システム要求からの形式仕様の導出方法☆

金指文明† 陸 暁松† 富樫 敦††

本論文では，状態遷移システムをモデルとするソフトウェアやシステムの開発に対して，開発者が意図している機能を宣言的に表した要求記述から自動的に状態遷移システムに変換する方法，およびこの方法論によるソフトウェアの開発支援環境を提供する．まず，システムの要求記述であるシステム要求を命題論理に基づいて定義し，このシステム要求から動作モデルとなる形式仕様（状態遷移システム）に変換する方法を提案する．この変換方法から生成される形式仕様が同型の場合を除いて唯一であることを，システム要求に対する形式仕様の健全性と完全性という新しい概念を定義することによって証明する．次に，命題論理に基づくシステム要求よりも少ない記述で複雑なシステムを記述する手法として，述語論理に基づいた拡張システム要求を提案し，その記述例を紹介する．最後に，本方法論の特徴を活かしたツールを提案し，効果的にシステムを開発するための統合型開発支援環境を紹介する．

A Derivation method of Formal Specifications from System Requirements

FUMIAKI KANEZASHI,† XIAOSONG LU† and ATSUSHI TOGASHI††

In this paper, we propose a new methodology for the description of system requirements and the derivation of formal specifications (state transition systems) from system requirements. We also propose a support system based on our methodology. We will specifically deal with the issues (1) mathematical treatment of system requirements and their relationship with formal specifications represented as state transition systems, (2) a sound and complete system with respect to a system requirement, i.e. a standard system of the system requirement specified as a unique model of the system requirements, (3) extended system requirements based on the predicate logic with illustrating examples and (4) a support system for system development.

1. はじめに

本論文では，状態遷移システムをモデルとするソフトウェアやシステムの開発に対して，開発者が意図している機能を宣言的に表した仕様記述から自動的に状態遷移システムに変換する方法，およびこの方法論に基づいたソフトウェアの開発支援環境を提案する．状態遷移システムは，状態の集合と遷移の集合から構成されている．よって，状態遷移システムを用いてシステムの仕様記述を行う場合，開発者が意図した動作を考慮しながら状態や遷移を構成していくことになる．この方法では，システム全体の動作を把握している必要があるが，開発の初期段階でシステムの動作全体を把握することは困難である．これに対して，本論文の方法論は，システム全体の動作を考える必要はなく，開発者が意図している機能に着目するだけで十分である．ここでの機能は，状態遷移システムの遷移に対応する．機能を宣言的に記述し，これらの記述から自動的に動作モデルとなる状態遷移システムを合成する．システムの動作全体を把握することなく，意図している機能から状態遷移システムを構成するため，開発者が詳細な動作を把握していなくてもプロトタイプとして動作するシステムを開発することができる．

一般に，システムやソフトウェアの仕様記述法は，大きく自然言語による方法[4),5)]とFDT（Formal Description Technique）による方法に分類することができる．前者の方法は，記述言語における曖昧さがあり，厳密な仕様を記述することができない．それに対して，FDTでは，前者の記述の曖昧さが排除され，

† 静岡大学大学院理工学研究科
Graduate School of Science and Engineering, Shizuoka University
†† 静岡大学情報学部
Department of Computer Science, Shizuoka University

☆ 本論文は，論文27)を修正・拡張した論文である．

厳密な仕様を記述することができる．FDT による代表的な方法として，状態遷移システムによる動作記述法[3),15)]，ペトリネットによる検証と仕様記述法[23)]，様相論理による動作記述法[21),22)]，形式的仕様記述言語 LOTOS[16),17),19)]，CCS[7)]，Z[12),13)]などによる仕様記述方法がある．状態遷移システムは，入力によりシステム内部の状態が変化するような，ユーザとシステムが対話的に動作するシステムの仕様記述に用いられる．ペトリネットは，非同期に並列動作するプロセスが協調して動作するシステムの設計に用いられる．また，設計と同時に動作検証を行うことができる．様相論理の表現を用いた手法[24)]では，論理式を利用した数学的記述により動作仕様である状態遷移システムを合成している．形式的仕様記述言語による方法では，LOTOS による通信プロトコル合成法[18),20)]や Z とプロセス計算を組み合わせることで動作システム仕様を記述する方法[10)]が提案されている．そのほかの手法として，プロダクションシステムを用いた方法[6)]があるが，本論文のような形式的かつ論理的な議論は行っていない．

