ソフトウェア工学 99－2
（1994．7．7）

Zにおける仕様記述変換手法

張漢明　荒木啓二郎

奈良先端科学技術大学院大学　情報科学研究科

ソフトウェア開発における形式的手法の課題として、システムをいかにして数学的なモデルで表現するか、という問題がある。本稿では、Zにおける二項関係モデル間の仕様記述変換の手法について述べる。具体例として予約管理システムの仕様記述を通して、二項関係モデルである全域関数、部分関数、関係の間の仕様記述変換の方法を説明し、それぞれの仕様記述の比較検討をおこなう。仕様記述方法論の提示は、仕様記述者に対し設計の指針を与え、また、仕様記述変換技術は、目的に応じた最適なモデルの表現の選択を可能にし、システムのモデル化および仕様設計の支援を図ることができる。

Specification Transformation Method with Z

Han-Myung Chang and Keijiro Araki

Graduate School of Information Science
Nara Institute of Science and Technology

One of the most important problems in software development is how to make a mathematical model for an application problem. In this paper we present transformation methods for specifications between binary relations in Z, which support building formal models for applications. We illustrate our methods with a simple but non-trivial example, and discuss relationships between transformed specifications in binary relation models such as total functions, partial finctions or relations. By transforming a specification in binary relation models, we can select an appropriate representation for the specification and treat it in a clear manner.

1 はじめに

本研究は、Zを用いた形式的手法によるソフトウェア開発において、モデル化支援および仕様設計支援を図ることを目的としている。形式的手法とは、コンピュータシステムのモデル化、設計、解析をおこなうための数学をベースとした技術である。形式的手法の実際の開発における有用性については、[1] において明解に述べられている。[5] では、形式的手法を用いた実際の開発プロジェクトの事例に対する分析と評価を報告している。形式的手法は、コンピュータシステムの開発において、他のエンジニアリングの分野で適用されている数学理論の役割を担うものとして期待されている。対象システムの仕様を数学モデルで表現することにより、仕様に対して簡潔かつ厳密な数学的議論を可能としている。Zは、集合論と一階述語論理を基にしたモデル指向の形式的仕様記述言語である。Zを用いた開発プロセスとしては、抽象度の高い仕様記述からプログラムコードと同等のレベルまで、段階的詳細化によるトップダウンの開発手法が提案されている [2] [3] [6]。最も抽象度の高い仕様を記述することは、対象システムを数学モデルで表現すること、すなわち概念のモデル化に相当する。形式的手法によるソフトウェアの開発において、対象システムのモデル化、つまり「対象システムをどのように数学モデルを用いて表現するか」は重要であり最も本質的な問題である。

本研究では、数学的なモデル間における形式的な仕様記述変換の手法を示すことにより、仕様記述者に対して対象システムのモデル化および仕様設計支援を図る。仕様設計者は、自由な発想でいろいろな視点からモデル化を試み、試行錯誤の結果、最適なモデルを決定する。このモデル化作業は、仕様設計者のひらめきや経験によるところが大きい。図1は仕様記述変換の概念を表している。Spec 1 から Spec 4 は、同じ概念を同一レベル（抽象度）でモデル化した仕様記述を表し、仕様記述間の矢印は、各モデル間の形式的な仕様記述変換を表している。この図では、Spec 1 は、対象システムの概念を直接モデル化した仕様記述であることを表し（実線の矢印）、Spec 2 から Spec 4 は、直接モデル化したものではなく、Spec 1 から仕様記述変換により得た仕様記述であることを表している（破線の矢印）。仕様設計者がさまざまなモデル化を試みる際には、それぞれゼロから仕様記述をおこなうのではなく、

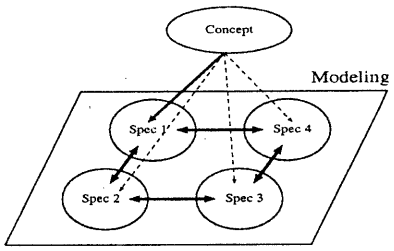

図 1: 仕様記述変換

それまでにおこなった仕様記述の概念や情報を基に、別のモデルによる仕様の再記述をおこなっている。仕様記述変換の手法の提示は、このような仕様設計者のプロセスを明らかにし、形式化することに相当する。仕様設計のプロセスでは、まず、最も考えやすいモデルでモデル化をおこない、その後、仕様記述変換により得られる様々なモデルで、仕様を検討および検証することができる。目的に適したモデルで仕様記述を検討および検証することにより、仕様記述の信頼性を向上させることが期待できる。以上のように、仕様記述変換は、仕様設計者に対して

