B-001

形式仕様言語 Z の自動検証化の試み
An Approach to Providing an Automatic Verification System for Z Specifications

石川 洋 †
Hiroshi Ishikawa

1. まえがき

形式仕様言語 Z は高い表現力を持っているが，それによって記述された仕様を検証するためには，人間が地道に演繹規則に従って計算しなければならない．ソフトウェアの規模が増大化し，構造が複雑化すると，もはや人手で処理することは困難である．この問題を解決するために，Z で記述された仕様 (以降，Z 仕様と呼ぶ) を実行可能な処理系で動作可能な記述に変換し，コンピュータ支援による検証を行なうというアプローチがいくつか提案されている [1],[5],[10]．本稿では Z 仕様を推論するための論理体系 [6] を利用することを想定し，Z 仕様の検証作業を実行可能な処理系を持つ代数仕様言語 [2],[8] を利用して自動化する方法について提案する．このアプローチはすでに [10] において提案されているが，Z 仕様の代数仕様への変換作業は手動で行っていた．この場合，大規模な仕様を扱う際，変換作業に時間を要する，変換ミスを誘発するといった問題が生じる．その問題を解決するための一案として，Z 仕様から代数仕様への変換部分の自動化を提案する．

2. 形式仕様言語 Z

Z[9] は集合論と一階述語論理に基づいた，スキーマと呼ばれる記述を単位とする表現能力の高い形式仕様言語である．

Z では仕様化する対象を，とりうる状態の集まり，初期状態，および状態に適用する操作の 3 種類で表現しており，それぞれを，状態スキーマ，初期状態スキーマ，および操作スキーマとして記述する．スキーマはスキーマ名，宣言部，述語部から構成される．

Z は仕様を記述することに重点を置いており，実行可能な処理系は備えていない．

3. Z の演繹システム

Z の記述に基づいた検証を行うための枠組が幾つか提案されているが，そのなかの一つに，Z の演繹システム [6] がある．Z の演繹システムはシーケントと演繹規則で構成される．シーケントは前件と後件の組 (前件 ⊢ 後件) であり，前件は宣言と述語のリストの組 (宣言,···,宣言 | 述語,···,述語)，後件は述語のリスト (述語,···,述語) で構成される．

演繹規則は，前提から結論を導く規則で，前提はシーケントのリスト，結論はシーケントとして与えられる．

4. 代数仕様言語 CafeOBJ

実行可能な処理系を持つ代数仕様言語 CafeOBJ[2],[8] は代表的な代数仕様言語 OBJ[3] に等式とは異なる，左辺から右辺への書き換え規則 [7] や，カプセル化されたオブジェクトの状態を扱うことが可能な隠蔽代数 [4] を導入し，拡張した言語である．したがって，CafeOBJ は OBJ が備えている機能であるパラメータ付きモジュールの利用，ソート間の順序の定義，名前を変更したモジュールの参照，および，自由度の高い項の記法といった特徴はすべて継承している．また，新たに導入，拡張された機能や概念により，等式論理の範囲で記述することが困難であった動的なシステムの状態遷移や，実装をブラックボックスと見なしたオブジェクトの記述が可能となった．本稿では CafeOBJ を等式論理の範囲で利用することを想定している．

5. Z 仕様の代数仕様への変換

Z の演繹システムを用いることで Z 仕様に基づく検証を形式的に行えるが，これらを代数仕様の表現に変換することで，検証作業の自動化が可能となる．本節では，Z 仕様を代数仕様に変換するための大まかな方針について述べる．

スキーマはスキーマの名前，宣言部，述語部から構成されていることに着目し，各々を N，D，P と略記すると，スキーマを，スキーマ名と宣言部の組，スキーマ名と述語部の組に分解してそれぞれ< N | D >および [D | P] と表現する．前件はこれらのリストの組であったので，

```
< N | D >, ... , < N' | D' > |
[ N | P ], ... , [ N' | P' ]
```

のように表現できる．後件を Con で表すことにすると，シーケントは，

```
< N | D >, ... , < N' | D' > |
[ N | P ], ... , [ N' | P' ] |- Con
```

と表現できる．これらの表現は，代数仕様言語 CafeOBJ における項の記述に基づくものであり，以下のように宣言している．

```
op <_|_>  : SchemaName Declaration -> NamedDec
op [_|_]  : SchemaName Predicate -> NamedPred
op  _,_   : NamedDec   NamedDecs -> NamedDecls
op  _,_   : NamedPred  NamedPreds -> NamedPreds
op _|_|-_ : NamedDecs  NamedPreds
            Preds -> Sequent
```

