NAIST-IS-MT351034

修士論文

抽象的順序機械型代数的仕様からのドキュメント生成システム

工藤 朋之

1995年2月20日

奈良先端科学技術大学院大学
情報科学研究科 情報処理学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科において
修士 (工学) 授与の要件として提出された修士論文である。

提出者： 工藤 朋之

指導教官： 伊藤 実 教授
嵩 忠雄 教授
関 浩之 助教授

抽象的順序機械型代数的仕様からのドキュメント生成システム*

工藤 朋之

内容梗概

本論文では, 代数的仕様の部分クラスである抽象的順序機械型代数的仕様から, 縮退遷移グラフおよび仕様の公理の説明文からなるドキュメントを生成するシステムの試作について述べる. 縮退遷移グラフとは, 抽象的順序機械の全状態空間を分割することにより得られた各部分状態空間と, それらの間の状態遷移を示す有向グラフである. 縮退遷移グラフは順序機械の大局的な状態遷移を示しており, 仕様の理解しやすさを向上させると考えられる. 本システムは, 利用者による状態空間分割を支援する機能や, 順序機械の動作の様子を縮退遷移グラフ上に表示する機能を持つ. また, 本システムは, 仕様で記述された公理を入力とし, 各部分状態空間からの状態遷移を説明する日本語文を生成する. 説明文の生成においては, 状態遷移モデルに基づく表現を用いることにより, もとの抽象的順序機械型代数的仕様の意味を説明文に正確に反映させる. また, 生成された説明文を利用者に提示した後, 関数名, タイプ名などに対して, 利用者が "別名" を指定できるようにしており, より理解しやすい説明文を生成することが可能である. 本論文では, OSI セションプロトコルの仕様の一部を入力としてドキュメントを生成した結果についても述べる.

キーワード

抽象的順序機械, 代数的仕様, 縮退遷移グラフ, ドキュメント, 文生成

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 修士論文, NAIST-IS-MT351034, 1995 年 2 月 20 日.

A Document Generation System for Abstract Sequential Machine Specifications*

Tomoyuki Kudoh

Abstract

In this paper, we describe a system which generates a document of a given algebraic specification of an abstract sequential machine. The document consists of a reduced transition graph and natural language sentences which explain the axioms in the specification. A reduced transition graph is a directed graph representing transitions among subspaces of states, and can help a user understand the global behavior of the abstract sequential machine. The system has functions of (1) guiding a user in generation of a reduced transition graph, and (2) showing the dynamic behavior of the abstract sequential machine on the graph. The natural language sentences generated by the system follow state transition models so that they explain the meanings of the axioms precisely. Moreover, a user can specify "aliases" of constants, function symbols, and type names in a specification to make the sentences easy to understand. This paper also presents an experimental result of generating a document of a part of the OSI session protocol specification.

Keywords:

abstract sequential machine, algebraic specification, document, reduced transition graph, sentence generation

*Master's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT351034, February 20, 1995.

目 次

図目次

表目次

1. まえがき

代数的仕様の部分クラスとして, 抽象的順序機械型代数的仕様と呼ばれるクラスが提案されている[5]. 抽象的順序機械型代数的仕様 (以下, 順序機械仕様と略記) では, 記述の対象となる系の状態を表すデータタイプを導入し, それを以下のように定義する. まず, 仕様で用いる関数を状態遷移関数, 状態成分関数, 外部操作関数に分類する. そして, 関数の意味を定義する以下のような公理を記述する.

- ある状態遷移関数の表す遷移が起こった後の各状態成分関数の値を, その遷移が起こる前の状態成分関数の値を用いて定義する.
- 外部操作関数をいずれかの状態遷移関数に対応づける.

順序機械仕様では, 各状態 σ を $\langle STATE8, true, 10\rangle$ のように状態成分値の組で直接表現するのではなく, σ を引数とするいくつかの状態成分関数によって表す. このため, 仕様に新たな状態成分を追加する際, 状態成分関数とそれに関する公理を追加するだけでよく, 既に記述した部分を変更する必要がない, という利点をもつ.

既に順序機械仕様を用いて, スクリーンエディタ, コンパイラ, 通信プロトコルなどの仕様記述とその詳細化, 仕様の正当性の検証が行なわれ, 順序機械仕様が実用規模のソフトウェア開発にも有効であることが確認されてきた[7]. また, 順序機械仕様から C プログラムへのコンパイラも作成されており, OSI セションプロコトル[1]の順序機械仕様を例としたコンパイル結果についても報告されている[9].

本論文では, 代数的仕様記述法を用いた以下のようなソフトウェア開発プロセスを想定している.

- 設計者は, 対象とするシステムの代数的仕様を記述し, 順序機械仕様まで詳細化していく.
- 得られた順序機械仕様に基づき, 実装者がコーディングを行なうか, または, コンパイラを用いてプログラムに変換する.
- システムに対する要求の変更は, まず仕様に反映させ, これに基づいてプログラムの保守を行なう.

さて, 順序機械仕様は前述のような特長をもつ反面, 例えば, 通信プロトコル仕様における通信フェーズ間の遷移などのように, 大規模な仕様における大局的な状態遷移を見い出すのが困難なことがある. そこで, 特定の状態成分関数の値に

着目して全状態空間を分割し, 各部分状態空間の間の状態遷移が示されれば, コーディングや保守の際に, 仕様の理解の助けとなると考えられる.

本論文では, このような部分状態空間の間の状態遷移を表すものとして, 縮退遷移グラフを定義している. 縮退遷移グラフとは, 頂点が部分状態空間, 辺が部分状態空間の間の状態遷移, 辺のラベルが遷移条件を表す有向グラフである.

さらに本論文では, 以下のようなドキュメント生成システム[3][4] について述べている.

- 入力された順序機械仕様に対し, 利用者と対話的に状態空間の分割を繰り返すことにより, 縮退遷移グラフを作成する.
- 順序機械の動的な振舞いを縮退遷移グラフ上に示す.
- 縮退遷移グラフの頂点および辺に対し, 日本語による公理の説明文を生成して関連づけることによって, 全体をハイパーテキスト化する. .

状態空間の分割において, 利用者は分割する頂点と分割の基準となる状態成分関数を指定する. 例えば, 頂点 S と, TRUE または FALSE を値とする状態成分関数 f を指定すると, S は, 3 つの頂点 $S_{f,\mathrm{TRUE}}$, $S_{f,\mathrm{FALSE}}$, $S_{f,\mathrm{UNDEFINED}}$ に分割される. ここで, $S_{f,\mathrm{TRUE}}$ は, S の表す部分状態空間に属する状態のうち, f の値が TRUE であるようなもの全体からなる部分状態空間を表す. 状態空間を分割した時, 分割の基準となる状態成分関数が異なれば, 一般に, 異なる縮退遷移グラフが得られる. 仕様の理解の助けとなる縮退遷移グラフを構成することが望ましいが, 仕様の規模が大きくなると, 適切な状態成分関数を選択するのは容易ではない. そこで, 本研究では, 分割の基準となる状態成分関数の候補を求める方法を考察した. 本方法では, 公理における各状態成分関数の出現に基づいて, 分割の基準となる状態成分関数の候補を求める. この方法に基づいて作成された状態空間の分割支援ツール [6] を用いて, OSI セションプロトコルの順序機械仕様の一部に対して, 縮退遷移グラフを作成した. その結果, 本方法により求められた候補が, 状態空間の分割の際に指定される状態成分関数として適していることが確認された.

また, 本システムでは, 日本語による公理の説明文の生成において, 状態遷移モデルに基づく表現を用いることにより, もとの順序機械仕様の意味を説明文に正確に反映させることができる. さらに, 生成された説明文を確認した後, 利用者は関数名, タイプ名などに対して “別名” を指定できる. これにより, 理解しやすい説明文を生成することが可能となった. 本論文では, OSI セションプロトコルの順序機械仕様の公理を入力として, 公理の説明文を生成した結果についても述べる.

2. 順序機械仕様

2.1 代数的仕様

本稿では, 始記号を指定しない文脈自由文法 G と公理の集合 AX の 2 字組 $SPEC = (G, AX)$ を代数的仕様と呼ぶ[2].

$G = (N, \Sigma, P)$ とし, N, Σ, P をそれぞれ非終端記号の集合, 終端記号の集合, 構文規則の集合とする. G において, $D \in N$ から生成される終端記号列の集合を $L_G[D]$ とし, $L_G = \bigcup_{D \in N} L_G[D]$ とする. L_G の元を代数的仕様 $SPEC$ の表現式 (あるいは, 式) と呼ぶ. 各非終端記号はデータタイプ (ソート) に対応しており, $exp \in L_G[D]$ のとき, 表現式 exp のデータタイプ (あるいは, 型) は D であると言う. 各終端記号は, 関数 (定数) 記号に対応している. 各構文規則は, 関数の構文宣言 (関数名, および引数と関数値のデータタイプの宣言) に対応している. 例えば, Int, Real を非終端記号とし, floor, "(", ")" を終端記号としたとき, Int → floor(Real) という構文規則は, 関数 floor が Real 型の引数をとり Int 型の値を返す 1 引数関数であることを宣言している. 以下では, 同じ意味を表すのに floor: Real → Int のような表記をすることがある.