- モデル化の指針
- 仕様の検討および検証の際の目的に応じた最適なモデルの選択

を与えることにより、対象システムのモデル化および仕様設計支援を図ることができる。

本稿では、モデルを二項関係に限定し、二項関係モデル間における仕様記述変換の手法について述べる。ここでは、二項関係モデルとして、全域関数、部分関数、関係における仕様記述変換を示す。まず、全域関数による仕様記述をおこない、その後、部分関数、関係へ仕様記述変換をおこなう。部分関数への仕様記述変換（部分関数化）は、部分関数の未定義項に対して、全域関数の仕様との意味的な対応付けにより実現する。また、関係への仕様記述変換（関係化）は、部分関数表記と関係表記の間の対応付けにより実現する。

本稿では、具体例として「予約管理システム」の仕様記述をとりあげ、我々の仕様記述変換のアプローチを示すことを試みる。2節では、全域関数における予約管理システムの仕様記述をおこなうと共に、Zの仕様構成概念を示す。3節では、部分関数および関係への

仕様記述変換の手法について述べる。最後に、4節では仕様記述変換についての考察をおこなう。なお、本稿におけるZの表記は、[4] による表記法を使用した。

2 予約管理システム

本節では、全域関数モデルにおける予約管理システムの仕様記述をおこなうと共に、Zの仕様構成概念を示す。

予約管理システムとは、「ある共有施設の予約管理をおこなうシステム」である。このシステムは、予約時間のデータを保持するデータベースと、予約の登録、削除、表示などのオペレーションから構成されている。本システムは、同じ時間に重複した予約は許さないことを想定しているので、データベースは、予約時間の非重複性を保証する必要がある。

Zは、英国のオックスフォード大学を中心に開発された、集合論と一階述語論理をベースにした形式的仕様記述言語である。Zは、数学をベースとしたモデルを構築し、目的対象が満たすべき性質を正確に記述し、証明することを目的としている。Zにおける仕様記述では一般的に、対象システムをシステムがとり得る状態空間（システムスペース）と、システム状態の操作（オペレーション）を定義することにより表現する。オペレーションの記述は、紙数の都合上、予約の登録についてのみおこなう。本節では、Zになじみのない読者にも理解しやすいように、読み進める上で必要な概念や表記は、適時、説明を補足している。

データ

まず、日付の定義をおこなう。ここでは、日付の内部構造については言及しない。したがって、日付は、ユーザ基本型で定義する。

$[DATE]$

ユーザ基本型は、ユーザが定義するプリミティブなデータ型で、$DATE$ は日付の集合を表す。

次に、時刻の定義をおこなう。時刻は時と分で構成されているとし、時は0から23までの整数、分は0から59までの整数とする。

$$Hour == 0\,..\,23;\ Min == 0\,..\,59$$

$$\begin{array}{|l} Time \\ \hline h : Hour \\ m : Min \\ \hline \end{array}$$

$Hour$ は0から23までの整数の集合、Min は0から59までの整数の集合を表す。そして、$Time$ は、時 h と分 m で構成されている。

また、時刻の順序関係を二項関係 $\underline{L}$ と $\underline{G}$ を定義することにより表現する。$A\ \underline{L}\ B$ は A が B より時刻が早いことを表し、$A\ \underline{G}\ B$ は A が B より時刻が遅いことを表す。

$$\begin{array}{|l} \_\,\underline{L}\,\_ : Time \leftrightarrow Time \\ \hline \forall x, y : Time \bullet x\ \underline{L}\ y \Leftrightarrow \\ \quad x.h < y.h \lor (x.h = y.h \land x.m < y.m) \\ \hline \end{array}$$

$$\begin{array}{|l} \_\,\underline{G}\,\_ : Time \leftrightarrow Time \\ \hline \forall x, y : Time \bullet x\ \underline{G}\ y \Leftrightarrow \\ \quad x.h > y.h \lor (x.h = y.h \land x.m > y.m) \\ \hline \end{array}$$

