コンピュータと教育 38－11
数理モデル化と問題解決 4－11
（1995. 11. 17）

Z言語を対象とした対話型診断システム

吉田 玉緒　松田 昇　岡本 敏雄
電気通信大学大学院 情報システム学研究科
〒182 東京都調布市調布が丘 1-5-1

あらまし

現在、多くの形式的仕様記述言語が開発されている。中でもZは最も普及している言語のうちの一つである。Zの使用による品質・コスト面での効果が報告されているが、同時にその教育にかかるコストも問題となっている。そこで、ユーザのZ習得を支援するために、特定の問題に関してZで記述された仕様書を診断するシステムを開発する。診断はZ記法に関する表記法・記述内容の論理性・課題に対する正解性について行なう。本システムはZ仕様書の記述の誤りの指摘やユーザからの質問への応答の他に、ユーザの要求に応じてZ仕様書を記述する機能の実装を試みている。この記述機能により、解答の一部をヒントとしてユーザに例示することができ、ユーザのZ習得を容易にすると考える。

Interactive Diagnosing System on Z Notation

Tamao Yoshida　Noboru Mastuda　Toshio Okamoto
The Graduate School of Information Systems,
University of Electro-Communications
1-5-1 Chofugaoka, Chofu-City, Tokyo 182, Japan

Abstract

Many formal specification languages for developing computer systems have been proposed so far. The Z notation (Z) is one of the most famous specification languages. Z is to be beneficial to cost and quality, but it's needed for learners long time and efforts to learn it. We are developing a diagnosing system to support learners constructing a specification in Z. This system diagnoses the documents in Z in terms of the syntax, the semantic, and the validity for the tasks. The system answers to user's question automatically constructing a specification as a hint in response to user's demand. Using this system, users (especially, the biginners) can easily learn how to make specification in Z.

1 はじめに

仕様書を作成する者とその仕様書を基にプログラムを作成する者とが同一人物でない場合、仕様書の解釈の相違という問題が生じることがある。仕様書の解釈の相違が基で作り込んだバグの摘出は困難を極め、高いコストを要する。

この問題に対応するためには仕様書を正確に記述し、解釈の相違をなくすことが必要である。そのために様々な形式的仕様記述言語が開発されている。Z言語(Z)はその一例である。Zに関してIBMとオックスフォード大学との共同研究で総コストの9%削減という成果が報告されている[2]。一方、柴山等によれば、仕様解釈の相違は皆無であったものの、担当者にZの知識がなかったために設計工程は従来の倍になっている[4]。

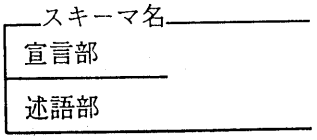
Zによるコスト削減は評価できる。しかし、設計現場にZを導入する際にネックとなるのは、その習得に要するコストである。本研究はこのような問題意識からZを習得しようとしているユーザを支援するためにZで記述された仕様書(Z仕様書)の診断及び記述を行なうシステムを構築する。

2 研究の目的

以上のことを踏まえて、本研究はZを習得しようとしているユーザを支援するためにZ仕様書の診断及び記述を行なうシステムを構築することを目的とする。具体的にはユーザの作成したZ仕様書の表記の間違いや矛盾を指摘し、ユーザからの質問に解答し、ユーザがZ仕様書の続きを作成するのに困難な場合は解答例を提示する。これらを実現するためにはシステムに(1) Zを記述する、(2) Z仕様書を理解する、(3)ユーザの質問に解答するといった3つの機能を付加しなければならない。本稿では(1)と(2)について述べる。

3 Z言語

Zでは、スキーマと呼ぶ構造でシステムの仕様を記述する。スキーマには、システムにおけるデータの関係や状態を表す状態空間スキーマ、システムの初期状態を表す初期状態スキーマ、システム状態に対する操作を表す操作スキーマがある。スキーマは次のような図的記法により表現する。