公理は変数を含む表現式の対 $exp = exp'$ の形で記述する. ただし, 公理の左辺または右辺に現れる各変数 x には, 文脈自由文法 G の非終端記号 D_x が一つ対応づけられ, 変数 x には, D_x から生成される表現式が任意に代入できると考える. この非終端記号 D_x を x のタイプ指定と呼ぶ. AX 中のすべての公理を満たす L_G 上の最小の合同関係を代数的仕様 $SPEC$ の指定する合同関係と言い, $\equiv_{SPEC}$ で表す. 文脈より $SPEC$ が明らかな時は, 単に $\equiv$ と書く.

2.2 順序機械仕様の定義

本節以降では, 整数型や Bool 型などの "基本データタイプ" に関する仕様 $SPEC_0 = (G_0, AX_0)$ (ただし $G_0 = (N_0, \Sigma_0, P_0)$) が定まっていると仮定する. さらに, 各 $D \in N_0$ について, 構成子表現式, 例えば Bool 型における TRUE, FALSE のように値を表す考えられる式の集合 $CONST[D]$ が指定されているとする. ただし, 任意の $exp, exp' \in CONST[D]$ について $exp \not\equiv_{SPEC_0} exp'$, すなわち $SPEC_0$ は無矛盾とする.

定義 1 代数的仕様 $SPEC = (G, AX)$ *(ただし $G = (N, \Sigma, P)$)* が以下の条件をすべて満たすとき, $SPEC$ を順序機械仕様と呼ぶ.

1. $SPEC$ は $SPEC_0$ を部分仕様として含む.

表 1 順序機械仕様で用いられる関数

1. 状態成分関数: 各状態における状態成分を表す関数である. State 型の引数をちょうど 1 個もつ. 関数値の型は基本データタイプである.
2. 外部操作関数: 外部から順序機械に与えられる操作を表す関数である. State 型の引数をちょうど 1 個もつ. 関数値の型は State である.
3. 状態遷移関数: 順序機械の状態を遷移させる関数である. State 型の引数をちょうど 1 個もつ. 関数値の型は State である.
4. 補助関数: 基本データタイプ上の演算を表す関数や, 多数の状態成分関数を含む複雑な式を簡潔に記述するためのマクロである. State 型の引数を高々1 個もつ. 関数値の型は基本データタイプである.

2. G は, 抽象的状態を表す表現式を生成する非終端記号 (State とする) をもつ. ただし, State $\in (N - N_0)$ である.

3. $(\Sigma - \Sigma_0)$ に属する関数は, 表 1に示す 4 種類の関数に分類される. 簡単のため, これらの関数が State 型の引数をもつ場合は, その引数は最右の引数とする.

4. $(AX - AX_0)$ に現れる公理は, 以下の条件をすべて満たす.

   (a) 公理の左辺は線形で, かつ重なりをもたない[8].

   (b) 公理の右辺に現れる変数は, その左辺にも現れる.

   (c) O を状態成分関数, T を外部操作関数, T' を状態遷移関数, A を補助関数とする. さらに, s を State 型の変数とし, $x_1, \ldots, x_n$, $y_1, \ldots, y_m$, $z_1, \ldots, z_k$, および $w_1, \ldots, w_\ell$ を State 以外の型の変数とする. 各関数の意味を定義する公理は以下の形式である.

   形式 (1) $T(z_1, \ldots, z_k, s) = \ldots$

   形式 (2) $O(x_1, \ldots, x_n, T'(y_1, \ldots, y_m, s)) = \ldots$

   形式 (3) $A(w_1, \ldots, w_\ell) = \ldots$ または

   $A(w_1, \ldots, w_\ell, s) = \ldots$

   ただし, 形式 (1) の公理の右辺は, 以下のようにして定義される表現式の部分集合 $\mathcal{E}$ $(\subseteq L_G)$ の要素でなければならない.

   i. T' が状態遷移関数ならば, $T'(y_1, \ldots, y_m, s) \in \mathcal{E}$ (条件 (b) より, 各変数 y_i $(1 \leq i \leq m)$ は $z_1, \ldots, z_k$ のいずれかと一致する).

ii. $exp_1, exp_2 \in \mathcal{E}$ であり, p が状態成分関数と補助関数のみから成る *Bool* 型の表現式ならば, "if (p) exp_1 else exp_2" $\in \mathcal{E}$. ただし, *if* 式は公理

$$\text{if (TRUE) } exp_1 \text{ else } exp_2 = exp_1$$
$$\text{if (FALSE) } exp_1 \text{ else } exp_2 = exp_2$$

で定義されるとする.

iii. $\mathcal{E}$ は, 上の *2* 条件を満たす最小の集合である.

また, 形式 *(2), (3)* の公理の右辺は, 状態成分関数と補助関数のみから成る表現式でなければならない. □

形式 (1) の公理では, 順序機械が外部操作 T を受けたときに起こすべき状態遷移を, 右辺の if 式の条件分岐によって対応づける. 形式 (2) の公理では, 状態遷移関数 T' で表される状態遷移後の状態成分関数 O の値を, 遷移前の状態における状態成分関数の値を用いて定義する.

図 1 に順序機械仕様の例を示す. この仕様では, 抽象的状態を表す表現式を生成する非終端記号として SPM を用いている. 公理 1 では, 順序機械 (この仕様の場合はプロトコル機械) が外部操作 ssynM_req を受けたとき, 状態成分関数 phase, ssynM, vrsp および補助関数 p13 の値が if 式の条件を満たすならば, ssynM_req1 で表される状態遷移を起こすことを規定している (図 1では, p13 を定義する公理は省略している). また, 公理 2 では, 状態遷移 ssynM_req1 が起こった直後の状態成分関数 vm の値は, 遷移前の vm の値に 1 を加えた値であると規定している.

順序機械仕様では, 状態をその成分値の組で直接表すのでなく, 各状態 s における成分値を, s を引数とする状態成分関数によって表す. そのため, 状態に新たな状態成分を追加する際には, 状態成分関数とそれに関する公理を局所的に追加するだけでよく, 既存の公理を変更する必要がないという特長がある. 一方, 仕様の規模が大きくなると, 例えばプロトコル仕様における通信のフェーズ間の遷移などのような, 大局的な状態遷移を見出すのが困難になる. したがって, 特定の状態成分関数の値に着目して全状態空間を分割し, 各部分状態空間の間の状態遷移が示されれば, 仕様の理解の助けとなると考えられる. 次節以降では, 与えられた順序機械仕様の状態空間をこの意味で構造化し, ドキュメントを生成する手法について述べる.

ただし, 本稿では, ドキュメントとは, 以下の 2 つから構成されるものを指す.

- 分割された各部分状態空間を頂点に, それらの間の状態遷移を辺に対応させたグラフ (縮退遷移グラフと呼ぶ).

- 各部分状態空間ごとに, 観測関数の値を考慮して簡単化された公理について, 説明を行なう日本語文.

```
Specification    SPM
Observation Functions       /* 状態成分関数 */
    vm, va : SPM -> Nat
    ssynM, awaitSSYNMrsp, vsc : SPM -> Bool
    phase    : SPM -> Phase_name
    vrsp     : SPM -> R_type
Interface Functions         /* 外部操作関数 */
    ssynM_req    : Ssp,SPM -> SPM
State Transition Functions /* 状態遷移関数 */
    ssynM_req1   : Ssp,SPM -> SPM
Axioms
  s :SPM;
  pa_SSYNMreq : SSP;
/* 形式 (1) の公理 */
ssynM_req(pa_SSYNMreq,s) =
    if(phase(s) == DATA_TRANS)
      if(!(ssynM(s)) && vrsp(s) == NO && p13(s))
            ssynM_req1(pa_SSYNMreq,s);
                                        /* 公理 1 */
/* 形式 (2) の公理 */
vm(ssynM_req1(pa_SSYNMreq, s)) = vm(s)+1;
                                        /* 公理 2 */
va(ssynM_req1(pa_SSYNMreq, s)) = if(vsc(s)) vm(s);
                                          else va(s);
ssynM(ssynM_req1(pa_SSYNMreq, s)) = TRUE;
awaitSSYNMrsp(ssynM_req1(pa_SSYNMreq, s)) = FALSE;
vsc(ssynM_req1(pa_SSYNMreq, s)) = FALSE;
phase(ssynM_req1(pa_SSYNMreq, s)) = phase(s);
vrsp(ssynM_req1(pa_SSYNMreq, s)) = vrsp(s);
```

図 1 OSI セションプロトコルの順序機械仕様 (大同期点設定部分の一部)

3. 縮退遷移グラフ

縮退遷移グラフとは, 頂点が部分状態空間を表し, 辺が部分状態空間の間の遷移を表す有向グラフである. 本システムは利用者と対話的に全状態空間を部分状態空間に分割することにより, 縮退遷移グラフを作成する.

以下では, $SPEC = (G, AX)$(ただし, $G = (N, \Sigma, P)$) を順序機械仕様とし, $D \in N$ を基本データタイプとする. $CONST[D]$ が有限集合のとき, D は有限データタイプであるという. 順序機械仕様 $SPEC$ で定義される状態成分関数のうち, 関数値が有限データタイプであり, かつ State 型以外の引数をもたない状態成分関数を $O_1, ..., O_m$ とし, O_i の関数値のデータタイプを D_i とする. 未定義を表す記号 $nil \notin N \cup \Sigma$ を導入し, 各 i について, $C_i = \{nil\} \cup CONST[D_i]$ とする. $L_G[D_i] \times C_i$ 上の関係 $\approx$ を以下のように定義する.

