Zによる要求仕様書のスキーマを基準とした類似度の提案

蜂 巣 吉 成 †

本稿では Z による要求仕様書の類似度の計算手法を提案する。Z ではシステムの各部分をスキーマとして記述するが、本手法ではスキーマを単位として類似度を計算する。

1. はじめに

円滑にソフトウェア開発を行うためにはコーディングやテストなどの下流工程だけでなく、要求分析やシステム設計などの上流工程を支援することも重要である。特に要求分析において、開発するシステムに曖昧な点や矛盾が存在すると、開発途中でのやり直しや要求されたシステムと大きく異なるシステムが出来上がることになる。

要求分析では成果物として要求仕様書が作成される。要求仕様書を記述するための体系として形式手法に基づいた仕様記述言語 Z が知られている。Z を用いることにより仕様の曖昧性を排除することができる [1)2)]。

Z による仕様記述を支援する研究・ツールには、Z の文法上の誤りの検出や型のチェック [3)4)5)]、仕様書の書式の変換 (LaTeX や HTML 形式に変換)[6)7)]、仕様の検証 [8)9)]、仕様書からプログラムの自動生成 [10)] などが多く、仕様の再利用についてはほとんど考えられていなかった。

本稿では Z よる仕様の再利用を支援するために、要求仕様書の類似度を計算する手法を提案する。Z ではシステムの各部分をスキーマとして記述するが、本手法ではスキーマを単位として類似度を計算する。類似度計算を利用して、作成したい仕様と類似した仕様を検索して再利用する、多くのスキーマと類似度の高いスキーマをパターンとして抽出する、などが行えると考えられる。

本稿では 2 節で Z について簡単に説明し、3 節で類似度を計算する手法を提案する。

† 南山大学 数理情報学部 情報通信学科

2. 仕様記述言語 Z

仕様記述言語 Z は集合論、述語論理、関係および関数の原理に基礎をおいた数学的表現を用いている。ここでは誕生日帳を例にして Z による仕様記述を説明する。誕生日帳は人名と誕生日を記録するシステムで、データの追加と検索を行えるものとする。なお、誕生日帳の仕様は文献 [1)] を参考にした。

まず、誕生日帳で使われるデータ型 NAME と DATE を宣言する。

$[NAME, DATE]$

次に誕生日帳の状態 BirthdayBook を記述する。この記述はスキーマと呼ばれる。

BirthdayBook

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

スキーマは中央の横線で区切られ、これより上部を宣言部と呼び、スキーマ中で使用する変数や関数を宣言する。本稿では宣言部の 1 行を宣言文と呼ぶ。BirthdayBook では NAME から DATE への部分関数 ($\pfun$)birthday と、known という名前の NAME の巾集合 ($\mathbb{P}$) 型の変数を宣言している。

区切り線より下部を述語部と呼び、変数や関数の制約や操作を記述する。本稿では述語部の 1 行を述語と呼ぶ。BirthdayBook では集合 known は関数 birthday の定義域 (dom) と等しいという制約が記述されている。

次に BirtdayBook にデータを登録する操作 AddBirthday をスキーマとして記述する。

AddBirthday

$\Delta BirthdayBook$
$n? : NAME$
$d? : DATE$

$n? \notin known$
$d' = birthday \cup \{n? \mapsto d?\}$

宣言文 $\Delta BirthdayBook$ は、操作 *AddBirthday* によりスキーマ *BirthdayBook* の状態 (つまり *BirthdayBook* の変数 birthday, known の値) が変化しうることを表している。$n?, d?$ のように名前の後に'?' が付加された変数は *AddBirthday* に対する入力変数を表す。述語部では操作の制約として、入力された $n?$ が既に登録されていないこと、操作を行った後の変数 *birthday* (スキーマ中では $birthday'$ と記述される) は、操作前の *birthday* に入力された人名と誕生日の組 $\{n? \mapsto d?\}$ を加えた集合と等しいことが記述されている。