宣言部にはスキーマで扱うデータ等を宣言する。述語部にはデータが満たす条件や関係を表す。更に、スキーマに対して離接・合接等の演算を行なうことによってスキーマ間の関係付けが可能である。これらのスキーマを用いてシステムの仕様を階層的に厳密に記述することができる。

Zの一例として、本システムで課題として扱っている簡単な図書管理システムのZ仕様の一部を挙げる[3]。

$[book, person]$

library

$loan, shelves, books : \mathbb{P}\, book$
$borrowers : \mathbb{P}\, person$
$lent : book \pfun person$

$loan = \mathrm{dom}\, lent$
$\mathrm{ran}\, lent \subseteq borrowers$
$loan \cup shelves = books$
$loan \cap shelves = \emptyset$

add_book_to_library_ok

$\Delta library$
$b? : book$
$r! : response$

$b? \notin books$
$shelves' = shelves \cup \{b?\}$
$r! = book\_added$

図 1: Z仕様書の例

ここで、bookとpersonは基本型(所与集合)と呼ばれ、システムが扱うデータである。データのシステム内における抽象的な状態としては、本の部分集合books(図書館の蔵書)、shelves(開架の本)、loan(貸し出し中の本)、人間の部分集合borrowers(利用者登録をした人)及び本と人間との対応関係lent(貸し出し中の本と借りている人の対応)がある。操作スキーマadd_book_to_library_okは図書館の蔵書に新着の本を加える操作である。修飾記号?、!、'はそれぞれ入力、出力、事後状態を表す。また、宣言部のΔはスキーマの状態が変化することを、Ξは変化しないことを表す。

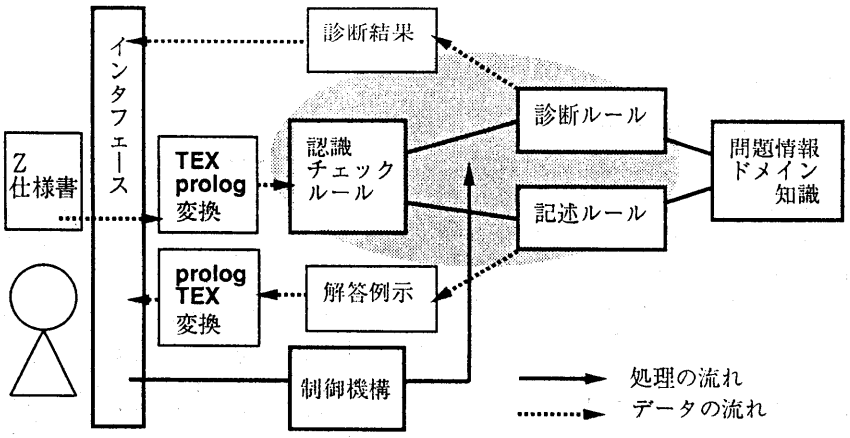


図 2: システム構成図

4 システムの構成

本システムには以下の基本的な機能を実装する。

1. ユーザの作成したZ仕様書に対する診断
   (a) 表記の誤りを指摘する「表記法診断」
   (b) 論理的矛盾を指摘する「論理性診断」
   (c) 課題に対する正当性を指摘する「正解性診断」
2. ユーザの指定箇所または作成中のZ仕様書の続きの解答の例示
3. 推論過程、Zスキーマの処理内容などに関するユーザからの質問への解答

本システムの全体構成を図2に示す。各モジュールの機能・役割は以下の通りである。

(1) 認識・チェックルール
入力されたZ仕様書に対して表記の誤りを対象とする「表記法診断」を行ない、次節に述べるZスキーマの認識処理を行なう。

(2) 診断ルール
Zスキーマの認識結果を受けて論理的矛盾を対象とする「論理性診断」及び課題に対する正当性を対象とする「正解性診断」を行なう。

(3) 記述ルール
Zスキーマの認識結果を受けて記述不足分を補った「解答例」を作成する。また、ユーザの指示により指定箇所のみの「解答例」の作成も行なう。Z仕様書作成作業を (1) 基本型の定義、(2) 状態空間スキーマ、(3) 初期状態スキーマ、(4) 操作スキーマ、(5) スキーマ演算による操作記述の5段階に分けて考えており、当該段階の「解答例」を作成することとしている。