> $exp \in L_G[D_i]$ とする. ある $con \in CONST[D_i]$ に対して $exp \equiv_{SPEC} con$ の時, $exp \approx con$ と書く. また, 任意の $con \in CONST[D_i]$ に対して $exp \not\equiv_{SPEC} con$ の時, $exp \approx nil$ と書く.

以降, L_G[State] を U と略記する.

3.1 縮退遷移グラフの定義

$SPEC$ を順序機械仕様とする. 以下の条件 (1), (2) を満たす任意の有向グラフ $\mathcal{G}_{SPEC} = (V, E)$ (V は頂点の集合, E は有向辺の集合) を $SPEC$ の縮退遷移グラフと呼ぶ.

(1) $V = \{v_1, v_2, ..., v_n\}$ は 以下の条件を満たす U のある分割 $\mathcal{S} = \{S_1, S_2, ..., S_n\}$ ($\bigcup_{1 \le i \le n} S_i = U$, $S_i \cap S_j = \phi (i \neq j)$) と一対一対応する. ただし, $v_j (1 \le j \le n)$ に対応する集合を S_j とする. 各 S_j を U の部分状態空間と呼ぶ. 各頂点 $v_j \in V$ に対して, いくつかの状態成分関数の値を指定するための次のような集合 $BIND[v_j]$ が与えられているとする.

$$BIND[v_j] = \{(O_{v_j,1}, c_{v_j,1}), (O_{v_j,2}, c_{v_j,2}), ..., (O_{v_j,r}, c_{v_j,r})\}$$

ここで, $O_{v_j,k} \in \{O_1, ..., O_m\}$, $O_{v_j,k} \neq O_{v,l} (k \neq l)$, $O_{v_j,i} = O_h$ とすると $c_{v_j,i} \in C_h$ である. $(O_i, c) \in BIND[v_j]$ は, v_j に対応する部分状態空間 S_j において, 状態成分関数 O_i の値が c に定まっていることを表す. 形式的には, S_j を以下のように定める.

$$S_j = \{\sigma \mid \sigma \in U \text{ かつ, } (O_i, c) \in BIND[v_j] \text{ ならば } O_i(\sigma) \approx c\}$$

以降簡単のため, 頂点 v_j と部分状態空間 S_j を同一視する.

(2) 辺 $S \overset{LABEL[S,S']}{\longrightarrow} S'$ は S から S' へ "遷移可能" である時かつその時のみ存在し, $LABEL[S,S']$ は "遷移可能" であるための条件を表す. 厳密には, 以下のように定義する.

$LABEL[S,S'] =$
$\{(T(z_1,...,z_k,s), COND\langle z_1,...,z_k,s\rangle) \mid$
$T(z_1,...,z_k,s) = exp'$ は AX に属する形式 (1) の公理. また, $COND\langle z_1,...,z_k,s\rangle$ は, 変数としては $z_1,...,z_k,s$, 関数としては状態成分関数と補助関数のみを含む Bool 型の式で, 「任意の $\sigma \in S$, z_i に代入可能で変数を含まない任意の a_i について, $COND\langle a_1,...,a_k,\sigma\rangle \equiv$ TRUE $\Leftrightarrow T(a_1,...,a_k,\sigma) \in S'$」を満たす. $\}$

3.2 状態空間の分割

$SPEC = (G, AX)$ を任意の順序機械仕様とする. $V_0 = \{U\}$, $BIND[U] = \phi$, $LABEL[U,U] = \{(T, \text{TRUE}) \mid T$は外部操作関数 $\}$, $E_0 = \{(U \overset{LABEL[U,U]}{\longrightarrow} U)\}$ とするとき, $\mathcal{G}_{SPEC,0} = (V_0, E_0)$ は, 頂点がただ一つであり, 状態空間の分割を全く行なっていない縮退遷移グラフである.

$\mathcal{G}_{SPEC,l} = (V_l, E_l)$ を, $\mathcal{G}_{SPEC,0}$ から状態空間の分割を l 回行なって得られた縮退遷移グラフとする. 分割すべき頂点 (部分状態空間) S と分割の基準となる有限データタイプの状態成分関数 O_i が指定された時に, $\mathcal{G}_{SPEC,l}$ から状態空間の分割を行なって得られる縮退遷移グラフ $\mathcal{G}_{SPEC,l+1} = (V_{l+1}, E_{l+1})$ を以下のように定める. まず, $S_c = \{\sigma | \sigma \in S, O_i(\sigma) \approx c\}$ とし,

$$V_{l+1} = (V_l - \{S\}) \cup \{S_c | c \in C_i\}$$
$$BIND[S_c] = BIND[S] \cup \{(O_i, c)\}$$

とする. また, 3.1 節の辺の定義 (2) にしたがって, すべての $S_c (c \in C_i)$ とすべての $S' \in V_l - \{S\}$ について $LABEL[S_c, S_c]$, $LABEL[S_c, S']$ および $LABEL[S', S_c]$ を求め, E_{l+1} を以下のように定める.

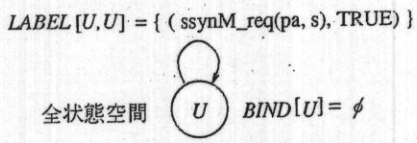

図 2 全状態空間の状態遷移を表す縮退遷移グラフ

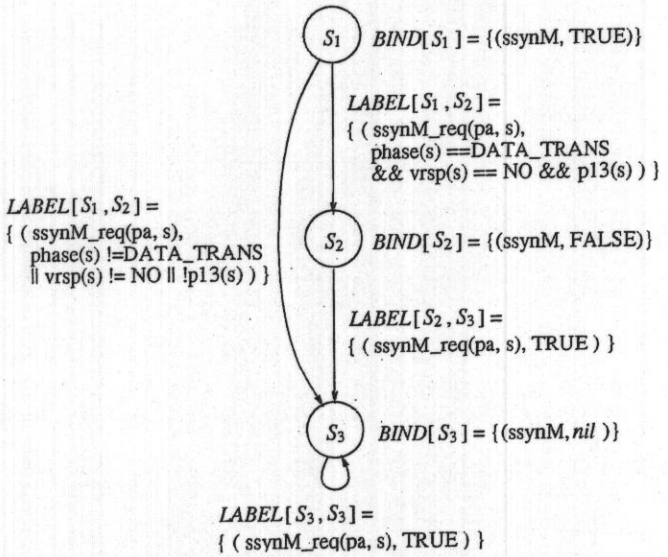

図 3 図 2 のグラフから状態空間分割を行なって得られた縮退遷移グラフ

$$
\begin{aligned}
E_{l+1} = \quad & E_l \\
& - \{ Z \mid Z = S \overset{LABEL[S,S'']}{\longrightarrow} S'' \text{ または}, \\
& \qquad Z = S'' \overset{LABEL[S'',S]}{\longrightarrow} S,\ Z \in E_l,\ S'' \in V_l \} \\
& \cup \{ X \overset{LABEL[X,Y]}{\longrightarrow} Y \mid \\
& \qquad (X,Y) \text{ は } (S_c,S_c),(S_c,S') \text{ または } (S',S_c), \\
& \qquad \text{ここで}, c \in C_i,\ S' \in V_l - \{S\} \}
\end{aligned}
$$

2 節における図 1 の仕様の全状態空間の状態遷移を表す縮退遷移グラフ (図 2) を分割する場合を考える. 利用者が 図 1 の頂点 U と状態成分関数 ssynM(s) を指定した時, 図 3 のような縮退遷移グラフが作成される. 直観的には, 全状態空間を, それぞれ ssynM が FALSE, TRUE, および未定義である部分状態空間に分割している.

3.3 状態空間分割の支援法

状態空間を分割した時, 分割の基準となる状態成分関数が異なれば, 一般に, 異なる縮退遷移グラフが得られる. 縮退遷移グラフは, 順序機械の大局的な状態遷移の理解を助けるものが良い. 状態空間の分割の際には, このような縮退遷移グラフを作成するのに適切な状態成分関数を分割の基準とすることが望ましい.

例えば, OSI セションプロトコル仕様においては, どの状態遷移も状態成分関数 phase の値に依存する. 一方, phase 以外の状態成分関数については, その関数の値に依存しない状態遷移が存在する. ここで, phase 以外の状態成分関数 o を基準として, 全状態空間を分割する場合を考える. この場合, 例えば phase が IDLE_AND_NO_TC の場合に起こる遷移のように, ある大局的な状態遷移に対応する辺が, 分割により得られたすべての頂点を始点として作成される. そのため, 利用者は大局的な遷移を理解し難くなると考えられる. そこで, 全状態空間を分割する場合, phase を分割の基準とするのが適切であると考えられる.

以上のように, 状態空間を分割する場合には, 大局的な状態遷移の理解を助けるように, 分割の基準となる状態成分関数を選択することが望まれる. しかし, 仕様の規模が大きくなると, このような選択は困難になる. そこで, 本論文では, 分割の基準となる状態成分関数の候補を求めることにより, 状態空間の分割を支援する方法を提案する.

大局的な状態遷移の理解を助けるように状態空間の分割を行なうには, 各状態遷移の遷移条件に現れる状態成分関数の値を基準にして状態空間を分割するのが良いと思われる. 本論文では, 形式 (1) の公理の条件部に現れる状態成分関数の出現に基づいて, 分割の基準となる状態成分関数の候補を求める.