同様に BirtdayBook から誕生日を検索する操作 FindBirthday は次のようにスキーマで記述される。

FindBirthday

$\Xi BirthdayBook$
$n? : NAME$
$d! : DATE$

$n? \in known$
$d! = birthday(n?)$

宣言文 $\Xi BirthdayBook$ は、操作 *FindBirthday* によりスキーマ *BirthdayBook* の状態が変化しないことを表している。$d!$ のように名前の最後に'!' が付加された変数は操作 *FindBirthday* からの出力変数を表す。操作の制約として、入力された $n?$ が登録されていること、出力 $d!$ は関数 *birthday* に $n?$ に適用した結果であることが記述されている。

すなわち、誕生日帳の仕様はデータ型宣言、BirthdayBook, AddBirthday, FindBirthday のスキーマから構成されることになる。

3. 仕様の類似度

本節では Z で記述された仕様の類似度の計算手法を提案する。仕様の例として誕生日帳と住所録 1、住所録 2、図書館システムを用いる (付録参照)。住所録は名前と住所、電話番号、メールアドレスを記録できるシステムで、誕生日帳の仕様を参考にして作成した。図書館システムは利用者が本の貸し借りを行えるシステムで文献 [2] を参考にした。本稿では誕生日帳を基準にして他の仕様との類似度を考察する。直観的には住所録 2, 住所録 1, 図書館システムの順で誕生日帳に類似していると言える (図書館システムは誕生日帳とはほとんど似ていないとも言える)。

Z ではシステムの状態や操作がスキーマとして記述されており、スキーマ中の宣言文、述語は意味的に強い関連がある。そこで本手法では類似度を計算する単位としてスキーマを選んだ。

一般に Z のスキーマは対象となるシステムの静的な状態を記述したスキーマ (以降、状態スキーマと呼ぶ) と、システムに対する動的な操作を記述したスキーマ (以降、操作スキーマと呼ぶ) に分類できる。状態スキーマではシステムを特徴づける変数や制約が記述される。操作スキーマでは入出力変数や事後変数 (変数名の最後に' が付けられた変数) が存在し、操作の事前条件や操作後の状態が記述される。例えば 2 節の誕生日帳の例では BirthdayBook が状態スキーマ、AddBirthday, FindBirthday が操作スキーマである。

本稿で提案する手法では次の手順で仕様の類似度を計算する。スキーマの種類に注目することで比較の際の組合せの数を減らすことができる。

(1) 状態スキーマを比較し、変数を対応づけ、状態スキーマの類似度を計算する。
(2) (1) で対応づけられた変数を利用して、操作スキーマの類似度を計算する。
(3) 状態スキーマ、操作スキーマを対応づけ、スキーマの類似度の平均値を仕様全体の類似度とする。

3.1 状態スキーマの類似度

次の手順にしたがい、2 つの状態スキーマ間で変数を対応づけ、状態スキーマの類似度を計算する。

(1) 型を参考にしてスキーマ間で変数の対応を仮定し、その変数が含まれている文 (宣言文、述語) を抽出する。
(2) (1) で抽出した宣言文を単一化する (変数名・型名を置換する)。単一化できれば (置換により 2 つのスキーマの宣言部が完全に一致すれば)、述語にも同じ置換を行い、(3) へ進む。単一化できなければ変数の対応が間違っていたので、(1) に戻り他の変数の対応でやり直す。
(3) (2) の結果得られた 2 つのスキーマの文の一致している割合を計算する。この割合が類似度となる。類似度は 1 が最大で、値が高い程スキーマが類似していることになる。

例えば、スキーマ BirthdayBook と AddressBook1 の類似度は次のように計算される。なお、スキーマ X の変数 x_1 とスキーマ A の変数 a_1, スキーマ X の変数 x_2 とスキーマ A の変数 a_2 の対応を $< x_1 \leftrightarrow a_1,$