(4) 問題情報 (ドメイン知識)
課題に関する情報を保持する。診断ルールの正解性診断及び記述ルールの解答例作成の際に使用する。

(5) 制御機構
ユーザの診断要求に応じて診断ルールを駆動し、診断結果を表示する。更にユーザからの質問に答えて、解答の例示 (記述ルールの駆動)、推論過程の提示、スキーマの説明などを行なう。

5 Z仕様の記述・認識

先に述べた通り、Zは最初に状態空間スキーマを定義し、これに基づいてデータ等を操作する処理を操作スキーマとして記述していく。そこで、状態空間スキーマにおけるデータの抽象的な状態を以下のものから構成することにする。即ち、

- 集合 (Set)
- 関数 (Function:a → b)
- ベクトル値関数 (Function:a → (b → c))

である。集合や関数という状態を取るデータに関しては、次節以降に述べるようにZの操作スキーマを何通りかの定型パターンで表すことができる。したがって、状態空間スキーマがこれらの集合、関数で記述されていれば、後に続く操作スキーマを自動的に作成することが可能であると考える。

5.1 操作スキーマの定型パターン化

ここでは操作スキーマの定型パターンについて考えるにあたって簡単のために最初に1つの集合のみを考える。このようなシステムにおける操作とは

- 集合への要素の登録
- 集合からの要素の削除

の2つとなる。更に、これらはいずれも正常系の処理であるから各々に対する異常系の処理が考えられる。つまり、集合が1つだけ存在する抽象状態においては要素の登録と削除各々の正常系と異常系の4種類の処理が存在する。

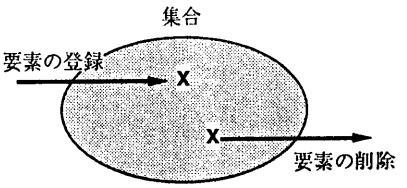
関数の場合は図4に示す通りより複雑になるが、集合の場合の拡張と考えられる。

ここで問題となるのは、このような規定を設けた場合にどの程度複雑な問題までシステムが扱うことができるかである。勿論、当該システム固有の条件などを上記のパターン化した操作スキーマに付記する必要はある。共通問題「在庫管理システムの作成」を例にとるとその大部分をこれら定型パターンだけで記述することができると考える [5]。

5.2 Zスキーマの記述

状態空間スキーマにおいて、集合で定義された部分に関して集合に基づく定型パターンを用いて正常系操作スキーマを記述する。また、関数で定義された部分に関しても同様に正常系操作スキーマを記述する。

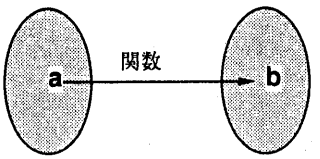
Zスキーマの定型パターンを用いて正常系操作スキーマを作成した様子を図5に示す。これらの定型パターンに対して課題により与えられた基本型、データ状態 (集合、関数など) を変換して図書の登録抹消処理を行なう操作スキーマの原型を作成する。



1つの集合からなるシステムにおける操作処理

- 要素の登録処理（正常系）
- 要素の削除処理（正常系）
- 登録のエラー処理（異常系）
- 削除のエラー処理（異常系）

図 3: システムの抽象状態が1つの集合のみの集合の操作処理



1つの関数からなるシステムにおける操作処理

- 対応（a, b）の登録処理
- 対応（a, b）の削除処理
- 定義域の要素の削除処理
- 値域の要素の削除処理
- 定義域の要素に対応する関数の値の抽出処理
- 関数の値に対応する定義域の要素の抽出処理
- 上記の処理に対する異常系の処理