まず, 頂点 S について簡約された形式 (1) の公理のうち, 外部操作関数 T を定義する以下のような公理があると仮定する.

$$
\begin{aligned}
T(z_1, ..., z_k, s) \;=\; & \text{if}\,(p_1)\; T_1'(..., s) \\
& \text{else if}\,(p_2)\; T_2'(..., s) \\
& \qquad \vdots \\
& \text{else if}\,(p_n)\; T_n'(..., s) \\
& (T_1', T_2', ...T_n' \text{は状態遷移関数})
\end{aligned}
$$

このとき, S における "T に対する遷移条件" の集合 $P_{S,T}$ を以下のように定める.

$$
\begin{aligned}
P_{S,T} \;=\; & \{p_1,\; !p_1 \,\&\&\, p_2,\; \ldots,\; !p_1 \,\&\&\, \cdots \,\&\&\, !p_{n-1} \,\&\&\, p_n, \\
& !p_1 \,\&\&\, \cdots \,\&\&\, !p_{n-1} \,\&\&\, !p_n\}
\end{aligned}
$$

そして, 頂点 S における "遷移条件" の集合 P_S を $\bigcup P_{S,T}$ と定める. とりうる値が有限である状態成分関数のうち, P_S 中の遷移条件に現れているものの集合を $\mathcal{O}_S$ とする. $\mathcal{O}_S$ の要素である状態成分関数 O の分割の基準としての "適切さ" を次のように考える.

$O, O' \in \mathcal{O}_S$ と仮定する. P_S に含まれるすべての遷移条件に対し, O' が現れている時に必ず O が現れているとする. このとき, O' に依存せずに O のみに依存する遷移条件が P_S に存在する可能性がある. このような遷移条件 p が存在する時に O' を基準として S を分割すると, p が成り立つ時の遷移に対応する辺が, 分割により得られたすべての頂点を始点として作成され, 利用者は大局的な遷移を理解し難くなると考えられる. したがって, S において, O の適切さは O' のそれと同じか上であると考えられる.

本論文では, 上の意味で適切さが "極大な" 状態成分関数を, 分割の基準となる状態成分関数の候補とする. 形式的には, 頂点 S の分割の基準としての適切さを表す, $\mathcal{O}_S$ 上の関係 $\geq_S$ を以下のように定義する.

> P_S に含まれるすべての遷移条件に対し, O' が現れている時に必ず O が現れるならば, $O \geq_S O'$ が成り立つ.

定義より明らかに反射律と推移律は成り立っているので, 関係 $\geq_S$ は原始順序関係である. そこで, $\mathcal{O}_S$ の要素のうち以下の条件を満たす状態成分関数 O を適切さが "極大な" ものと考え, 分割の基準となる状態成分関数の候補とする.

$$\forall O' \, (O \geq_S O' \vee O' \not\geq_S O)$$

最後に, 分割の基準となる状態成分関数の候補を求める例として, 以下のような形式 (1) の公理を含む仕様について考える. ただし, この仕様における外部操作関数は T_1 と T_2 のみと仮定する.

$$\begin{aligned} T_1(s) &= \text{if}\,(O_1)T'_{11}(s); \text{else if}\,(!O_2)T'_{12}(s); \\ T_2(s) &= \text{if}\,(O_3)T'_{21}(s)\text{else}\ s; \end{aligned}$$

これらの公理に対し, 全状態空間 U における関係 $\geq_U$ は以下のようになる.

$$\{(O_1, O_1), (O_1, O_2), (O_2, O_2), (O_3, O_3)\}$$

したがって, O_1 と O_3 が全状態空間の分割の基準となる状態成分関数の候補である.

3.4 公理の簡約

$SPEC = (G, AX)$ を順序機械仕様, $\mathcal{G}_{SPEC} = (V, E)$ を $SPEC$ の一つの縮退遷移グラフとする. 頂点 $S \in V$ が一つ指定されると, $BIND[S]$ に従って, $SPEC$ におけるいくつかの状態成分関数の値が定まる. そこで, これらの状態成分関数の値を用いて, AX に属する公理を簡約する方法を述べる. これを頂点 S に関する公理の簡約といい, 次の (1)–(4) からなる.

(1) もし $(O_i, c) \in BIND[S]$ ならば, AX に属する公理の右辺に現れるすべての部分式 $O_i(s)$ (s は State 型の変数) を, c に置き換える.

(2) 基本データタイプの公理などを用いて, (1) で得られた公理の集合をできる限り簡約する. 現状では, Bool 型に関する公理, および, 以下の規則を用いて簡約を行なっている.

- $f(..., nil, ...)$ を nil に書き換える (fは関数)
- if (nil) exp_t を nil に書き換える
- if (nil) exp_t else exp_e を nil に書き換える
- if (p) nil を nil に書き換える
- if (p) exp_t else nil を if (p) exp_t に書き換える
- if (p) nil else exp_e を if $(!\,p)$ exp_e に書き換える

(3) T を外部操作関数, O を状態成分関数とする. 以下のようにして, 左辺が $O(x_1, ..., x_q, T(z_1, ..., z_k, s))$ の形の公理, すなわち, 頂点 S において外部操作 T を行なった後の状態成分関数 O の値を表す公理を作り, 各 T と O について得られた公理の和集合をとる:

(2) で得られた公理の集合に次のような形式 (1), 形式 (2) の公理が含まれるとする.

$$\begin{aligned} T(z_1, ..., z_k, s) \;=\; & \text{if}\,(p_1)\; T_1'(..., s) \\ & \text{else if}\,(p_2)\; T_2'(..., s) \\ & \quad\vdots \\ & (T_1', T_2', ...\text{は状態遷移関数}) \end{aligned} \tag{3.1}$$

$$O(x_1, ..., x_q, T_j'(y_1, ..., y_r, s)) = exp_j \;(j \geq 1) \tag{3.2}$$

さらに, 状態成分関数の State 型の引数が必須[8] であることから導かれる性質を表す次の公理を仮定する.

$$
\begin{aligned}
&O(x_1, ..., x_q,\ \text{if}\,(p)\ s_t\ \text{else}\ s_e) \\
&\quad =\ \text{if}\,(p)\ O(x_1, ..., x_q, s_t)\ \text{else}\ O(x_1, ..., x_q, s_e) \\
&(p \text{ は Bool 型の変数}, s_t, s_e \text{は State 型の変数})
\end{aligned}
\tag{3.3}
$$

そして, 以下の式

$$
O(x_1, ..., x_q, T(z_1, ..., z_k, s)) \tag{3.4}
$$

に対して, 公理 (3.1), (3.3), (3.2) をこの順に適用すると, 次の形の式が得られる.

$$
\begin{aligned}
&\text{if}\,(p)\ exp'_1(..., s) \\
&\text{else if}\,(p_2)\ exp'_2(..., s) \\
&\qquad\vdots \\
&(\ T'_j \text{に対して}, (3.2) \text{ の形式の公理が存在しない} \\
&\text{場合は } exp'_j = O(x_1, ..., x_q, T'_j(..., s))\ \ )
\end{aligned}
\tag{3.5}
$$

この時, (3.4) を左辺, (3.5) を右辺とする公理を作る.

(4) (3) で得られた公理集合に対して, (2) と同一の操作を行なう.

例えば, 図 3 の縮退遷移グラフにおける頂点 S_1 について公理の簡約を行なうと, 図 4 のようになる.

本システムでは, 簡約された公理を用いて, 以下のようにして $LABEL$ を求めている. まず, $LABEL[S, S']$ に含まれるすべての組 $(T(z_1, ..., z_k, s)$, $COND\langle z_1, ..., z_k, s\rangle)$ を求めるために, すべての $T(z_1, ..., z_k, s)$ に対して次のような方法で $COND\langle z_1, ..., z_k, s\rangle$ を定める. まず, 頂点 S に対し, 公理に (1) および (2) を適用して得られた公理を $AX'[S]$ とする. また, 頂点 S' に対し, $BIND[S'] = \{(O_1, c_1), ..., (O_m, c_m)\}$ とする. そして, O_i, T_j に対して $COND_{ij}\langle z_1, ..., z_k, s\rangle$ を定義する. $AX'[S]$ に次のような形式 (2) の公理が含まれるとする.

$$
\begin{aligned}
O_i(T'_j(z_1, ..., z_k, s)) \ =\ &\text{if}\,(p_{ij1})\ exp_{ij1} \\
&\text{else if}\,(p_{ij2})\ exp_{ij2} \\
&\qquad\vdots \\
&\text{else}\ exp_{ijh}
\end{aligned}
$$

この時, $COND_{ij}\langle z_1, ..., z_k, s\rangle$ は以下のように定義される.

$$\begin{aligned} COND_{ij}\langle z_1, ..., z_k, s\rangle \quad = \quad & p_{ij1} \,\&\&\, exp_{ij1} == c_i \\ || \quad & !p_{ij1} \,\&\&\, p_{ij2} \,\&\&\, exp_{ij2} == c_i \\ & \vdots \\ || \quad & !p_{ij1} \,\&\&\, \cdots \,\&\&\, !p_{ijh} \,\&\&\, exp_{ijh} == c_i \end{aligned}$$

左辺に O_i と T_j が現れるような公理を $AX'[S]$ が含まない時は, $COND_{ij}\langle z_1, ..., z_k, s\rangle$ は $c_i == nil$ と定義される.