$x_2 \leftrightarrow a_2 >$ と記述する。

(1) $< birthday \leftrightarrow addr, known \leftrightarrow known >$ の対応を仮定し、BirthdayBookから変数 $birthday, known$ を含む文を抽出する。同様に AddressBook1 から変数 $addr, known$ を含む文を抽出する。

$BirthdayBook'$

$birthday : NAME \pfun DATE$

$known : \mathbb{P} NAME$

$known = \mathrm{dom}\ birthday$

$AddressBook1'$

$addr : NAME \pfun ADDRESS$

$known : \mathbb{P} NAME$

$known = \mathrm{dom}\ addr$

$known = \mathrm{dom}\ tel$

$known = \mathrm{dom}\ mail$

(2) (1) で得られた BirthdayBook' に $\alpha/birthday$, $\beta/known$, $P/NAME$,$Q/DATE$ の置換☆を, AddressBook1'に $\alpha/addr, \beta/known, P/NAME$, $Q/ADDRESS$ の置換を施す。このとき、それぞれのスキーマの宣言文は単一化される。

$BirthdayBook''$

$\alpha : P \pfun Q$

$\beta : \mathbb{P} P$

$\beta = \mathrm{dom}\ \alpha$

$AddressBook1''$

$\alpha : P \pfun Q$

$\beta : \mathbb{P} P$

$\beta = \mathrm{dom}\ \alpha$

$\beta = \mathrm{dom}\ tel$

$\beta = \mathrm{dom}\ mail$

(3) (2) で得られた BirthdayBook", AddressBook1" には計 8 個の文があり、そのうち 3 個が一致している (つまり 6 個は同じものである)。したがって、変数を $< birthday \leftrightarrow addr, known \leftrightarrow known >$ と対応づけたときのスキーマ BirthdayBook と AddressBook1 の類似度は $6/8 = 0.75$ である。

同様にして $< birthday \leftrightarrow tel, known \leftrightarrow known >$ と $< birthday \leftrightarrow mail, known \leftrightarrow known >$ の対応でも類似度が計算でき、どちらも 0.75 となる。これ以外の変数の対応 (例えば $< birthday \leftrightarrow known, known \leftrightarrow tel >$) は、手順 2 で宣言文を単一化できないため、類似度は計算できない。

住所録 2 では状態を記述したスキーマは DATA と AddressBook2 の 2 つ存在するが、BirthdayBook と DATA では単一化できない。BirthdayBook と AddressBook2 では $< birthday \leftrightarrow book, known \leftrightarrow known >$ の対応で類似度が 1 になる。

誕生日帳の BirthdayBook と図書館システムの Library では $< birthday \leftrightarrow stock, known \leftrightarrow shelved >$ の対応のとき類似度は 0.57、$< birthday \leftrightarrow issued, known \leftrightarrow shelved >$ の対応のとき類似度は 0.40 となる。birthday と stock の対応が、birthday と issued の対応より類似度が高いのは、issued に関する述語が多く、手順 (1) でより多くの文が抽出されるためである。

3.2 操作スキーマの類似度

3.1 節の方法で対応づけた変数を利用して、次の手順で 2 つの操作スキーマの類似度を計算する。

(1) 操作スキーマの宣言文で、Δ や Ξ を用いて状態スキーマを参照している場合は、状態スキーマを展開する。スキーマを展開することで、操作スキーマに必要な変数などが得られる。

(2) (1) で得られたスキーマに、状態スキーマの類似度計算のときに行った変数・型の置換を施す。入出力変数がある場合はさらに単一化を行う。

(3) (2) の結果得られた 2 つのスキーマの文の一致している割合を計算する。この割合が類似度となる。類似度は 1 が最大で、値が高い程スキーマが類似していることになる。

例えば、$< birthday \leftrightarrow addr, known \leftrightarrow known >$ の対応のとき、スキーマ AddBirthday と Regist1 の類似度は次のように計算される。

