989.

[6] James Martin etal, Diagramming Techniques for Analysts and Programmers, Prentice-Hall, Inc., 1985.

[7] Edward Yordon, Object-Oriented Analysis, OOA CASE Techinical Forum in Autumn '90, 1990.

[8] P.A.M. Dirac, The Principles of Quantum Mechanics, Oxford At The Clarendon Press、 1958.