次に, $COND\langle z_1, ..., z_k, s\rangle$ を定める. $AX'[S]$ に次のような形式 (1) の公理が含まれるとする.

$$\begin{aligned} T(z_1, ..., z_k, s) \quad = \quad & \text{if } (p_1)\ T'_1(..., s) \\ & \text{else if } (p_2)\ T'_2(..., s) \\ & \vdots \\ & \text{else if } (p_n)\ T'_n(..., s) \end{aligned}$$

$(T'_1, T'_2, ...T'_n$ は状態遷移関数)

$COND_j\langle z_1, ..., z_k, s\rangle = COND_{1j}\langle z_1, ..., z_k, s\rangle \,\&\&\, \cdots \,\&\&\, COND_{mj}\langle z_1, ..., z_k, s\rangle$ として, 以下の式を頂点 S について簡約したものが, $COND\langle z_1, ..., z_k, s\rangle$ である.

$$\begin{aligned} & (p_1 \,\&\&\, COND_1\langle z_1, ..., z_k, s\rangle) \\ || \quad & (!p_1 \,\&\&\, p_2 \,\&\&\, COND_2\langle z_1, ..., z_k, s\rangle) \\ & \vdots \\ || \quad & (!p_1 \,\&\&\, \cdots \,\&\&\, !p_{n-1} \,\&\&\, p_n \,\&\&\, COND_n\langle z_1, ..., z_k, s\rangle) \\ || \quad & (!p_1 \,\&\&\, \cdots \,\&\&\, !p_n \,\&\&\, c_1 == nil \,\&\&\, \cdots \,\&\&\, c_n == nil) \end{aligned}$$

```
vm(ssynM_req(pa_SSYNMreq, s)) =
    if(phase(s) == DATA_TRANS &&
       vrsp(s) == NO && p13(s))
            vm(s)+1;

va(ssynM_req(pa_SSYNMreq, s)) =
    if(phase(s) == DATA_TRANS &&
       vrsp(s) == NO && p13(s))
            if(vsc(s)) vm(s);
                else va(s);

ssynM(ssynM_req(pa_SSYNMreq, s)) =
    if(phase(s) == DATA_TRANS &&
       vrsp(s) == NO && p13(s))
            TRUE;

awaitSSYNMrsp(ssynM_req(pa_SSYNMreq, s)) =
    if(phase(s) == DATA_TRANS &&
       vrsp(s) == NO && p13(s))
            FALSE;

vsc(ssynM_req(pa_SSYNMreq, s)) =
    if(phase(s) == DATA_TRANS &&
       vrsp(s) == NO && p13(s))
            FALSE;

phase(ssynM_req(pa_SSYNMreq, s)) =
    if(phase(s) == DATA_TRANS &&
       vrsp(s) == NO && p13(s))
            phase(s);

vrsp(ssynM_req(pa_SSYNMreq, s)) =
    if(phase(s) == DATA_TRANS &&
       vrsp(s) == NO && p13(s))
            vrsp(s);
```

図 4 公理の簡約の例

4. 公理の説明文の生成

縮退遷移グラフの頂点 S に関する公理の簡約で得られた公理の集合を $AX[S]$ と書く. 公理の説明文は以下のように提示される.

(1) 利用者が頂点 S を指定した場合: $AX[S]$ に属するすべての公理の説明文を提示する.

(2) ユーザが辺 $S \overset{LABEL[S,S']}{\longrightarrow} S'$ を指定した場合:
$LABEL[S,S'] = \{(T_1(...), COND_1\langle...\rangle), ..., (T_n(...), COND_n\langle...\rangle)\}$ とすると, $AX[S]$ に属する公理のうち, 外部操作関数 $T_i(...)(1 \leq i \leq n)$ が左辺に現れるすべての公理の説明文を提示する.

以下に, 公理の説明文を生成する方法を示す.

4.1 基本方針

公理の説明文は以下の手順で生成する.

1. 公理 ax を構文解析することにより, ax 自身か ax に現れる各部分式 e について, 以下の情報を得る.
   (a) e の種類 (変数, 終端記号, 関数, if 式, または公理).
   (b) e が公理の左辺と右辺のどちらに現れているか.
   (c) e が if 式の条件部に現れているか.
   (d) e のデータタイプ.
   (e) e が関数ならば, e の関数の種類 (外部操作関数, 状態成分関数, 補助関数) と各引数のデータタイプ.

2. 公理 ax から日本語文 $t[ax]$ を生成する. e を ax 自身 あるいは ax に現れる部分式とし, $e_1, ..., e_m$ を e に現れる真部分式とした時, $t[e]$ の定義は以下の形式である. 厳密には, $t[e]$ は, e に関する上述の情報 (a)-(e) に依存する.

$$t[e] = 「\alpha_1 t[e_1] \cdots \alpha_m t[e_m] \alpha_{m+1}」$$

ここで, $\alpha_k(1 \leq k \leq m+1)$ は, e に対して (1) で得られた情報に依存して定まる文字系列である. 例えば, 本生成法では, 公理 $exp = exp'$ に対して,

$$t[exp = exp'] = 「t[exp] \text{ は } t[exp'] \text{ となる.}」$$

と定めている.

本生成法では状態遷移モデルに基づく表現を用いた日本語文を生成する. 具体的には, 公理 $O(...T(...)) = exp'$ から「外部操作 T を行なうと, 状態成分関数 O の値は, 公理右辺の値となる」という日本語文を生成する. ただし, 外部操作 T を行なうは $T(...)$ から生成される日本語句 $t[T(...)]$, 状態成分関数 O の値は日本語句 $t[O(...T(...))]$ のうち, 関数 O に対応する部分, 公理右辺の値は, $t[exp']$ を表す. また, 公理の右辺に現れる状態成分関数から「遷移前の状態成分関数の値」という日本語句を生成する. 例えば,

```
vm(ssynM_req(pa_SSYNMreq, s)) = vm(s)+1;
```

に対して以下の日本語句を生成する.

```
pa_SSYNMreq について ssynM_req を行なうと,
vm は, 遷移前の vm + 1 となる.
```

なお, 公理左辺が $O(x_1, ..., x_n, T(..., s))$ であり, $O(x_1, ..., x_n, s)$ が公理の右辺全体, if 式の then 部全体, あるいは if 式の else 部全体に現れる場合, $O(x_1, ..., x_n, s)$ から「変化しない」という日本語句を生成する. 例えば,

```
vrsp(ssynM_req(pa_SSYNMreq, s)) = vrsp(s);
```

に対して, 以下の日本語句を生成する.

```
pa_SSYNMreq について ssynM_req を行なうと,
vrsp は変化しない.
```

さらに, 生成される日本語文を読みやすくし, 仕様記述者の意図を日本語文に反映させるために, 生成部の利用者は関数, 変数, あるいは定数 e に対し, e から生成される日本語句 $t[e]$ を新しく定義することができる. この新しい $t[e]$ の定義を別名指定と呼ぶ. 生成部の $t[e]$ の定義は, 別名指定に置き換えられる. 例えば, 生成部の定義では, 任意の補助関数 A について, $t[A(w_1, ..., w_l)] =$「$A(t[w_1], ..., t[w_l])$」であるが, 利用者は x と y の最小値を返す補助関数 $min(x, y)$ について,

$$t[min(x, y)] =「t[x] と t[y] の最小値」$$

と定義し直すことができる. また, 表 2 のように別名指定が行なわれた場合,

```
vm(ssynM_req(pa_SSYNMreq, s)) = vm(s)+1;
```

に対して以下の日本語文が生成される.

```
大同期点発行要求を受けると,
次通し番号は, 遷移前の次通し番号 + 1 となる.
```

表 2 セションプロトコル仕様の関数に対する別名指定の例

e	$t[e]$
ssynM_req(PA, S)	大同期点発行要求を受ける
ssynM(S)	大同期点発行応答の受信を待っている
phase(S)	コネクションの状態
vrsp(S)	再同期の型
vm(S)	次通し番号

4.2 生成法の改良

生成される日本語文をさらに読みやすくするため, 生成法を改良した点について述べる.

(1) 公理の右辺から日本語句を生成する際, if式のネストや条件分岐の "複雑さ" に従って, 平文を生成するかインデントをつけた日本語句を生成するかを自動的に選択する. 例えば, `if (x >= y) x else y` から生成される日本語句は, 以下のように平文にする.

`x` が `y` 以上であるならば `x`, その他の場合 `y`

また, `if (x > 0) if (x >= y) x else y` から生成される日本語句は, 以下のようにインデントをつける.

```
x が 0 より大きいならば
    x が y 以上であるならば
        x
    その他の場合
        y
```

現システムでは, if式 *iexp* が以下のいずれかの条件を満たす時, *iexp* から生成する日本語句を平文で生成し, それ以外の場合, インデントをつける. ただし, 以下において exp_i は if 式を含まない式とする.

- *iexp* が $\mathrm{if}\,(p_1)\ exp_1$ の形式であり, 他の if 式の then 部 に現れる場合.

- *iexp* を部分式として含む公理右辺が以下の形式である場合.
  - if (p_1) exp_1
  - if (p_1) exp_1else exp_2
  - if (p_1) exp_1else if (p_2) exp_2
  - if (p_1) exp_1else if (p_2) exp_2else exp_3
  - if (p_1) exp_1else if (p_2) exp_2else if (p_3) exp_3

表 3 p の部分式 *bexp* から生成される日本語句の語尾変化

語尾	式 C			
	$\cdots$ *bexp* &&	$\cdots$ *bexp* ||	! $(\cdots$ *bexp*$)$	その他
「ある」	あり	あるか		ある
「ない」	なく	ないか	ないことが	ない
「い」	く	いか	く	い
「いる」	いて	いるか	い	いる

(2) *bexp* を, if 式 if (p) exp_1 else exp_2 における条件式 p の部分式とする. *bexp* から生成される日本語句の語尾を, 自然な日本語表現になるように変化させる. 変化させる語尾としては, 肯定表現「ある」, 否定表現「ない」, 形容詞語尾「い」, 動作進行表現「いる」の 4 種を考慮しており, 条件式 p での, 部分式 *bexp* の出現場所に依存して, 表 3のように語尾変化させる. 例えば, if $(a == b$ && $c == d)\cdots$ から生成される日本語句は, 「a と b が等しく, かつ c と d が等しいならば…」 となる.