(1) 参照している状態スキーマ (BirthdayBook, AddressBook1) を展開する (図 1)。

(2) (1) で得られた AddBirthday' に $\alpha/birthday$, $\beta/known$, $P/NAME$, $Q/DATE$, Regist1' に $\alpha/addr, \beta/known, P/NAME$, $Q/ADDRESS$ という置換を施す。さらに、入出力変数に対して AddBirthday' では $x?/n?, y?/d?$, Regist1' では $x?/n?, y?/a?$ という置換を施す (図 2)。

(3) (2) で得られたスキーマ AddBirthday", Search1" に対して類似度を計算する。全部で文は 22 個あり、7 個の文が一致しているので類似度は (7*2)/22=0.63 となる。

同様にして、他の操作スキーマ間の類似度を計算すると表 1 のようになる。

表 2 に誕生日帳と住所録 2 の操作スキーマの類

☆ 置換 a/x は名前 x を名前 a で置き換えることを表す

AddBirthday′

$birthday, birthday' : NAME \pfun DATE$
$known, known' : \mathbb{P} NAME$
$n? : NAME$
$d? : DATE$

$known = \mathrm{dom}\ birthday$
$n? \notin known$
$birthday' = birthday \cup \{n? \mapsto d?\}$

Regist1′

$addr, addr' : NAME \pfun ADDRESS$
$tel, tel' : NAME \pfun TEL$
$mail, mail' : NAME \pfun MAIL$
$known, known' : \mathbb{P} NAME$
$n? : NAME$
$a? : ADDRESS$
$t? : TEL$
$m? : MAIL$

$known = \mathrm{dom}\ addr$
$known = \mathrm{dom}\ tel$
$known = \mathrm{dom}\ mail$
$n? \notin known$
$addr' = addr \cup \{n? \mapsto a?\}$
$tel' = tel \cup \{n? \mapsto t?\}$
$mail' = mail \cup \{n? \mapsto m?\}$

図 1 AddBirthday と Regist1 の類似度 (1)

AddBirthday′

$\alpha, \alpha' : P \pfun Q$
$\beta, \beta' : \mathbb{P} P$
$x? : P$
$y? : Q$

$\beta = \mathrm{dom}\ \alpha$
$x? \notin \beta$
$\alpha' = \alpha \cup \{x? \mapsto y?\}$

Regist1′

$\alpha, \alpha' : P \pfun Q$
$tel, tel' : P \pfun TEL$
$mail, mail' : P \pfun MAIL$
$\beta, \beta' : \mathbb{P} P$
$x? : P$
$y? : Q$
$t? : TEL$
$m? : MAIL$

$\beta = \mathrm{dom}\ \alpha$
$\beta = \mathrm{dom}\ tel$
$\beta = \mathrm{dom}\ mail$
$x? \notin \beta$
$\alpha' = \alpha \cup \{x? \mapsto y?\}$
$tel' = tel \cup \{x? \mapsto t?\}$
$mail' = mail \cup \{x? \mapsto m?\}$

図 2 AddBirthday と Regist1 の類似度 (2)

表 1 誕生日帳と住所録 1 の類似度

	Regist1	Search1
AddBirthday	0.63	0.38
FindBirthday	0.33	0.43

表 3 誕生日帳と図書館システムの類似度

	Issue	Return	AddNTit	AddNMem
AddBirthday	0.26	0.30	0.47	0.19
FindBirthday	0.32	0.36	0.26	0.34

似度を示す。表 3 に誕生日帳と図書館システムの $<$ *birthday*↔*stock*, *known*↔*shelved* $>$ の対応のもとでの類似度を示す。

