数理モデル化と問題解決　22－11
（1998．11．27）

モデル形成支援のための仕様記述変換技術

張 漢明[1,2]　荒木 啓二郎[1,3]

[1] (財) 九州システム情報技術研究所
[2] 奈良先端科学技術大学院大学 情報科学研究科
[3] 九州大学 大学院 システム情報科学研究科

本研究の目的は，ソフトウェア開発の仕様記述の工程におけるモデル化の作業を支援することである．仕様記述の過程では，システムに対する理解が深まるにつれて，より最適なモデルを用いて仕様は書き換えられる．本研究では，モデル間の仕様記述の変換手法を提示することにより，対象システムのモデル化支援を図る．本論文では，形式仕様記述言語 Z を用いて 2 項関係モデル間の対応を分析し，2 項関係モデル間の形式的な仕様記述変換の手法を提示する．仕様記述の内容を変えることなく仕様記述を系統的に変更する手法を与えることにより，仕様記述の変更に伴うエラーの混入を防ぎ，仕様記述の可読性および信頼性の向上が期待できる．

Specification Transformation Techniques to Support Modeling for an Application System

Han-Myung CHANG[1,2]　Keijiro ARAKI[1,3]

[1] Institute of Systems & Information Technologies
[2] Nara Institute of Science and Technology
[3] Kyushu University

One of the most important issues in software development is how to make a mathematical model for an application system. In this paper we present transformation methods for specifications in Z, which support building a formal model for an application system. We discuss relationships between transformed specifications in binary relation models such as total functions, partial functions or relations. The development of specification transformation methods means to clear and formalize the specifier's process of modeling and specification. We expect that reliability for the specification increases because of examining the specification with an appropriate model for the system.

1　はじめに

本研究の目的は，ソフトウェア開発の仕様記述の工程におけるモデル化の作業を支援することである．仕様記述の過程では，仕様記述者は自由な発想で対象システムの仕様記述を試みる．仕様記述者は仕様記述を通して無意識のうちにシステムの分析を行い，対象システムに対する理解を深める．システムに対する理解が深まるにつれて，より最適な仕様記述を得るために仕様記述を再構築する．本研究では，このような仕様記述の再構築の過程に着目した．仕様記述の再構築を仕様記述変換と捉え，再構築の過程を系統的に行う手法を確立することを目指している．

仕様記述の変換を形式的に扱うために形式的手法を導入し，モデル間の対応を形式的に表現することにより，仕様記述の変換を系統的に行う手法を提示する．形式的手法とは，コンピュータシステムのモデル化，設計，解析を行うための数学を基にした技術である．近年，実際のソフトウェア開発に対して，形式的手法の適用が盛んに試みられ，文献 [1] 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が報告されている．また，ソフトウェア開発における形式的手法の有用性については文献 [2] において明解に述べられている．

本論文では，形式仕様記述言語 Z[3] を用いて 2 項関係モデル間の対応を分析し，2 項関係モデル間の形式的

な仕様記述変換の手法を提示する．Zは集合と一階述語論理を基にしたモデル指向の形式仕様記述言語である．二項関係モデルとして，全域関数，部分関数および関係の間の対応関係を明確にし，それぞれのモデル間の仕様記述の変換手法を示す．

仕様記述変換技術は，仕様記述者が仕様記述を変更する時の過程を形式化することに相当する．仕様記述者が勘と経験に頼ってモデル化を行っていた作業を明示することにより，なぜモデルを変更するかの意義を明確にするとともに，仕様記述を構築する時の指針となることが期待できる．また，モデル間で仕様記述を変換することができれば，仕様の検討および検証をする際に，目的に応じた最適なモデルを選択することが可能となる．仕様記述の内容を変えることなく仕様記述を系統的に変更する手法を与えることにより，仕様記述の変更に伴うエラーの混入を防ぎ，仕様記述の可読性および信頼性の向上が期待できる．

本論文の構成は，まず，第2章において仕様記述変換の概念およびZにおける2項関係モデル間の仕様記述の変換手法を提示する．第3章では，具体例を用いて変換手法の適用例を示し，最後に，仕様記述変換の有効性について議論する．なお，本論文におけるZの表記は文献[3]による表記法を使用した．