次に, 公理の説明文の生成例を示す. 3 節の 図 の公理を入力として, 生成される公理の説明文は以下のようになる.

pa_SSYNMreq について ssynM_req を行なうと,

vm は,
遷移前の (phase) と DATA_TRANS が等しいならば
　遷移前の (vrsp) と NO が等しく, かつ p13(SPM) が TRUE で
　あるならば遷移前の (vm) + 1
となる.

va は，
遷移前の (phase) と DATA_TRANS が等しいならば
　遷移前の (vrsp) と NO が等しく，かつ p13(SPM) が TRUE で
　あるならば
　　(vsc) が TRUE であるならば
　　遷移前の (vm)
　　その他の場合は
　　変化なし
となる．

ssynM は，
遷移前の (phase) と DATA_TRANS が等しいならば
　遷移前の (vrsp) と NO が等しく，かつ p13(SPM) が TRUE で
　あるならば TRUE
となる．

awaitSSYNMrsp は，
遷移前の (phase) と DATA_TRANS が等しいならば
　遷移前の (vrsp) と NO が等しく，かつ p13(SPM) が TRUE で
　あるならば FALSE
となる．

phase は，
遷移前の (phase) と DATA_TRANS が等しいならば
　遷移前の (vrsp) と NO が等しく，かつ p13(SPM) が TRUE で
　あるならば変化なし
となる．

vrsp は，
遷移前の (phase) と DATA_TRANS が等しいならば
　遷移前の (vrsp) と NO が等しく，かつ p13(SPM) が TRUE で
　あるならば変化なし
となる．

さらに, 同じ入力に対し, 表 2 の別名を指定して公理の説明文の生成を行なうと, 以下のようになる.

大同期点発行が行なわれると,

次通し番号は,
遷移前のコネクションの状態とセションコネクションが
確立している状態が等しいならば
　遷移前の再同期の型と実行中の再同期なしが等しく,
　かつ p13(SPM) が TRUE であるならば 遷移前の次通し番号 + 1
となる.

同期点を期待する最小の通し番号は,
遷移前のコネクションの状態とセションコネクションが
確立している状態が等しいならば
　遷移前の再同期の型と実行中の再同期なしが等しく,
　かつ p13(SPM) が TRUE であるならば
　　小同期点応答を発行する権利を持っているならば
　　　遷移前の次通し番号
　　その他の場合は
　　　変化なし
となる.

大同期点発行応答の受信を待っていることを表すフラグは,
遷移前のコネクションの状態とセションコネクションが
確立している状態が等しいならば
　遷移前の再同期の型と実行中の再同期なしが等しく,
　かつ p13(SPM) が TRUE であるならば TRUE
となる.

大同期点発行への応答を待っていることを表すフラグは,
遷移前のコネクションの状態とセションコネクションが
確立している状態が等しいならば
　遷移前の再同期の型と実行中の再同期なしが等しく,
　かつ p13(SPM) が TRUE であるならば FALSE

となる.

コネクションの状態は,
遷移前のコネクションの状態とセションコネクションが
確立している状態が等しいならば
　　遷移前の再同期の型と実行中の再同期なしが等しく,
　　かつ p13(SPM) が TRUE であるならば 変化なし
となる.

再同期の型は,
遷移前のコネクションの状態とセションコネクションが
確立している状態が等しいならば
　　遷移前の再同期の型と実行中の再同期なしが等しく,
　　かつ p13(SPM) が TRUE であるならば 変化なし
となる.

5. システムの実現

本生成システムは順序機械仕様を入力とし, 公理の説明文と縮退遷移グラフからなるドキュメントを出力する. 本生成システムのブロック図を 図 5 に示す.

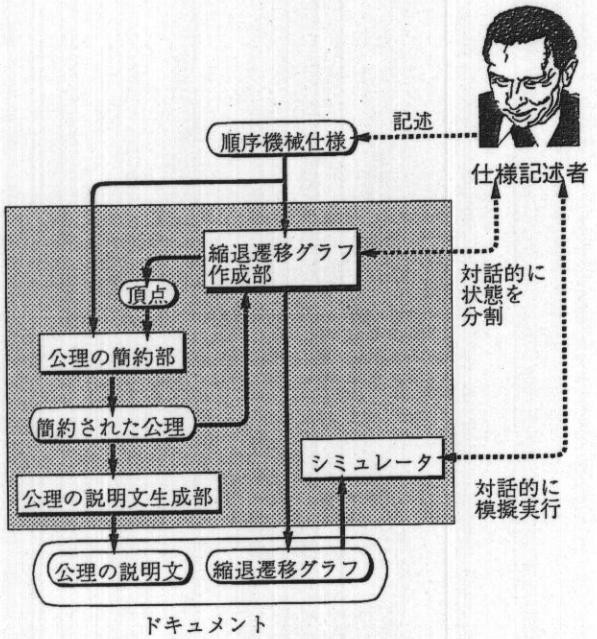

図 5 ドキュメント生成システム

縮退遷移グラフ作成部は, 順序機械仕様を入力として, 縮退遷移グラフを作成する (図 6). 規模は, Tcl/Tk で記述した部分が約 1200 行, Prolog で記述した部分が 約 50 節である. 本作成部は, 利用者と対話的に状態空間の分割を繰り返して, 縮退遷移グラフを作成する. 状態空間の分割は以下のように行なわれる.

1. 利用者が分割すべき状態空間に対応する頂点を指定する.
2. 本作成部が分割の基準となる状態成分関数の候補を提示する.
3. 利用者は状態成分関数の候補の中から一つを選択して指定する.
4. 指定された頂点を, 選択された状態成分関数の値に着目して分割する.

状態空間の分割が行なわれると, 公理の簡約部の結果を用いて, 分割の結果新しく作成された頂点を始点あるいは終点とする辺を作成する. 現在, ある頂点 S,

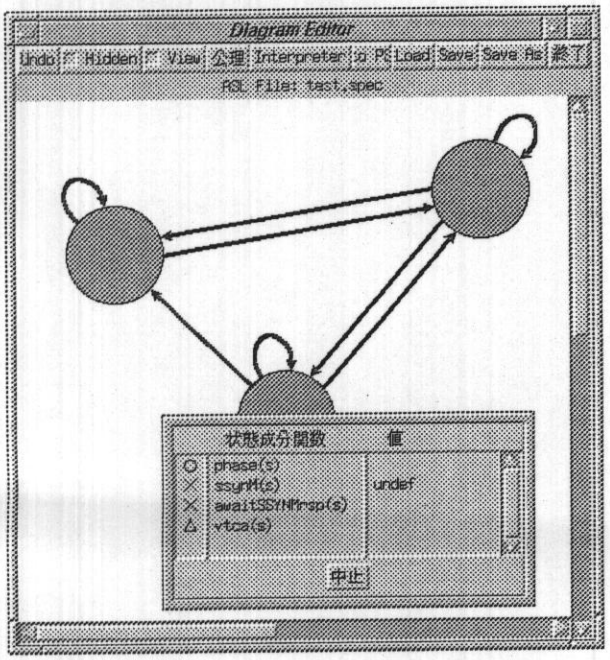

図 6 縮退遷移グラフ作成の様子

S' に対し, すべての $(T(...), COND\langle...\rangle) \in LABEL[S, S']$ について $COND\langle...\rangle$ を FALSE に書き換えることができなかった場合にのみ, 辺 $S \overset{LABEL[S,S']}{\longrightarrow} S'$ を表示するようにしている.

公理の簡約部は Prolog で実現しており, 規模は約 140 節である. 本簡約部は縮退遷移グラフの頂点 S に対する $BIND[S]$ を入力として, S について簡約された公理の集合 $AX[S]$ を出力する. 簡約部は, 各公理 $exp = exp'$ を, exp から exp' に書き換える規則とみなして, $O(x_1, ..., x_q, T(z_1, ..., z_k))$ の形の式を書き換えることにより, $AX[S]$ を作成する.

公理の説明文生成部 (図 7) は Lex, Prolog を用いて実現している. 規模は, 以下のようになっている.