表 2 誕生日帳と住所録 2 の類似度

	Regist2	Search2
AddBirthday	1	0.50
FindBirthday	0.50	0.77

3.3 仕様全体の類似度

操作スキーマ、状態スキーマの類似度を求めたら、この中で最も類似度の高いスキーマの組合せを選び、類似度の平均値を求める。この値を仕様全体の類似度とする。

例えば、誕生日帳と住所録 1 では、(BirthdayBook, AddressBook1), (AddBirthday, Regist1), (FindBirthday, Search1) の組合せで、類似度は (0.75+0.63+0.43)/3 = 0.60 となる。同様に誕生日帳と住所録 2 では (BirthdayBook, AddressBook2), (AddBirthday, Regist2), (FindBirthday, Search2) の組合せで、類似度は (1+1+0.77)/3 = 0.92、誕生日帳と図書館システムでは (BirthdayBook, Library), (AddBirthday, AddNewTitle), (FindBirthday, Return) の組合せで、類似度は (0.57+0.47+0.36)/3 = 0.46 となる。

3.4 ま と め

誕生日帳と住所録 2 が類似度 0.92 と高く、一番類似していることになる。また、誕生日帳と図書館シス

テムでは類似度が 0.5 未満なので、文のうち半分以上が一致していないことになる。これらの結果は直観的な類似性と一致する。

本手法では、一致した文の割合を類似度とした。今回は対象とした仕様が単純だったため、完全に一致する文が多かったが、複雑な仕様では完全に一致することは少なくなると考えられる。文の中から特徴的な式を抽出し、文を単純化してから一致する割合を調べるなどの方法を今後提案する必要がある。

4. おわりに

本稿では Z で記述された仕様をスキーマ単位で比較を行い、類似度を計算する手法を提案した。スキーマを状態スキーマと操作スキーマに分類し、スキーマ間で対応をとり、類似度を計算した。

4.1 類似度の応用例

仕様の類似度の応用例として、再利用を支援するシステムが考えられる。

4.1.1 仕様の検索・再利用

新しいシステムの仕様を作成するとき、システムの状態といくつかの基本的な操作をスキーマとして記述し、それらのスキーマから類似した仕様を検索する。検索された仕様を参考にして、その他のスキーマを記述できる。

例えば、誕生日帳にデータの削除を行うスキーマを追加する場合、住所録 2 に削除のスキーマが定義されていればそれを参考にすることができる。

4.1.2 仕様のパターン抽出

複数のシステム間で類似度の高いスキーマは、仕様記述で頻繁に用いられるパターンと考えられる。パターンは再利用部品として利用できる。

4.2 今後の課題

今後の課題として以下が挙げられる。

4.2.1 他の基準による類似度

本稿ではスキーマを単位に類似度を計算した。その他の基準による類似度の提案、比較は今後の課題である。

例えば、図書館システムの AddNewTitle では入力変数 $id?$ と $t?$ の組を $stock$ に追加している。これは誕生日帳の AddBirthday で名前と誕生日の組を追加する部分と類似している。スキーマ単位の類似度では、このようなスキーマの一部分の類似性は分かりづらい。変数に注目することで、スキーマの特定の宣言や制約式に特化した類似度が計算できると考えられる。

また、文献 [11] では C++や Java のクラスライブラリについて、クラス名やメソッド名から類似度を計算する手法を提案している。この手法を Z の仕様に応用して、スキーマ名やスキーマ中の変数名から類似度を計算する方法も考えられる。

4.2.2 大規模な仕様への適用

本稿では単純な仕様について類似度を計算した。今後、より多くの仕様、また複雑な仕様に対して本手法を適用し、有効性を確認する必要がある。複雑な仕様については、スキーマの中から特徴的な式を抽出するなどの作業を行い、単純化してから比較を行う方法などが考えられる。

また、今回は手作業により仕様の類似度を計算したが、大規模な仕様に適用するにはツールを作成し、自動で計算を行うことが望ましい。われわれは Z による仕様記述の支援環境 ZWB(Z WorkBench) を提案している [12]。現在、ZWB を用いた類似度の計算ツールを作成中である。