2 仕様記述変換技術

本章では，まず，仕様記述変換の概念について述べた後，Zにおける2項関係モデル間の仕様記述の変換手法を提示する．

2.1 仕様記述変換の概念

本研究では，仕様記述の過程を

1. 対象システムに対する理解，
2. 最適なモデルの形成，
3. システムに求められる性質の記述
4. システムの分析

の繰り返しであるとみなしている．仕様記述者は実際に仕様を記述することにより，システムの分析を暗黙のうちに行い，徐々に対象システムに対する理解が深まると考えられる．仕様記述においてシステムの性質を表現する方法はモデルに依存する．より最適な仕様記述を得るためには，対象システムのモデル化が最も重要である．仕様記述の過程では，対象システムの分析を基にして，より最適な仕様記述が試行錯誤で得られると考えられる．

図1は仕様記述の概念を図示したものである．仕様記述者は，まず対象システムをモデルAを用いて仕様記述Aを作成した（モデル化A）．仕様記述者は，仕様記述Aを用いてシステムの分析を行いシステムに対する理解が深まり，より最適なモデルBを用いて仕様記述を再構築し仕様記述Bを作成した（モデル化B）．ここで，モデル化Bは対象システムの概念から直接モデル化されたのではなく，モデル化Aで得られた情報を基にして，モデルAからより最適なモデルBへと変換したとみなすことができる．モデル化Aの実線の矢印は，

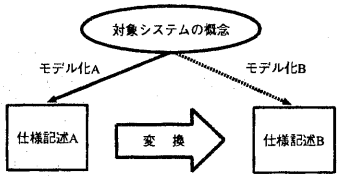

図 1: 仕様記述変換の概念

仕様記述Aが対象システムの概念から直接モデル化されたことを表し，モデル化Bの破線の矢印は，仕様記述Bは概念から直接モデル化されたのではなく，仕様記述Aで得られた情報から再構築されたことを表している．本研究では，この再構築の過程を仕様記述変換とみなし，モデルの変更に伴う仕様記述の変換手法を提示することにより，仕様記述におけるモデル化の支援を図る．

2.2 2項関係モデルにおける仕様記述変換手法

本論文では，システムの内部状態を表現するための最も基本的なモデルである2項関係モデルに着目し，2項関係モデルにおける仕様記述変換の手法について議論する．Zにおいて2項関係モデルの中で最も特徴的なモデルは，全域関数（$\longrightarrow$），部分関数（$\nrightarrow$）および関係（$\leftrightarrow$）である．全域関数，部分関数および関係の対応関係を形式的に表現することにより，3つのモデル間の変換手法を提示する．具体的には，2項関係モデルの変換手法として，

1. 全域関数と部分関数間の変換手法，および
2. 部分関数と関係間の変換手法

を提示する．

2.3 全域関数と部分関数間の変換

まず，全域関数と部分関数間の変換について述べる．

2.3.1 基本アイデア

全域関数と部分関数間の変換の基本アイデアは，

- 部分関数の定義域に属さない要素に対する補助関数を導入して，全域関数が有する情報と部分関数が有する情報を同一視すること

である．図2では，補助関数（AF）を導入して，全域関数（TF）と部分関数（PF）を同一視する概念を示している．AF，TF，およびPFの間には次の条件を満足する必要がある．

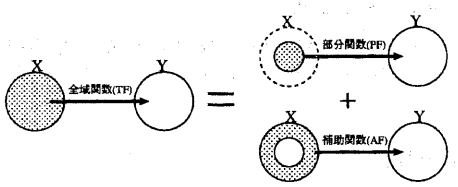

図 2: 補助関数の導入による全域関数と部分関数の同一視

TPACondition[*X*, *Y*]