字句解析部	Lex 約 200 行
構文解析部	Prolog 約 220 節
生成部 (テンプレート含む)	Prolog 約 200 節

本生成部は公理を入力として, 公理の構文木をボトムアップに書き換えることにより, 公理の説明文を出力する. 本生成部は別名指定を行なうことにより, 出力

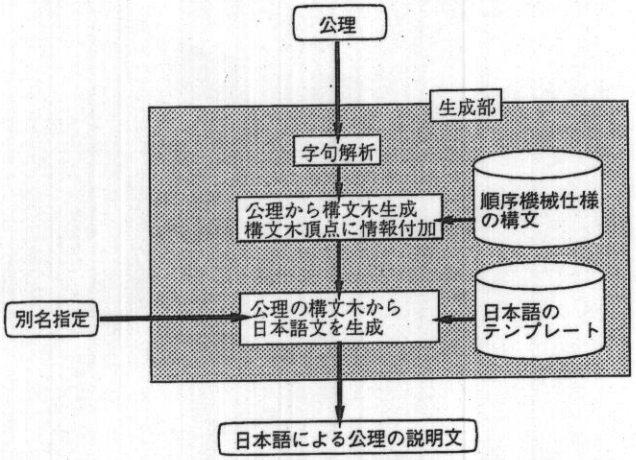

図 7 公理の説明文生成部

される公理の説明文をより理解しやすい文にすることができる. 本システムでは, 縮退遷移グラフの頂点および辺に対し, 日本語による公理の説明文を生成して関連づける. 公理の説明文の提示は, 次のように行なわれる

1. 利用者は縮退遷移グラフの辺あるいは頂点を指示する.
2. 辺が指定された場合はその辺に対応する外部操作関数のリスト, 頂点が指定された場合はすべての外部操作関数のリストが提示される.
3. 利用者は提示された外部操作関数の中から一つを選択する.
4. 簡約された公理のうち, 選択された外部操作関数が左辺に現れる公理に対し, 説明文を提示する (図 8).

シミュレータは, Tcl/Tk と Prolog で実現されており, 利用者と対話的に仕様で記述されている順序機械を模擬実行することができる. 本システムは, シミュレータを実行して, 順序機械の動的な振舞いを縮退遷移グラフ上で示すことができる.

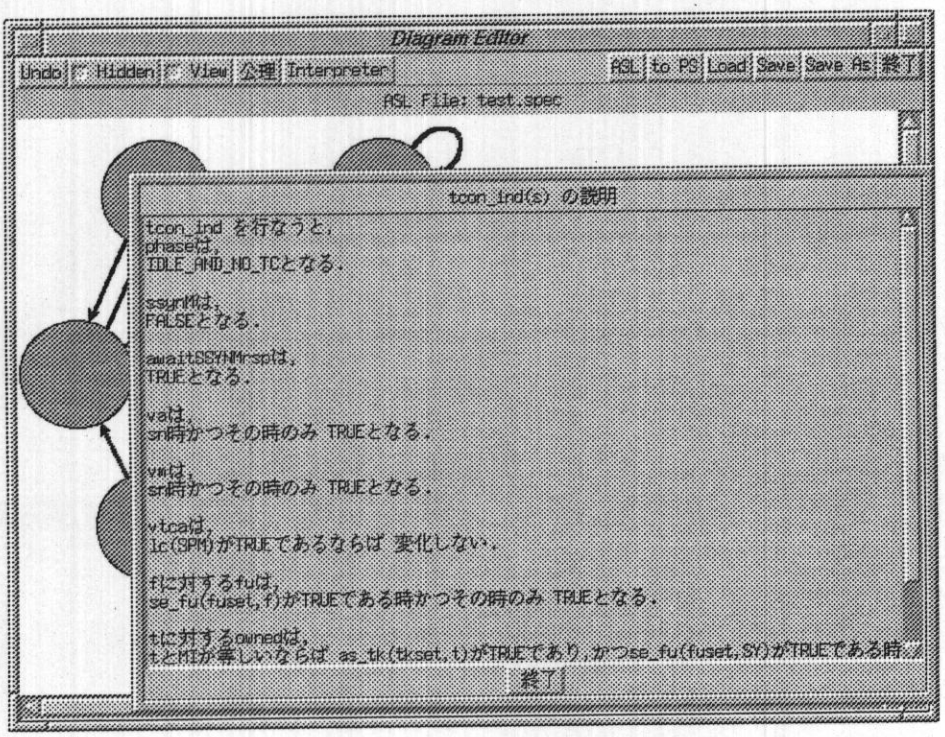

図 8 公理の説明文の提示例

6. システムの実行例

6.1 縮退遷移グラフの作成例

以下では, OSI セションプロトコルの順序機械仕様のうち, セションコネクション確立フェーズを記述している部分と大同期点機能を記述している部分を入力とした時の, 縮退遷移グラフの作成例を示す. 本システムは, まず最初に全状態空間のみからなる縮退遷移グラフを提示する. そして, 状態空間の分割を繰り返して縮退遷移グラフを作成する. 状態空間の分割を行なうには, まず分割する状態空間を指定する. すると, 本システムは, 状態成分関数のリストを示す. このリストでは, 各状態成分関数の値および, その関数の分割の基準としての適切さも表している. リストの各状態成分関数の前についているのが分割の基準としての適切さを表す印で, 分割の基準となる状態成分関数の候補を ○ で表し, どの遷移条件にも現れない状態成分関数を × で表し, その他の状態成分関数を △ で表している (図 9).

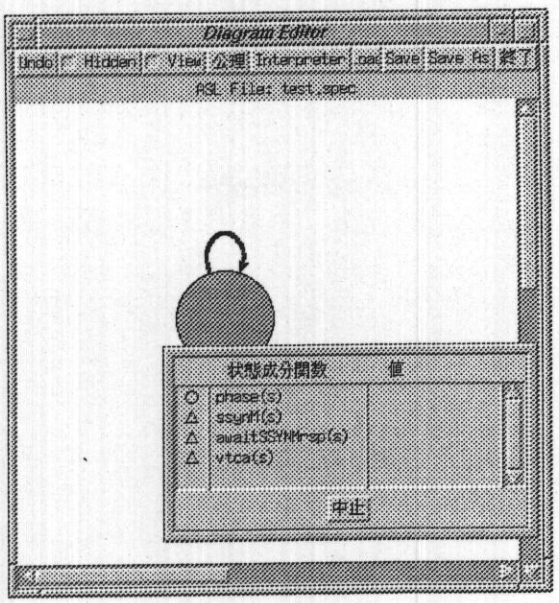

図 9 縮退遷移グラフの作成例 (1)

まず, 全状態空間を分割する場合を考える (図 9). 入力された仕様では, どの状態遷移も状態成分関数 phase の値に依存しているため, phase を分割の基準とするのが適切であると思われる. 本システムが提示する分割の基準なる状態成分関

数の候補も, phase のみである. そこで, phase を分割の基準として全状態空間を分割する.

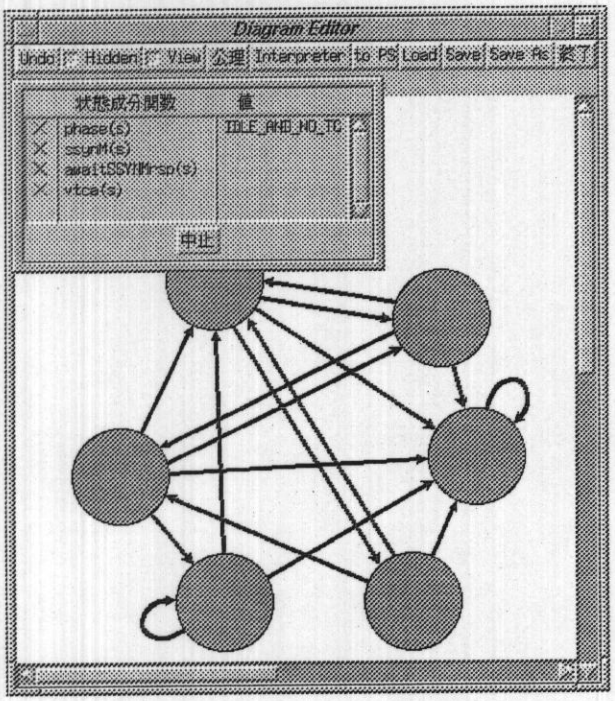

図 10 縮退遷移グラフの作成例 (2)

次に, 分割の結果作成された状態空間のうち, phase が IDLE_AND_ NO_TC である状態空間に着目する (図 10). この状態空間では, すべての状態遷移はどの状態成分関数にも依存しない. そのため, この状態空間は分割するべきではない. 本システムでも, すべての状態成分関数が×で表されているので, この状態空間は分割すべきでないことが分かる.

一方, phase が DATA_TRANS である状態空間では (図 11), awaitSSYNMrsp の値に依存する状態遷移は必ず ssynM の値に依存しているが, ssynM の値に依存する状態遷移には, awaitSSYNMrsp の値に依存しないものがある. そこで, この状態空間は ssynM を基準として分割するのが良いと思われる. 本システムが提示する分割の基準なる状態成分関数の候補も, ssynM のみである. ssynM を指定して分割を行なった結果, 新しく作成された頂点は, 図 12 のようになる.

他の状態空間についても, 本システムが提示する分割の基準となる状態成分関数の候補は, 分割の基準として適切であることが確認された.

また, 本システムは, ある頂点とその頂点の出入りする辺を隠すことができる. 例えば, 図 13 において上の 2 つのの頂点を隠すと, 図 14 のようになる. これにより, データ転送フェーズにおける大同期のように, いくつかの状態空間の間だけに起こる大局的な状態遷移に着目することができる.