謝　辞

ご討論いただいた南山大学の野呂昌満教授、張漢明講師、ならびに野呂ゼミのみなさまに感謝いたします。

本研究は南山大学パッヘ研究奨励金 I-A、堀情報科学財団の助成を受けた。

参考文献

1) Spivey, J. M.: *The Z Notation: A Reference Manual*, Prentice Hall International Series in Computer Science, 2nd edition (1992).
2) Potter, B.F., Sinclair, J.E. and Till, D.: *An Introduction to Formal Specification and Z*, Prentice Hall International Series in Computer Science, 2nd edition (1996).
3) Toyn, I. and McDermid, J. A.: CADiZ: An Architecture for Z Tools and its Implementation, *Software—Practice and Experience*, Vol. 25, No. 3, pp. 305–330 (1995).
4) Spivey, J. M.: *The fuzz Manual*, 34 Westlands Grove, Stockton Lane, York YO3 0EF, UK, 2nd edition (1992).
5) Xiaoping Jia: *ZTC: A Type Checker for Z – User's Guide*, Institute for Software Engineering, Department of Computer Science and Information Systems, DePaul University, Chicago, IL 60604, USA (1994).
6) Bowen, J. P. and Chippington, D.: Z on the Web using Java, In Bowen et al.[13], pp. 66–80.
7) Ciancarini, P., Mascolo, C. and Vitali, F.: Visualizing Z Notation in HTML Documents, In Bowen et al.[13], pp. 81–95.
8) Hewitt, M. A.: *PiZA:User Guide* (1997).
9) Lüth, C., Karlsen, E.W., Kolyang, Westmeier, S. and Wolff, B.: HOL-Z in the UniForM-

Workbench – A Case Study in Tool Integration for Z, In Bowen et al.[13], pp. 116–134.
10) Bloesch, A., Kazmierczak, E., Kearney, P. and Traynor, O.: Cogito: A Methodology and System for Formal Software-Development, *International Journal of Software Engineering and Knowledge Engineering*, Vol. 5, No. 4, pp. 599–617 (1995).
11) Michail, A. and Notkin, D.: Assessing software libraries by browsing similar classes, functions and relationships, *Proceedings of the 1999 International Conference on Software Engineering*, pp. 463–472 (1999).
12) 蜂巣吉成: リポジトリを用いたソフトウェア仕様記述の支援環境, 情報処理学会研究報告, Vol.2000, No. 4, pp. 99–104 (2000).
13) Bowen, J. P., Fett, A. and Hinchey, M. G.(eds.): *ZUM'98: The Z Formal Specification Notation, 11th International Conference of Z Users, Berlin, Germany, 24–26 September 1998*, Lecture Notes in Computer Science, Vol. 1493, Springer-Verlag (1998).

付　　録

A.1 仕 様 例

A.1.1 誕 生 日 帳

$[NAME, DATE]$

BirthdayBook

$$birthday : NAME \pfun DATE$$
$$known : \mathbb{P}\, NAME$$
$$known = \mathrm{dom}\, birthday$$

AddBirthday

$$\Delta BirthdayBook$$
$$n? : NAME$$
$$d? : DATE$$
$$n? \notin known$$
$$birthday' = birthday \cup \{n? \mapsto d?\}$$

FindBirthday

$$\Xi BirthdayBook$$
$$n? : NAME$$
$$d! : DATE$$
$$n? \in known$$
$$d! = birthday(n?)$$

A.1.2 住 所 録 1

名前、住所、電話番号、メールアドレスが記録できる住所録の仕様を Z で記述した。スキーマ Regist1 は新たなデータを登録し、Search1 は名前からデータを探す。

$[NAME, ADDRESS, TEL, MAIL]$

*AddressBook*1

$$addr : NAME \pfun ADDRESS$$
$$tel : NAME \pfun TEL$$
$$mail : NAME \pfun MAIL$$
$$known : \mathbb{P}\, NAME$$
$$known = \mathrm{dom}\, addr$$
$$known = \mathrm{dom}\, tel$$
$$known = \mathrm{dom}\, mail$$