$tf : X \longrightarrow Y$
$pf, af : X \nrightarrow Y$

$pf = \mathrm{dom}(pf) \lhd tf$
$af = \mathrm{dom}(pf) \lhdd tf$

上記のスキーマ *TPACondition* は関数間の満たすべき条件を記述している．宣言部では全域関数 *tf*，部分関数 *pf* および補助関数 *af* の宣言を行っている．述語部において，1 行目の制約

$$pf = \mathrm{dom}(pf) \lhd tf$$

は「*pf* の対応関係は，*pf* の定義域に属している *tf* の対応関係と同じであること」を表現している．また，2 行めの制約

$$af = \mathrm{dom}(pf) \lhdd tf$$

は「*af* の対応関係は，*pf* の定義域に属していない *tf* の対応関係と同じであること」を表現している．

以上の関係を満足する部分関数と補助関数を定義すれば，全域関数が有する情報と部分関数が有する情報を同一視することができる．

2.3.2 変換手法

まず，全域関数から部分関数への変換について考える．変換は次の手順でおこなう．

1. 補助関数の定義と，変換の対象となる全域関数，部分関数および補助関数の関係をスキーマで記述する．
2. スキーマ合成を用いて部分関数による仕様記述を得る．

手順の詳細は以下に示す．

関数間のスキーマの定義

補助関数の定義および関数間の関係をスキーマを用いて記述する．

TPARelation

$tf : X \longrightarrow Y$
$pf, af : X \nrightarrow Y$

$tf = pf \cup af$
Predicate

宣言部では全域関数 *tf*，部分関数 *pf* および補助関数 *af* の宣言を行う．述語部では，関数間の関係と補助関数 *af* の定義をおこなう．1 行目の制約

$$tf = pf \cup af$$

は，関数間の関係を表現している．2 行目の制約 *Predicate* において，関数間の条件 *TPACondition* を満足する補助関数 *af* の定義をおこなう．*Predicate* では，条件 *TPACondition* を満足するように *af* を定義する必要がある．

部分関数による仕様記述の獲得

全域関数による仕様記述 *TSpec* に対する変換について考える．*TSpec* は以下の通りである．

TSpec

$tf : X \longrightarrow Y$
Decl

Predicate

Decl は変換の対象となる変数 *tf* 以外の変数の宣言を表し，*Predicate* は述語を表す．部分関数による仕様記述 *PSpec* は以下の定義により得られる．

$$PSpec \mathrel{\widehat=} (TSpec \wedge TPARelation) \setminus (tf, af)$$

上の定義では，スキーマ *PSpec* は，

1. 全域関数によるスキーマ *TSpec* と関数間のスキーマ *TPARelation* を論理積によるスキーマ合成により結び付け，
2. 全域関数の変数 *tf* と補助関数の変数 *af* を変数を隠蔽する演算子（$\setminus$）を用いて宣言から取り除く

ことを意味している．したがって *PSpec* の宣言部は

PSpecDecl

$pf : X \longrightarrow Y$
Decl

となり，*PSpec* は部分関数を用いた仕様記述となる．

以上で，全域関数から部分関数への変換の手法を示した．逆に，部分関数から全域関数への変換は，同様にして

$$TSpec \mathrel{\widehat=} (PSpec \wedge TPARelation) \setminus (pf, af)$$

により，全域関数による仕様記述 *TSpec* を，部分関数の仕様記述 *PSpec* から得ることができる．

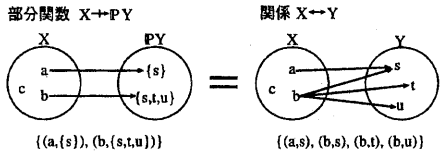

図 3: 部分関数と関係の内容の同一視

2.4 部分関数と関係間の変換

本節では，部分関数と関係間の変換について述べる．ここでは，集合 X から 集合 Y の集合への部分関数 $X \pfun \mathbb{P}\, Y$ と関係 $X \rel Y$ の間の変換について考察する．

2.4.1 基本アイデア

部分関数 $X \pfun \mathbb{P}\, Y$ と関係 $X \rel Y$ 間の変換の基本アイデアは，

- 部分関数 $X \pfun \mathbb{P}\, Y$ と関係 $X \rel Y$ の表記内容を同一視すること

である．これは，例えば集合 X, Y をそれぞれ

$$X = \{a, b, c\}, Y = \{s, t, u\}$$

としたとき，図3に示すように，部分関数 $X \pfun \mathbb{P}\, Y$ の集合表記

$$\{(a, \{s\}), (b, \{s, t, u\})\}$$

と，関係 $X \rel Y$ の集合表記

$$\{(a, s), (b, s), (b, t), (b, u)\}$$

の内容を同一視することである．

これは，部分関数の視点からみれば，「X から Y の集合への部分関数を X と Y の関係として解釈すること」に相当する．逆に，関係の視点からみれば，「X と Y の関係を X から Y の集合への部分関数として解釈すること」に相当する．以上のような部分関数と関係の間の解釈は，以下のスキーマで表現することができる．