$\underline{L}$ の宣言部では、$\underline{L}$ が中置記法であり、$Time$ と $Time$ の関係であることを示し、述語部において $\underline{L}$ の制約を記述している。$\Leftrightarrow$ は同値関係を表す二項演算子である。したがって、$x\ \underline{L}\ y$ は、「時が x の方が小さいか、時が同じで分が x の方が小さい」ことを表す。$\underline{G}$ についても同様である。

次に、予約時間の定義をおこなう。予約時間は、開始時刻と終了時刻で構成されているとする。

$$\begin{array}{|l} SETime \\ \hline s, e : Time \\ \hline s\ \underline{L}\ e \\ \hline \end{array}$$

$SETime$は、開始時刻 s と 終了時刻 e で構成されており、予約時間の開始時刻は、終了時刻より早いという制約がある。

最後に予約時間の集合を定義する。

$$Rset == \{r : \mathbb{P}\ SETime \mid (\forall x, y : r \mid x \neq y \bullet x.e\ \underline{L}\ y.s \lor x.s\ \underline{G}\ y.e)\}$$

$Rset$ は予約時間の集合の集合を表す。すなわち、$Rset$ は考え得る全ての予約時間の集合の集合である。集合定義の述語部では、予約時間が互いに重なりあうことはない、つまり予約時間の非重複性を表している。

システムスペース

本システムは、予約時間のデータをデータベースに保持する。システムの内部状態は、このデータベースそのものに相当するとみなす。ここでは、日付から予約時間の集合への全域関数として表現する。

$$
\begin{array}{|l}
\hline
TSystem \\
db : DATE \longrightarrow Rset \\
\hline
\end{array}
$$

db は、$DATE$ から $Rset$ への全域関数であることを宣言している。システムが保証しなければならない予約時間の非重複性は、前に定義した $Rset$ において保証されている。システムスペースで定義される制約は、システムの内部状態の状態不変条件とみなすことができる。

ところで、システムの内部状態の初期状態を定義する必要がある。システムの初期状態は、何も予約されていない状態とする。

$$InitTSystem \mathrel{\widehat=} [TSystem' \mid \forall d : DATE \bullet db'(d) = \varnothing]$$

$\mathrel{\widehat=}$ は、スキーマの定義を表し、$TSystem'$ は後状態を表わす。Zでは、変数名の最後に $'$ がついた変数は後状態の変数を表し、そのままの変数は、前状態の変数を表す。$InitTSystem$ は、「すべての日付に対して予約時間の集合が空集合である」ことを表す。

オペレーション

オペレーションは、システムの内部状態変化を、システムの前状態と後状態の関係を定義することにより表現する。

予約の登録は、日付と予約時間を入力とし、入力された予約時間をデータベースに加える。

$$
\begin{array}{|l}
\hline
TEntry \\
\Delta TSystem \\
d? : DATE \\
t? : SETime \\
\hline
t? \notin db(d?) \\
db' = db \oplus \{d? \mapsto db(d?) \cup \{t?\}\} \\
\hline
\end{array}
$$

$\Delta TSystem$ は、オペレーション実行における、システムの前状態と後状態の変数の宣言を表す。したがって、

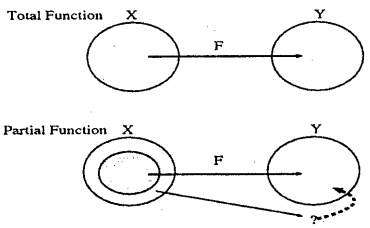

図 2: 部分関数化

db と db' は、実行前と実行後のシステムの内部状態を表す。また、$d?$ は日付、$t?$ は予約時間の入力を表す。Zでは、変数名の最後に ? がついた変数は、それが入力であることを示している。述語部では、「実行後の内部状態は、実行前の内部状態に入力の予約時間を加えた状態である」ことを表している。ところで、$\Delta TSystem$ による、システムの前状態と後状態の変数の宣言は、オペレーションの実行前後において、システムの内部状態の状態不変条件が満足されていることを表している。$TEntry$ では、予約時間の非重複性の保証は陽に示されていないが、$\Delta TSystem$ による状態不変条件の保存により、オペレーション実行後のシステムの予約時間の非重複性は保証されている。