6. 検証例

誕生日帳の例 [9] を用いて，2 つのスキーマ

BirthdayBook

$known : \mathrm{P}\ NAME$
$birthday : NAME \pfun DATE$

$known = \mathrm{dom}\ birthday$

†福山大学, Fukuyama University

AddBirthday

$\Delta BirthdayBook$
$name? : NAME$
$date? : DATE$

$name? \notin known$
$birthday' = birthday \cup \{name? \mapsto date?\}$

から, 等式

$$known' = know \cup \{name?\}$$

が成り立つことを，先に導入した記法に基づいて示す．
まず，所与のスキーマと示すべき等式のシーケントによる表現は次のようになる．

```
< BirthdayBook|known:\power NAME >,
< BirthdayBook|birthday: NAME \pfun DATE >,
< BirthdayBook'|known':\power NAME >
< BirthdayBook'|birthday': NAME \pfun DATE >
< AddBirthday|name? : NAME >,
< AddBirthday|date? : DATE > |
[ BirthdayBook|known = \dom birthday ],
[ BirthdayBook'|known' = \dom birthday' ]
[ AddBirthday|name? \notin known ],
[ AddBirthday|birthday' = birthday
              \cup { name? \mapsto date?}]
|- known' = known \cup { name? } .
```

ここで，述語 known' = \dom birthday' の右辺を，示すべき等式の known' に代入すると，シーケントは次のように変化する‡.

```
D | P
|- \dom birthday' = known \cup { name? } .
```

つぎに，

```
birthday'= birthday
         \cup { name? \mapsto date? }
```

を birthday' に代入するとシーケントは以下のように変化する．

```
D | P
|- \dom(birthday \cup {name? \mapsto date?})
       = known \cup { name? }
```

集合に関する演算により，示すべき等式の左辺は矢印 (==>) の方向に，

```
\dom(birthday \cup {name? \mapsto date?})
==>
\dom(birthday)
         \cup \dom({name? \mapsto date?})
==>
\dom(birthday) \cup { name? }
```

‡前件は最初のシーケントから変化しないので以降 D | P と略記する．

のように書き換えられる．一方，右辺は，

```
known \cup { name? }
==>
\dom(birthday) \cup { name? }
```

となり，最終的に以下のようなシーケントを得ることで検証が終了する．

```
D | P
|-  \dom(birthday) \cup { name? }
   = \dom(birthday) \cup { name? }
```

7. おわりに

本稿では，Z 仕様を代数仕様に自動変換するための大まかなデザインを示した．仕様の変換プログラムは現在開発中である．変換後の検証作業は [10] においてその有効性が報告されるので，本稿で提案した仕様変換システムが提供できれば，Z 仕様の検証の自動化がより一層進み，検証作業の効率化が期待できる．

参考文献

[1] R.D. Arthan, “On Formal Specification of a Proof Tool,” *Proceedings of Formal Software Development Methods(VDM'91)*, LNCS 551, pp.356–370, 1991.

[2] R. Diaconescu and K. Futatsugi, CafeOBJ *Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*, World Scientific, 1998.

[3] J. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.P. Jouannaud, *Introducing OBJ*, Technical Report, SRI–CSL–92–03, 1992.

[4] J. Goguen and G.Malcolm. “A Hidden Agenda,” *Theoretical Computer Science*,Vol 245, No 1, pp.55–101, 2000.

[5] X. Jia, “An Approach to Animating Z Specifications,” *Proceedings of the 19th Annual IEEE International Computer Software and Application Conference(COMPSAC'95)*, pp.108–113, 1995.

[6] A. Martin, “Encoding $\mathcal{W}$: A logic for Z in 2OBJ,” *Proceedings of Industrial-Strength Formal Methods(FME'93)*, LNCS 670, pp.462–481, 1993.

[7] J. Meseguer, “Rewriting as a Semantic Framework for Concurrency: A Progress Report,” *Proceedings of 7th International Conference of Concurrency Theory*, LNCS 1119, pp.331–372, 1996.

[8] A.T. Nakagawa, T. Sawada and K. Futatsugi, CafeOBJ User's Manual — ver.1.4 —, 1998.

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

[10] 谷津，二木，代数仕様による Z 仕様の検証支援．In コンピュータソフトウェア，Vol.13, No.6, pp.26–42, 1996.