IdentifyPR[X, Y]

$$pf : X \pfun \mathbb{P}\, Y$$
$$rl : X \rel Y$$

$$pf = \{x : \mathrm{dom}(rl) \bullet x \mapsto rl(\!|\{x\}|\!)\}$$
$$rl = \bigcup \{x : \mathrm{dom}(pf) \bullet \{y : pf(x) \bullet x \mapsto y\}\}$$

部分関数 pf は関係 rl を用いて表現することが可能で，逆に関係 rl も部分関数を用いて表現することが可能である．*IdentifyPR* の関係からデータ構造を変換する関数を定義することは可能である．この変換関数を用いることにより，どちらのデータ構造にでも仕様記述を変換することができる．

2.4.2 変換手法

部分関数から関係への変換関数 $P2R$ および関係から部分関数への変換関数 $R2P$ を定義し，仕様記述の変換の方法を示す．

変換関数

変換関数 $P2R$ および $R2P$ を定義する．変換関数 $P2R$ は，部分関数 $X \pfun \mathbb{P}\, Y$ のデータ構造を関係 $X \rel Y$ のデータ構造に変換し，逆に，$R2P$ は，関係 $X \rel Y$ のデータ構造を部分関数 $X \pfun \mathbb{P}\, Y$ のデータ構造に変換する関数である．

[X, Y]

$$P2R : (X \pfun \mathbb{P}\, Y) \fun (X \rel Y)$$
$$R2P : (X \rel Y) \fun (X \pfun \mathbb{P}\, Y)$$

$$\forall pf : X \pfun \mathbb{P}\, Y \bullet$$
$$P2R(pf) = \bigcup \{x : \mathrm{dom}(pf) \bullet \{y : pf(x) \bullet x \mapsto y\}\}$$
$$\forall rl : X \rel Y \bullet$$
$$R2P(rl) = \{x : \mathrm{dom}(rl) \bullet x \mapsto rl(\!|\{x\}|\!)\}$$

図3で示した表記例に対して，この変換関数を適用すれば，

$$P2R(\{(a, \{s\}), (b, \{s, t, u\})\}) = \{(a, s), (b, s), (b, t), (b, u)\}$$
$$R2P(\{(a, s), (b, s), (b, t), (b, u)\}) = \{(a, \{s\}), (b, \{s, t, u\})\}$$

を得る．このようにして，変換関数を用いて部分関数と関係の間のデータ構造を自由に換えることができる．

仕様記述の変換

まず，部分関数から関係への仕様記述の変換について考える．この変換では，部分関数の「変数」のデータ構造を関係のデータ構造に変換する．もちろん，変換前の仕様記述の述語部では，「変数」は部分関数として記述されているので，変換後の述語部に変換前と同じ記述を使うことはできない．しかし，関係のデータの意味を部分関数と解釈して，「変数」を部分関数に変換して制約を記述することは可能である．これは，関係の「変数」に対して $R2P$ を適用することにより実現することができる．したがって，変換前と同じ仕様記述を得るには，$R2P$ を用いて変換前の述語部と同じ制約を記述すればよい．これは，変換前の述語部に対して，対象となる変数に $R2P$ を適用することにより得ることができる．

以上の議論より，部分関数 $X \pfun \mathbb{P}\, Y$ から関係 $X \rel Y$ の仕様記述は，以下の手順で得ることができる．

1. 変換の対象となる変数の型を $X \pfun \mathbb{P}\, Y$ から $X \rel Y$ に換える．
2. 変換の対象となる変数に $R2P$ を適用する．

以上で，部分関数 $X \pfun \mathbb{P}\, Y$ から関係 $X \rel Y$ への変換手法を示した．逆に，関係から部分関数の仕様記述は，以下の手順で得ることができる．

1. 変換の対象となる変数の型を $X \rel Y$ から $X \pfun \mathbb{P}\, Y$ に換える．
2. 変換の対象となる変数に $P2R$ を適用する，