本論文は，論文 25)∼27) を基礎に，システム要求記述と形式仕様の合成方法について提案する．これらの論文では，システム要求記述から動作モデルとなる状態遷移システムを自動的に合成する手法を提案している．しかし，そこではシステム要求記述から合成される状態遷移システムが，健全かつ完全な形式仕様となることを証明してはいない．本論文では，健全かつ完全な形式仕様を合成する方法とその証明を行っている．また，命題論理に基づいた仕様記述では，大規模なシステムを記述することが困難である．本論文では，述語論理に基づいた拡張システム要求を提案する．この拡張により，述語を用いて Z で書かれた仕様記述を本論文の記述法を用いて部分的に記述可能になる．Z はオックスフォード大学で開発された汎用的な仕様記述言語であり，Zermelo-Fraenkel（ZF）流の集合論を基礎に仕様を形式的に記述するための言語である．本手法は，状態遷移の考え方を基礎に機能要求の仕様を前提条件と後提条件を記述する点においては Z と共通する．しかし，基礎とする意味の与え方に関し，本手法が Kripke 構造流の様相論理的な意味の与え方を採るのに対し，Z は ZF 流の集合論的な意味の与え方であり，意味論が異なる．本方法論において，たとえば Java などのプログラミング言語による記述への変換が可能であるが，Z においては非形式的な記述を含むため，制限を加えるなどの処置をほどこさない限り，オブジェクトコードへの自動変換器の構成は難しい．以上の他に VDM-SL[14)]などの試みが本方法と密接な関係がある．

論文の最後では，本論文で提案する方法論に基づいたシステム開発を効率良く行うための，開発支援環境を提案する．開発ツールとして

(1) システム要求の記述・編集部
(2) 形式仕様の合成・表示・編集部
(3) システムの動作シミュレータ部
(4) プロトタイプ・プログラムの自動生成部

が必須と考えられる．本開発環境では，合成された形式仕様をグラフ表示するほか，システムのプロトタイプシステムとして Java のソースコードを生成するように設計している．形式仕様からのプログラム生成は容易であるため，Java 以外の言語に適応することも可能となる．これにデバッグ・検証などの機能を付加して 4 つのツールを統合した環境を提案する．

本論文の構成は次のとおりである．まず 2 章ではシステム要求と形式仕様を定義し，3 章で健全性と完全性について述べる．4 章ではシステム要求から形式仕様への合成方法を示し，5 章では述語論理を利用した拡張システム要求についての議論を行う．6 章では本方法論に基づいた開発支援統合環境の構成と各々のツールを提案する．7 章は結論である．

2. システム要求と形式仕様

この章では，システムの要求記述を表すシステム要求 $\mathcal{R}$ とシステム要求の動作モデルとなる形式仕様 M を定義する．システム要求は機能要求と呼ばれる機能の集合から成り立っている．機能要求では，開発者が意図するシステムの機能を論理式を用いて記述する．形式仕様とはシステム要求の動作モデルであり，状態遷移システムで表される．形式仕様の各状態は解釈をともない，この解釈を用いて機能要求の論理式が評価される．つまり，機能要求をアクションの実行可能性に関する様相を備えた様相論理式と見なすと，形式仕様である状態遷移システムは Kripke 構造[1),2)]に相当する．

本論文での仕様記述法を簡単なタイマを例に紹介する．本論文では，システムの仕様記述をシステム要求と呼ぶ．

例 2.1 タイマのシステム要求 $\mathcal{R}_{timer} = \langle R, \gamma_0, \mathcal{C} \rangle$ を以下に示す．

$$
\begin{aligned}
&R = \{\\
&\quad \textbf{set\_alarm}:\\
&\quad \neg alarm\_on \overset{set\_alarm}{\Longrightarrow} alarm\_on,
\end{aligned}
$$

on_the_moment :
$alarm\_on \wedge \neg sound \stackrel{on\_the\_moment}{\Longrightarrow} sound,$
button_down :
$sound \wedge \neg button \stackrel{push\_button}{\Longrightarrow}$
$\neg sound \wedge button,$
button_up :
$\neg sound \wedge button \stackrel{push\_button}{\Longrightarrow}$
$sound \wedge \neg button$
},
$\gamma_0 = \neg alarm\_on \wedge \neg sound \wedge \neg button,$
$\mathcal{C} = \{sound \wedge button\}.$

ここで，*alarm_on*, *sound*, *button* は命題を表し，それぞれ“アラームが設定されている”，“音が鳴っている”，“ボタンが押されている”ことを示している．*set_alarm*, *push_button*, *on_the_moment* は機能に対する入力を示し，それぞれ“アラームをセットする”，“ボタンを押す”，“設定した時間になる”という機能に対する入力を表している．

$\mathcal{R}_{timer}$ は，機能要求の集合 R，初期条件 γ_0，禁止条件の集合 $\mathcal{C}$ から成り立っている．機能要求は，機能名，前提条件，入力，後提条件から構成される．これは，機能を実行する場合には前提条件を満たす必要があり，入力によって機能を実行した後は，後提条件を満たす必要があるからである．たとえば，機能要求

$$\textbf{button\_down} : sound \wedge \neg button \stackrel{push\_button}{\Longrightarrow} \neg sound \wedge button$$