*Regist*1

$$\Delta AddressBook1$$
$$n? : NAME$$
$$a? : ADDRESS$$
$$t? : TEL$$
$$m? : MAIL$$
$$n? \notin known$$
$$addr' = addr \cup \{n? \mapsto a?\}$$
$$tel' = tel \cup \{n? \mapsto t?\}$$
$$mail' = mail \cup \{n? \mapsto m?\}$$

*Search*1

$$\Xi AddressBook1$$
$$n? : NAME$$
$$result! : NAME \times ADDRESS \times TEL \times MAIL$$
$$n? \in known$$
$$result! = (n?, addr(n?), tel(n?), mail(n?))$$

A.1.3 住 所 録 2

名前、住所、電話番号、メールアドレスが記録できる住所録の仕様を Z で記述した。この仕様では住所、電話番号、メールアドレスの組をスキーマとして定義した。

$[NAME, ADDRESS, TEL, MAIL]$

DATA

$$address : ADDRESS$$
$$tel : TEL$$
$$mail : MAIL$$

*AddressBook*2

$$book : NAME \pfun DATA$$
$$known : \mathbb{P}\, NAME$$
$$known = \mathrm{dom}\, book$$

*Regist*2

$\Delta AddressBook2$

$n? : NAME$

$d? : DATA$

$n? \notin known$

$book' = book \cup \{n? \mapsto d?\}$

*Search*2

$\Xi AddressBook2$

$n? : NAME$

$result! : NAME \times DATA$

$n? \in known$

$result! = n? \mapsto book(n?)$

A.1.4 図 書 館

簡単な図書館システムの仕様を Z で記述した。図書館システムは次の特徴を持つ。

- 図書館は本 (Title) の在庫と、登録された利用者 (Reader) からなる
- それぞれの本 (Title) には固有の識別子 (ID) がある
- 利用者に貸し出し中の本があり、残りの本は図書館の本棚にあり貸し出し可能である
- すべての利用者に同時に借りることができる本の最大数 (maxloans) が決められている

以上の特徴をスキーマ Library で定義し、さらに図書館の操作として次のスキーマを定義した。

Issue　利用者に本を貸す

Return　利用者が借りた本を返す

AddNewTitle　図書館に新しい本を登録する

AddNewReader　新しい利用者を登録する

$[ID, Title, Reader]$

$maxloans : \mathbb{N}$

Library

$stock : ID \pfun Title$

$issued : ID \pfun Reader$

$shelved : \mathbb{P}\, ID$

$readers : \mathbb{P}\, Reader$

$shelved \cup \mathrm{dom}\, issued = \mathrm{dom}\, stock$

$shelved \cap \mathrm{dom}\, issued = \phi$

$\mathrm{ran}\, issued \subseteq readers$

$\forall r : readers \bullet \#(issued \rhd \{r\}) \leq maxloans$

Issue

$\Delta Library$

$id? : ID$

$r? : Reader$

$id? \in shelved$

$r? \in readers$

$\#(issued \rhd \{r?\}) < maxloans$

$issued' = issued \oplus \{id? \mapsto r?\}$

$stock' = stock$

$readers' = readers$

Return

$\Delta Library$

$id? : ID$

$id? \in \mathrm{dom}\, issued$

$issued' = \{id?\} \lhd issued$

$stock' = stock$

$readers' = readers$

AddNewTitle

$\Delta Library$

$id? : ID$

$t? : Title$

$stock' = stock \cup \{id? \mapsto t?\}$

$shelved' = shelved \cup \{id?\}$

$issued' = issued$

$readers' = readers$

AddNewReader

$\Delta Library$

$r? : Reader$

$r? \notin readers$

$readers' = readers \cup \{r?\}$

$stock' = stock$

$shelved' = shelved$

$issued' = issued$