3 適用例

本章では，具体例として予約管理システムの仕様記述を用いて仕様記述変換の適用例を示す．

3.1 予約管理システム

予約管理システムとは「ある共有施設の予約管理を行うためのシステム」である．本システムは，予約に関するデータを記録するデータベースを保持し，システムの使用者に対して予約の登録，削除，表示などのサービスを提供する．本論文では操作の例として予約の登録についてのみ取り扱う．

State

$$tf : DATE \longrightarrow (RTime \pfun NAME)$$

$$\forall d : \mathrm{dom}(tf) \bullet \mathrm{dom}(tf(d)) \in RTimeSet$$

Entry

$$\Delta State$$
$$d? : DATE;\ st?, et? : Time;\ n? : NAME$$

$$(st?, et?) \in RTime$$
$$(st?, et?) \notin \mathrm{dom}(tf(d?))$$
$$tf' = tf \oplus \{d? \mapsto (tf(d?) \cup \{(st?, et?) \mapsto n?\})\}$$

State はシステムの内部状態を定義し，*Entry* は登録の操作を定義している．データの定義として，*Date* は日付，*Time* は時刻，*Rtime* は予約時間，*NAME* は名前，*RTimeSet* は予約時間の集合を表している．データの定義は紙面の都合上省略する．

3.1.1 全域関数から部分関数への変換

本節では全域関数から部分関数への変換の適用を試みる．第3.1節では，システム内部状態を表すスキーマ *State* において，データベースの構造を全域関数

$$tf : DATE \longrightarrow (RTime \pfun NAME)$$

としてモデル化した．ここでは，これを部分関数

$$pf : DATE \pfun (RTime \pfun NAME)$$

としてモデル化した仕様記述に変換する．

全域関数では，全ての日付に対応する予約情報をデータベースとして保持するという概念に基づいてモデル化されている．これを部分関数を用いてモデル化するためにはどのような補助関数を導入すればよいだろうか．ここでは，

- 部分関数で保持していない日付に対しては「予約がない」とみなす

というモデル化を考える．「予約がない」は，空集合を対応させることで表現することができる．補助関数として以下の af を導入する．

$$af = \{d : DATE \mid d \notin \mathrm{dom}(pf) \bullet d \mapsto \emptyset\}$$

関数間のスキーマを以下のように定義する．

TPARel

$$State$$
$$pf, af : DATE \pfun (RTime \pfun NAME)$$

$$tf = pf \cup af$$
$$af = \{d : DATE \mid d \notin \mathrm{dom}(pf) \bullet d \mapsto \emptyset\}$$

上の定義では，pf と af の定義域は排他的である．つまり定義域の要素に重なりはないので，関数間の条件 *TPACondition* の制約

$$pf = \mathrm{dom}(pf) \ndres tf \land af = \mathrm{dom}(pf) \nrres tf$$

は確かに満足されている．

全域関数による仕様記述 *State*，*Entry* から部分関数による仕様記述 *PState*，*PEntry* への変換を以下に示す．

$$PState \mathrel{\widehat=} (State \land TPARel) \setminus (tf, af)$$
$$PEntry \mathrel{\widehat=} (Entry \land TPARel \land TPARel') \setminus (tf, af, tf', af')$$

3.1.2 部分関数から関係への変換

3.1.1節では，データベースのデータ構造を

$$pf : DATE \pfun (RTime \pfun NAME)$$

としてモデル化した．Zでは，関数は直積の集合として捉えているので，pf は，

$$pf : DATE \pfun \mathbb{P}(RTime \times NAME)$$

と記述することができる．ただし，pf には関数の制約

$$\forall r : \mathrm{ran}(db) \bullet r \in RTime \pfun NAME$$

を付け加える必要がある．3.1.1節における，システムの内部状態を表すスキーマ *PState* は，以下のように展開することができる．

PStateAlt

$$pf : DATE \pfun \mathbb{P}(RTime \times NAME)$$

$$\forall r : \mathrm{ran}(pf) \bullet r \in RTime \pfun NAME$$
$$\forall d : \mathrm{dom}(pf) \bullet \mathrm{dom}(pf(d)) \in RTimeSet$$

上のスキーマ *PStateAlt* に対して変換を適用すれば，関係による内部状態のスキーマ *RState* を得る．

RState

$$rl : DATE \leftrightarrow (RTime \times NAME)$$

$$\forall r : \mathrm{ran}(R2P(rl)) \bullet r \in RTime \pfun NAME$$
$$\forall d : \mathrm{dom}(R2P(rl)) \bullet \mathrm{dom}(R2P(rl)(d)) \in RTimeSet$$