3 仕様記述変換

本節では、全域関数から部分関数、関係への仕様記述変換の手法を、前節で示した予約管理システムの仕様記述を用いて説明する。部分関数への仕様記述変換は、部分関数の未定義項に対して、全域関数の仕様との意味的な対応づけにより実現する。また、関係への仕様記述変換は、部分関数表記と関係表記の間の対応付けにより実現する。

3.1 部分関数化

ここでは、全域関数から部分関数への仕様記述変換の手法を予約管理システムの仕様記述を用いて説明する。

部分関数では未定義項が存在するので、この未定義項が表現している意味を考える必要がある。この仕様記述変換においては、全域関数における仕様の意味と、部分関数における仕様の意味は同じでなければならな

い。部分関数の定義域内の写像は、全域関数と同じでよい。そこで、定義域外の未定義項に対して、その意味を考え、全域関数における意味との対応付けが可能であれば、部分関数化が可能となる（図2）。

予約管理システムの例では、未定義項は「予約データがない」という意味に対応させることができる。「予約データがない」は、予約時間の集合では空集合で表されている。したがって、未定義項は空集合に対応させることができる。つまり、部分関数における未定義項を空集合に対応させて仕様記述を変換すれば、全域関数と部分関数で表現している意味は同じになる。

それでは、予約システムの仕様記述に対して仕様記述変換をおこなう。システムスペースの部分集合による記述は、

$$\begin{array}{|l} PSystem \\ \hline db : DATE \pfun Rset \\ \hline \end{array}$$

となる。

次に、システムスペースの初期状態について考える。前掲の $InitTSystem$ の述語部は、

$$\forall d : DATE \bullet db'(d) = \varnothing$$

である。ここで、$db'(d) = \varnothing$ というのは、日付 d に対して関数 db' の値が未定義であるとみなして、

$$\forall d : DATE \bullet d \notin \mathrm{dom}(db')$$

と記述することもできる。これは、次のことを表している。

$$db' = \varnothing$$

したがって、システムの初期状態は以下のように定義することができる。

$$InitPSystem \mathrel{\widehat=} [PSystem' \mid db' = \varnothing]$$

次に、予約の登録のオペレーションの仕様記述変換をおこなう。部分関数による仕様記述は、全域関数による仕様記述から、

1. 関数の適用をおこなっている項の抽出
2. 定義域内、定義域外による場合わけ
3. 定義域外での未定義項の対応付けによる変換

により、形式的に変換することができる。

予約管理システムの例では、$db(d?)$ が関数の適用として抽出される。したがって、変数 $d?$ において $d?$ が db の定義域に含まれるか、含まれないかの場合わけにより定義をおこなう。$d?$ が db の定義域に含まれている場合には、$TEntry$ の述語部と同じ論理式を適用し、$d?$ が db の定義域に含まれていない場合には、$db(d?)$ を 空集合 $\varnothing$ に変換した論理式を適用する。以上の変換を、$TEntry$ に対して施すと、

$$\begin{array}{|l} PEntry \\ \hline \Delta PSystem \\ d? : DATE \\ t? : SETime \\ \hline (d? \in \mathrm{dom}(db) \land \\ \quad (t? \notin db(d?) \land \\ \quad db' = db \oplus \{d? \mapsto db(d?) \cup \{t?\}\})) \lor \\ (d? \notin \mathrm{dom}(db) \land \\ \quad (t? \notin \varnothing \land \\ \quad db' = db \oplus \{d? \mapsto \varnothing \cup \{t?\}\})) \\ \hline \end{array}$$

を形式的に得る。さらに、この述語部を簡単化すれば、

$$\begin{array}{|l} PEntry \\ \hline \Delta PSystem \\ d? : DATE \\ t? : SETime \\ \hline (d? \in \mathrm{dom}(db) \land \\ \quad (t? \notin db(d?) \land \\ \quad db' = db \oplus \{d? \mapsto db(d?) \cup \{t?\}\})) \lor \\ (d? \notin \mathrm{dom}(db) \land \\ \quad db' = db \oplus \{d? \mapsto \{t?\}\}) \\ \hline \end{array}$$