図 4: システムの抽象状態が1つの関数のみの集合の操作処理

これにZの表記法や集合・関数などに関する一般規則を表したルールを適用することによって課題の要求するZ仕様書を作成していく。

異常系のデータ条件は正常系のデータ条件の補集合である。したがって、正常系スキーマのデータ条件からその補集合を定め、エラーメッセージの出力など必要な操作を記述すれば異常系操作スキーマを作成することができる。

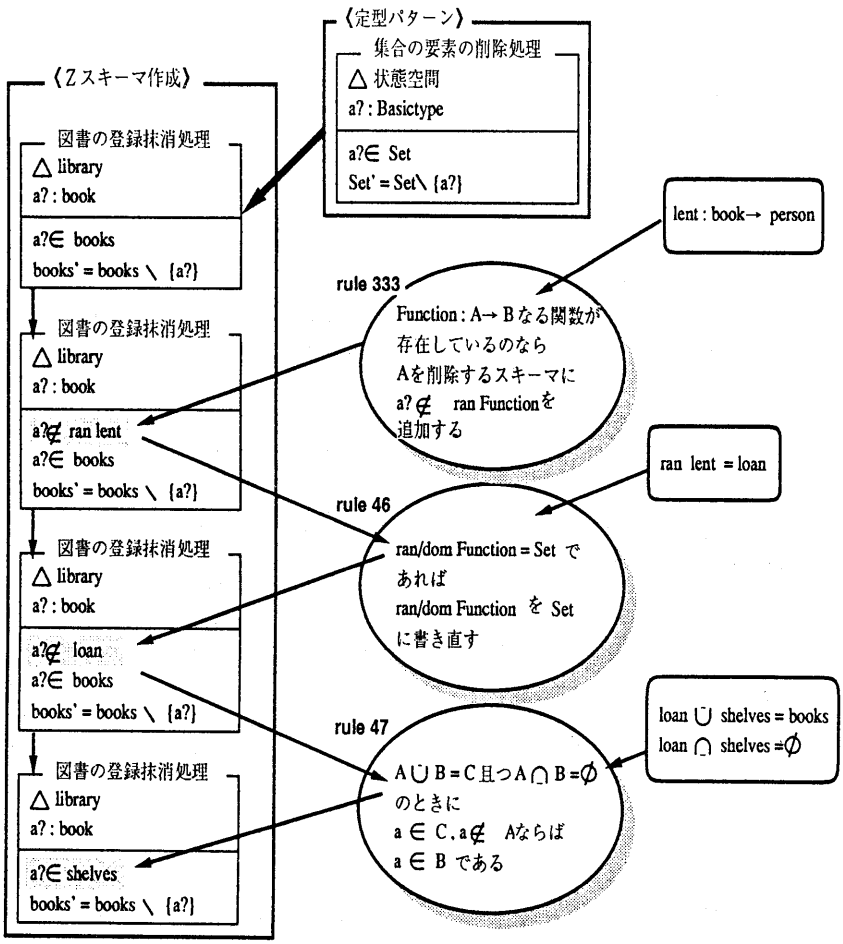


図 5: 記述用ルールを適用した様子

5.3 Zスキーマの認識

本システムがZ仕様書を作成する場合の処理は前節に述べた通りである。逆に本システムがユーザの作成したZ仕様書を診断するためにこれを解釈する場合も操作スキーマの定型パターンを用いる。

正常系の操作スキーマでは表1に示すように記述中にその操作が特徴的に現れてくるので、認識は比較的容易である。異常系の操作スキーマは正常系の操作スキーマのデータ条件の補集合またはその部分集合をデータ条件としているのでそれにより認識することが可能である。