同様にして，*PEntry* に対しても変換を適用すると，関係による仕様記述 *REntry* を得る．

$$
\begin{array}{|l}
\hline
REntry \\
\Delta RState \\
d? : DATE;\ st?, et? : Time;\ n? : NAME \\
\hline
(st?, et?) \in RTime \\
d? \notin \mathrm{dom}(R2P(rl)) \land \\
\quad R2P(rl') = R2P(rl) \cup \\
\qquad \{d? \mapsto \{(st?, et?) \mapsto n?\}\} \lor \\
d? \in \mathrm{dom}(R2P(rl)) \land \\
\quad (st?, et?) \notin \mathrm{dom}(R2P(rl)(d?)) \land \\
\quad R2P(rl') = R2P(rl) \oplus \\
\qquad \{d? \mapsto (R2P(rl)(d?) \cup \\
\qquad\quad \{(st?, et?) \mapsto n?\})\} \\
\hline
\end{array}
$$

4 ソフトウェア開発における変換手法の意義

形式的手法に基づいたソフトウェア開発手法として，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 [4][5][6]．最も抽象度の高い記述から，データ詳細化およびアルゴリズム詳細化の技術をもとに，系統的にプログラムコードを得るための手法が示されている．しかし，仕様記述の基になる最も抽象度の高い記述がどのようにして得られたかは，そこには述べられていない．

本論文で示した仕様記述変換の技術は，同じレベル（抽象度）の記述において，記述の意味において等価な仕様記述を得るるための変換手法を与える．したがって，仕様記述者は形式的にモデルの変換を行いながら，対象システムの仕様記述を検討することができる．

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過程には，仕様設計者のひらめきと経験によるところが大きいであろう．仕様記述変換手法の開発は，この仕様記述者が経験的に行っている仕様記述の書換えの過程を形式化する，ということを意図する．

本論文の具体例で示した仕様記述の流れは，仕様を構築する際の手法，つまり指針となり得る．全域関数によるモデル化では，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，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．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．また，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考え易いモデルで仕様記述を検討した後，仕様記述変換を用いて部分関数化，さらに関係化という過程をとることにより，この仕様の複雑化の軽減を図り，また，容易にインプリメンテーションに近い記述を得るという手法を得ることができた．

このように，仕様記述変換技術は，仕様記述者がモデル化をおこなう際，また，仕様設計をおこなう際の有効な手段であるということがいえる．

5 おわりに

本論文では，形式的手法による仕様記述変換技術として，二項関係モデルにおける仕様記述変換の手法を提示した．Z において 2 項関係モデルの中で最も特徴的なモデルである，全域関数（$\longrightarrow$），部分関数（$\nrightarrow$）および関係（$\leftrightarrow$）の間の対応関係を分析し，

1. 全域関数と部分関数間の変換手法，および
2. 部分関数と関係間の変換手法

を提示した．

仕様記述変換技術は，仕様記述者が仕様記述を変更する時の過程を形式化することに相当する．また，仕様記述変換は，

- 仕様記述を構築する時の指針の提供，
- 仕様の検討および検証をする際の目的に応じた最適なモデルの選択，
- 仕様記述の変更に伴うエラーの混入防止，

を可能にすることにより，仕様記述に対する信頼性向上に寄与できると考えられる．

今後の課題としては，

- 同等な制約記述の分析による記述変換手法の提示
- 仕様記述変換の機械による支援

が挙げられる．形式的な仕様を記述するためには，論理式による記述は避けられない．同じ制約を表現する論理式の記述を分析し，それらの間の論理式の関係を変換規則として提示することにより，仕様記述の支援を図ることができるであろう．このような変換規則も仕様記述変換技術とみなすことができる．今後，様々な変換規則を開発し，仕様記述の変換を機械で支援する環境を構築することにより，仕様記述の支援を図る予定である．

参考文献

[1] 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).

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

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

[4] J.Woodcock and J. Davies:Using Z Specification, Refinement, and Proof, Prentice Hall(1996).

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

[6] C.Morgan:Programming from Specifications, Prentice Hall(1990).