を得ることができる。

3.2 関係化

ここでは、部分関数による予約管理システムの仕様記述から、関係による仕様記述への変換をおこなう。

部分関数によるスキーマ $PSystem$ における db の変数の型は、日付から予約時間の集合への部分関数である。これを、日付と予約時間の関係に仕様記述の変換をおこなう。つまり、X から Y の集合への関数 $fun : X \pfun \mathbb{P}\, Y$

から、X と Y の関係 $rel : X \leftrightarrow Y$ への変換を考える。ここでは、部分関数表記と関係表記の間の対応付けにより変換をおこなう。仕様記述変換におけるの変換の対象は、

1. 部分関数の集合表記
2. 関数の適用の項

である。この二つの対象に対して、部分関数と関係の間の表記の対応付けをおこなう必要がある。

部分関数の集合表記の対応付けの基本アイデアは、例えば、X を $\{a, b, c\}$、Y を $\{s, t, u\}$ としたとき、部分関数の集合表記

$$fun = \{a \mapsto \{s\}, b \mapsto \{s, t, u\}\}$$

を、関係では、

$$rel = \{a \mapsto s, b \mapsto s, b \mapsto t, b \mapsto u\}$$

で表現し、その意味を同一視することである。したがって、部分関数の集合表記

$$fun = \{x : X; y : \mathbb{P}\, Y \mid P \bullet x \mapsto y\}$$

は、次に示す関係による表記に変換することができる。

$$rel = \bigcup\{x : X; y : \mathbb{P}\, Y \mid P \bullet \{e : y \bullet x \mapsto e\}\}$$

但し、P は変数を制約する述語である。

また、関数の適用の表記 $fun(x)$ は、関係においては、$rel(\!|\{x\}|\!)$ と対応付けることができる。

それでは、予約システムの仕様記述に対して仕様記述変換をおこなう。部分集合によるシステムスペース

PSystem

$$db : DATE \pfun Rset$$

は、

PSystem

$$db : DATE \pfun SETime$$

$$\forall d : \mathrm{dom}(db) \bullet db(d) \in Rset$$

と記述することもできる。ここで、変換の対象となる $db(d)$ に対して変換を施すと、関係におけるシステムスペースの記述

RSystem

$$db : DATE \leftrightarrow SETime$$

$$\forall d : \mathrm{dom}(db) \bullet db(\!|\{d\}|\!) \in Rset$$

を得る。システムスペースの初期状態は、変換の対象はないので、

$$InitRSystem \mathrel{\widehat=} [RSystem' \mid db' = \emptyset]$$

である。

次に、予約の登録のオペレーションの変換をおこなう。*PEntry* の述語部は、

$$\begin{array}{l}(d? \in \mathrm{dom}(db) \land \\ \quad t? \notin db(d?) \land \\ \quad db' = db \oplus \{d? \mapsto db(d?) \cup \{t?\}\}) \lor \\ (d? \notin \mathrm{dom}(db) \land \\ \quad db' = db \oplus \{d? \mapsto \{t?\}\})\end{array}$$

である。ここで、変換の対象となる、部分関数の集合表記と、関数の適用の項を抽出すれば、

$$db(d?), \{d? \mapsto db(d?) \cup \{t?\}\}, \{d? \mapsto \{t?\}\}$$

がある。ここでは、この中で、$\{d? \mapsto db(d?) \cup \{t?\}\}$ に対して変換を試みる。これは、

$$\bigcup\{\{e : db(\!|\{d?\}|\!) \cup \{t?\} \bullet d? \mapsto e\}\}$$

と変換される。集合の要素は一つだから、

$$\{e : db(\!|\{d?\}|\!) \cup \{t?\} \bullet d? \mapsto e\}$$

と変形することができ、また、集合を分けて表記すると、

$$\{e : db(\!|\{d?\}|\!) \bullet d? \mapsto e\} \cup \{e : \{t?\} \bullet d? \mapsto e\}$$