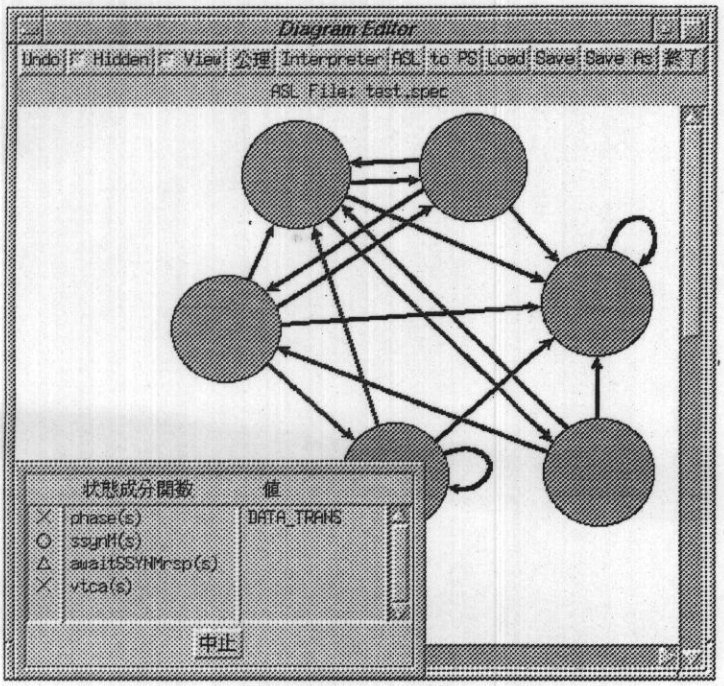

図 11 縮退遷移グラフの作成例 (3)

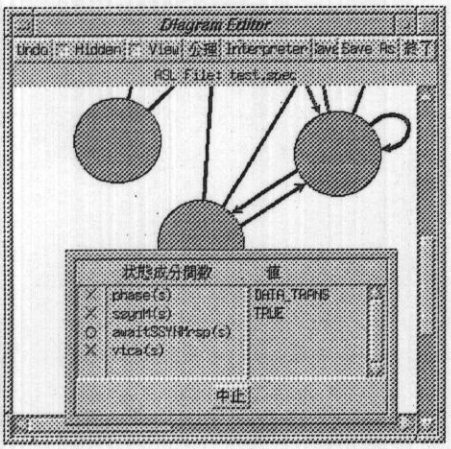

図 12 縮退遷移グラフの作成例 (4)

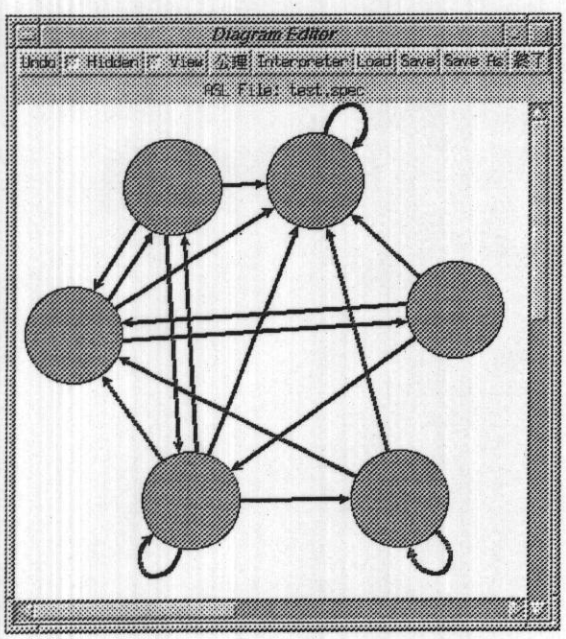

図 13 縮退遷移グラフの作成例 (5)

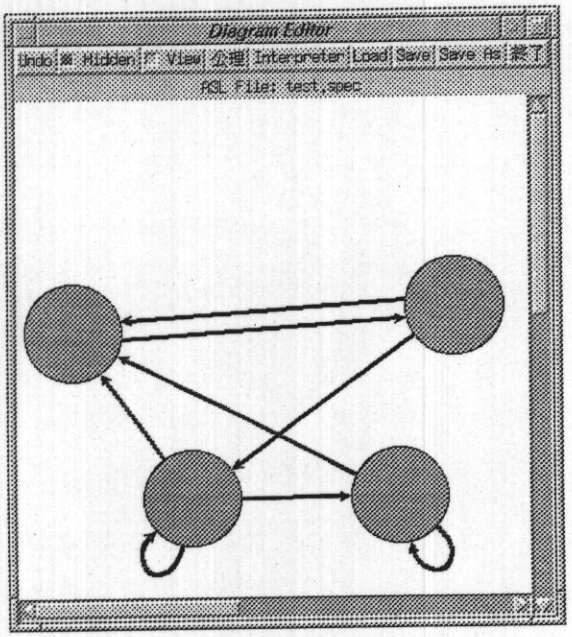

図 14 縮退遷移グラフの作成例 (6)

6.2 公理の説明文の生成実験

OSI セションプロトコル主要部 (カーネル, 半二重, 全二重, 大同期, 小同期, 再同期, 折衝解放機能単位) の順序機械仕様を入力として, 別名指定を行ない, 公理の説明文を生成する実験を行なった. ただし, 形式 (2) の公理の説明文のみを生成し, 生成の際には状態遷移関数を外部操作関数と同一視した. 仕様規模は, 形式 (2) の公理が 687, 状態成分関数 31, 状態遷移関数 54 である. DECstation 5000/25 上で本生成部を実行した結果, 実行 CPU 時間は 約 64 秒 であった. この時の, 入力文字数は 約 41000 バイトで, 出力文字数は 約 43000 バイトであった.

本研究では, その他の仕様を入力として公理の説明文を生成する実験も行なっており, その結果は以下のようになっている.

仕様	公理数	出力の規模 (文字数)
キュー	6	約 1500
自動販売機	6	約 1300
HDLC 手順	34	約 4300

さらに, 被験者に別名指定を行なってもらい, OSI セションプロトコル主要部の仕様を入力として公理の説明文を行なう実験を行なった. 被験者はセションプロトコルには精通しているが, 代数的仕様に関しては初心者である. 実験では, 19 個の定数, 85 個の関数に対して別名指定を行なった. 生成される公理の説明文が自然な日本語文になるまでに, 公理の説明文の生成は 4 回行なわれた. 各生成時に別名指定を作成するのに要した時間を以下に示す.

1 回目	60 分
2 回目	15 分
3 回目	10 分
4 回目	15 分

被験者からは, 「大規模な仕様に対して, 別名指定を改善する場合, 不自然な表現を直す手間よりも, 不自然な表現を探す手間の方が大きい」, 「読みやすさ, 理解しやすさは, 別名指定の作成者の技量に依存するところが大きい」などの指摘があり, これらの問題点への対策は今後の課題である.

7. まとめ

本研究では, 順序機械仕様を入力としてドキュメントを生成するシステムを実現した. 縮退遷移グラフの作成については, 状態空間の分割支援法を考案し, この方法に基づいて状態空間の分割を支援する機能を本システムで実現した. OSI セションプロトコルの順序機械仕様の一部を入力として縮退遷移グラフの作成を行なった結果, 状態空間の分割支援機能によって提示される状態成分関数を基準として状態空間の分割を行なえば, 大局的な状態遷移を表す縮退遷移グラフを作成できることが確認された. また, 本システムでは, 日本語による公理の説明文の生成において, 状態遷移モデルに基づく表現を用いることにより, もとの順序機械仕様の意味を説明文に正確に反映させることができた. さらに, 生成された説明文を確認した後, 利用者は関数名, タイプ名などに対して “別名” を指定できる. これにより, 理解しやすい説明文を生成することが可能となった.

謝辞

本研究にあたり, 常日頃より適切な御指導を賜わり, また貴重な御意見を頂きました伊藤実教授ならびに嵩忠雄教授に心より感謝致します. 本研究当初から本稿作成に至るまで終始, 研究の詳細に至るまで熱心に御指導をいただきました, 関浩之助教授に, 深く感謝致します. また, 本研究全過程および本稿作成全過程を通じて, 詳細に渡り御指導して頂いた石原靖哲助手に深く感謝致します. ならびに最後まで温かく見守って頂きました, 伊藤研究室の学生の方々に深く感謝します.

参考文献

[1] ISO: "Basic Connection Oriented Session Protocol Specification", ISO 8327 (1987).

[2] 嵩忠雄, 谷口健一, 杉山裕二, 関浩之: "代数的言語 ASL/* — 意味定義を中心に —", 信学論 (D), **J69-D**, 7, pp. 1066–1074 (1986–07).

[3] 工藤朋之, 石原靖哲, 関浩之: "抽象的順序機械型代数的仕様からのドキュメント生成システムの試作", 情報処理学会第 49 回全国大会, 4M-5 (1994–09).

[4] 工藤朋之, 石原靖哲, 関浩之: "抽象的順序機械型代数的仕様からのドキュメント生成システム", 信学技報, **SS94-35**, pp.1–8 (1994–11).

[5] 杉山裕二, 谷口健一, 嵩忠雄: "基底代数を前提とする代数的仕様", 信学論 (D), **J64-D**, 4, pp. 324–331 (1981–04).

[6] 高原大介: "抽象的順序機械型代数的仕様からのドキュメント生成システムにおける状態分割支援部とシミュレータの実現", 大阪大学基礎工学部情報工学科特別研究報告 (1995-02).

[7] 東野輝夫, 関浩之, 谷口健一: "代数的仕様から関数型プログラムの導出とその実行", 情報処理, **29**, 8, pp. 881–896 (1988–08).

[8] 二木厚吉, 外山芳人: " 項書き換え型計算モデルとその応用", 情報処理, **24**, 2, pp. 133–146 (1983–02).

[9] ル一光, 粟屋英司, 関浩之, 藤井護, 二宮清: "抽象的順序機械の形で記述された代数的仕様からプログラムへの変換について", 信学論 (D-I), **J73-D-I**, 2, pp. 201–213 (1990–02).