異常系の操作スキーマのデータ条件について図6に示す。本の返却処理の場合、返却図書がloanの要素であれば、正常に処理を行なう。loanの要素でなければエラー処理を行なうこととなる。このとき、一律にエラー処理を行なうのか、それともshelvesの要素とbooksの補集合の要素との各々に異なったエラー処理を行なうかは問題情報に依存する。1つの正常系の操作スキーマに対する異常系の操作スキーマは1つとは限らないのである。

5.4 Z仕様書の自動診断

本システムはユーザの作成したZ仕様書に対して、最初に表記法診断を行なう。続いて、前節に述

表 1: 定型パターンにおける特徴的な操作

項番	定型パターン	特徴的な操作の例
1	登録処理	Set' = Set ∪ { a? } Function' = Function ∪ { a? ↦ b?}
2	削除処理	Set' = Set \ {a?} Function' = Function \ {a? ↦ b?}
3	定義域の要素の削除処理	Function' = { a? } ⩤ Function
4	値域の要素の削除処理	Function' = Function ⩥ { a? }
5	定義域の要素の抽出処理	Set! = { a:定義域 | Function(a) = b? }
6	値域の要素の抽出処理	Set! = { b:値域 | Function(a?) = b }

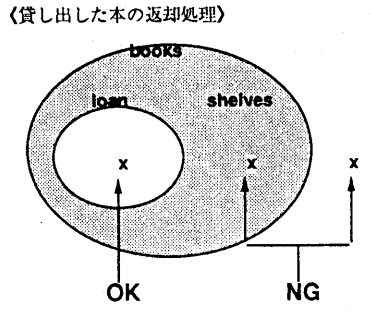


図 6: 異常系スキーマのデータ条件

べたZスキーマの認識処理を行ない、Zスキーマ毎に論理性診断・正解性診断を行なう。前節までに述べたように定型パターンに応じたZスキーマを同定することができれば、そのスキーマに記述すべきことはZ記法の文法と問題情報 (ドメイン知識) から明らかである。システムは記述用のルールを駆動することで記述すべき内容を知ることができる。記述すべき内容とユーザの作成したZ仕様書の内容とを照合することによってシステムはZ仕様書の診断を行なう。但し、3の (3) 記述ルールで述べた通り、Z仕様書には作成段階があるので、認識処理以降はこの作成段階毎 (基本型、状態空間スキーマ、初期状態スキーマ、操作スキーマ、操作記述) に行なっている。

表記法診断としてはΔ、Ξ、' などの記号を用いたZの記法の正誤など、論理性診断としては操作スキーマの合成を行なう操作記述と操作スキーマとの矛盾など、正解性診断としては前節で述べた異常系操作スキーマの記述内容と問題情報との矛盾などを診断している。

6 今後の課題

本稿では、Z仕様書の診断に関して表記の誤り、論理的矛盾、課題に対する正誤の3つに分類し、階層化した診断法を提案した。また、Zスキーマに関して定型パターンを設けることとZの表記法および集合論等に関する知識ベースを持つことでZ仕様書を診断・記述する方法について述べた。現在、prologを用いてシステム構築を行なっている。今後の課題について以下にまとめる。

- 具体的で理解しやすい診断メッセージの工夫
- ユーザの質問に解答する機能
- インタフェース等の機能

これらについて、今後、ユーザのZ言語習得支援という観点から機能の検討及びその実装を行なう。

参考文献

[1] The z notation:2nd ed J.M.Spivey Prentice Hall 1992

[2] ソフトウェア仕様記述の先進技法－Z言語 B. ポター他トッパン 1993

[3] Software Development with Z J.B.Wordsworth Addison-Wesley Pub.Co., 1992

[4] Z言語によるソフトウェアの仕様開発 柴山武彦, 内記美絵, 篠木裕二, 大槻繁 情報処理学会研究報告 SE-89 Vol.92,No.100 1992

[5] 新しいプログラミング・パラダイムによる共通問題の設計 二村良彦, 雨宮真人, 山崎利治, 淵一博 情報処理 Vol.26,No.5 1985