となる。これは、さらに、

$$\{d?\} \lhd db \cup \{d? \mapsto t?\}$$

と変形することができる。

ここで、論理式

$$db' = db \oplus \{d? \mapsto db(d?) \cup \{t?\}\}$$

に対して変換を試みると、これは、前の結果を用いて、

$$db' = db \oplus (\{d?\} \lhd db \cup \{d? \mapsto t?\})$$

と変換される。$\oplus$ の性質により

$$db' = (\{d?\} \lhd db) \cup (\{d?\} \lhd db \cup \{d? \mapsto t?\})$$

となり、また、和集合の結合則より、

$$db' = (\{d?\} \lhd db \cup \{d?\} \lhd db) \cup \{d? \mapsto t?\}$$

と変形できる。さらに、これは、

$$db' = db \cup \{d? \mapsto t?\}$$

と変形することができる。

PEntry の述語部において、変換の対象となる項に対して、全て変換を施せば、

$$\begin{aligned}&(d? \in \mathrm{dom}(db) \land \\ &\quad t? \notin db(\!|\{d?\}|\!) \land \\ &\quad db' = db \cup \{d? \mapsto t?\}) \lor \\ &(d? \notin \mathrm{dom}(db) \land \\ &\quad db' = db \oplus \{d? \mapsto t?\})\end{aligned}$$

となる。$d? \notin \mathrm{dom}(db)$ のとき、$db(\!|\{d?\}|\!)$ は空集合とみなすことができるので、$t? \notin db(\!|\{d?\}|\!)$ は真となる。したがって、上記の論理式は

$$\begin{aligned}&(d? \in \mathrm{dom}(db) \land \\ &\quad t? \notin db(\!|\{d?\}|\!) \land \\ &\quad db' = db \cup \{d? \mapsto t?\}) \lor \\ &(d? \notin \mathrm{dom}(db) land \\ &\quad t? \notin db(\!|\{d?\}|\!) \land \\ &\quad db' = db \cup \{d? \mapsto t?\})\end{aligned}$$

と変形できる。これは、

$$t? \notin db(\!|\{d?\}|\!) \land db' = db \cup \{d? \mapsto t?\}$$

と変形することができ、さらに、

$$d? \mapsto t? \notin db \land db' = db \cup \{d? \mapsto t?\}$$

となる。したがって、関係における予約の登録のオペレーションは、

REntry

$\Delta RSystem$
$d? : DATE$
$t? : SETime$

$d? \mapsto t? \notin db$
$db' = db \cup \{d? \mapsto t?\}$

となる。

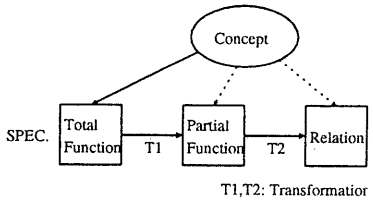

図 3: 仕様記述変換の流れ

4 考察

本稿では、予約管理システムの仕様記述を例にして、二項関係モデルにおける仕様記述変換の手法を示した。図 3は、本稿で示した仕様記述変換の流れを表している。T1 は、全域関数から部分関数への変換、また T2 は、部分関数から関係への変換を表している。

モデル化支援

我々は、「対象システムの数学モデルによるモデル化」という形式的手法の本質的な問題に対して、仕様記述変換技術が有効であると考えている。仕様設計者は、自由な発想でいろいろな視点からモデル化を試み、試行錯誤の結果、最適なモデルを決定する。仕様記述を別のモデルで書き直すことを決断したプロセスには、仕様設計者のひらめきと経験によるところが大きい。この時、仕様設計者はゼロから仕様記述をおこなうのではなく、元の仕様記述の情報を基にしておこなっている。仕様記述変換手法の開発は、仕様記述者が経験的におこなっている仕様記述の書換えのプロセスを形式化する、ということを意味する。これは、仕様記述者に対してモデル化の指針（方向性）を与える。また、様々な仕様記述変換の提示は、一つの仕様記述から他の仕様記述への変換の可能性を提示する、ということを意味する。これは、仕様設計者に対してモデル変換に対する「ひらめき」を与える。

以上のように仕様記述変換技術は、仕様設計者に対して、

1. モデル化の指針
2. モデル変換に対する「ひらめき」

を与えることにより、対象システムのモデル化の支援を図ることができる。

仕様設計支援

本稿では、対象システムを全域関数から部分関数、関係へ変換する手法を提示した。これらの仕様記述のレベル（抽象度）は同じである。仕様のレベルは同じであるが、それぞれの仕様記述では視点が異なる。

予約管理システムの例では、全域関数による仕様記述は、定義域となる日付を無限として扱っているために、仕様記述は扱いやすく、また、対象システムの概念の本質を端的に表しており、システムの本質を理解するには、最も適したものと考えられる。しかし、無限の日付の情報をシステムが保持することは困難であり、全域関数による仕様記述は、インプリメンテーションの観点からは望ましい記述ではない。そこで、部分関数によって内部状態を表せば、関数を有限の状態で保持することができる。しかし、部分関数による仕様記述では、未定義項の関数適用という問題が発生し、仕様記述は複雑になる傾向がある。そこで、まず、全域関数のもとで未定義項の関数適用を特別扱いせずに取り扱った後に、仕様記述変換を用いて部分関数化するプロセスをとることにより、この仕様の複雑化の軽減を図っている。関係による仕様記述では、システムの内部状態を保持するにあたって、データベースシステムにおける表に対して、最も親和性の高い記述となっている。これは、インプリメンテーションにより近い記述であると考えられる。このように、仕様のレベルが同じでも、概念よりの仕様記述や、インプリメンテーションよりの仕様記述がある。

このような一連の仕様記述変換は、概念に近い表現とインプリメンテーションに近い表現を結び付けているもの、と解釈することができる。関係による表現だけでは、仕様設計者が意図している概念を読みとることは難しい。関係による表現では、関数で表現しうる暗黙の概念や情報が欠落する。この暗黙の概念や情報は、仕様理解において重要であり、仕様記述変換による仕様記述間の結びつけが、システムの仕様理解に有効である。

また、様々な仕様記述変換の提示は、いろいろなモデルによる仕様の検討および検証を可能にする。仕様設計者は、目的に応じて様々な観点から仕様を観察することができる。問題に対して最適なモデルで仕様を検討することにより、仕様の信頼性の向上が期待できる。

以上のように仕様変換技術は、

1. 概念とインプリメンテーションの結び付き
2. 目的に応じた最適なモデルによる仕様の検討と検証

により、仕様設計の支援を図ることができる。

今後の課題

逆変換

本稿では、全域関数から部分関数、関係への単方向の仕様記述変換の手法を示した。モデル間の自由な変換をおこなうには、逆の変換も可能でなければならない。部分関数から全域関数への変換は、全域関数から部分関数への変換における対応付けが可能であれば、論理式の変形により、ここで示した変換の逆の形にすれば可能であると考えられる。また、関係から部分関数への変換は、関数の条件を付加することにより可能であると考えられる。具体的な手法の提示は今後の課題である。

一般的手法の提示

本稿では、予約管理システムにおける仕様記述変換の手法を示した。二項関係モデルにおける仕様記述変換を形式的におこなうには、一般的な変換手法を示す必要がある。今後、二項関係モデル間の一般的な考察をおこない、二項関係モデル間における、一般的な仕様記述変換の手法の提示をおこなう予定である。

参考文献

[1] A.Hall:Seven Myths of Formal Method, IEEE Software, Vol.7,No.5,pp.11-19,1990

[2] S.King: Z and the Refinement Calculus, Lecture Notes in Computer Science, Vol.428, Springer-Verlag, pp.164-188, 1990

[3] J.B. Wordsworth: Software Development with Z:A Practical Approach to Formal Methods in Software Engineering, Addison-Wesley, 1992

[4] J.M.Spivey: The Z Notation - A Reference Manual 2nd ed., Prentice Hall,1992

[5] D.Craigen,S.Gerhart and T.Ralston: An International Survey of Industrial Applications of Formal Methods, Volume 1 Study Methodology, Tech.Report PB93-178556/AS, National Technical Information Service, 1993

[6] K.R.Wood: A practical approach to software engineering using Z and the refinement calculus, ACM SOFTWARE ENGINEERING NOTES, Vol.18, No.5, pp.79-88, 1993