は音が鳴っており，かつボタンが押されていないとき，ボタンを押すと音が止まりボタンが押された状態になることを表している．このように，個別な機能に着目することで全体としてのシステム要求を記述する．以上のように，機能要求は互いに独立であり，追加・削除を自由に行うことができる．

システム要求 $\mathcal{R}_{timer}$ から変換を行うことで自動的に形式仕様 $\mathcal{T}(\mathcal{R}_{timer})$ を導出する．本論文では，状態遷移システムのことを形式仕様と呼ぶ．導出した形式仕様は**図1** のようになる．ここで，楕円形は状態を示し，その内部には状態にともなう解釈を表した論理式が書かれている．また四角形は機能を表している．

本論文のシステム要求の特徴として，1つの機能要求が複数の遷移を表現している点があげられる．例として $\mathcal{R}_{timer}$ に対して新たに機能要求を付け加える場合を考える．新たな機能要求として，アラームの設定がされているときに，アラームの設定を解除する機能 *reset_alarm* を付け加える．*reset_alarm* は

reset_alarm :
$alarm\_on \stackrel{reset\_alarm}{\Longrightarrow} \neg alarm\_on$

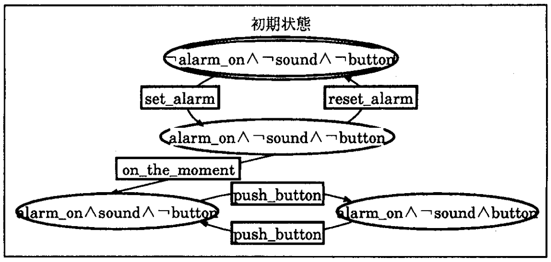


図1　$\mathcal{R}_{timer}$ の状態遷移システム
Fig. 1　State transition system for $\mathcal{R}_{timer}$.

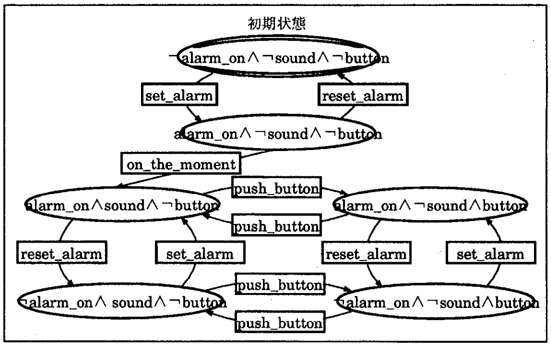


図2　新しい $\mathcal{R}_{timer}$ の状態遷移システム
Fig. 2　State transition system for new $\mathcal{R}_{timer}$.

として定義する．この機能要求の付加により導出された形式仕様は**図2** である．ここで，機能 *reset_alarm* を付加したことにより，形式仕様において2つの状態と6つの遷移が新たに付加されたことが分かる．

以下では $\mathcal{P}$ をシステム要求の記述に必要な素命題の集合とする．$p \in \mathcal{P}$ あるいは素命題の否定 $\neg p$ をリテラルといい，l などの文字を用いて表す．リテラルの論理積を

$l_1 \wedge \ldots \wedge l_n$

のように表現し，記号 γ, γ_i などを用いて表す．

機能要求は，前提条件，入力，出力，後提条件（機能実行後の条件）などから成り立っている．

定義 2.1　機能要求は5項組 $\rho = \langle id, a, f_{in}, o, f_{out} \rangle$ である．ここで，

(1)　id は，機能名である；
(2)　a は，機能の入力である；
(3)　f_{in} は機能の前提条件であり，リテラルの論理積として表現される；
(4)　o は，機能の出力である；
(5)　f_{out} は後提条件であり，リテラルの論理積として表現される．　□

理論的に取り扱う場合には，機能名 id と機能出力 o は本質的でないので，簡単のため省略することがあ

る．その場合，機能要求は $\rho=\langle a, f_{in}, f_{out}\rangle$，あるいは $\rho: f_{in} \stackrel{a}{\Longrightarrow} f_{out}$ のように略記される．本論文では，主に後者の記法を用いる．前提条件は，機能を実行する前に成り立っていなければならない条件である．後提条件は，機能を実行した後に成り立っていなければならない条件である．機能要求の定義を用いて，システム要求の定義を行う．

定義 2.2 システム要求は，3 項組 $\mathcal{R}=\langle R, \gamma_0, \mathcal{C}\rangle$ である．ここで，

(1) R は，機能要求の集合である；

(2) γ_0 は初期条件であり，リテラルの論理積として表現される；

(3) $\mathcal{C}$ は禁止条件の集合であり，個々の禁止条件はリテラルの論理積として表現される． □

初期条件 γ_0 は，開発しようとするシステムの初期状態が満たすべき条件である．禁止条件は，存在しえない状態の条件を意味する．例 2.1 はシステム要求の例である．

次に，システム要求の動作モデルとなる状態遷移システムを定義する．本論文では，状態遷移システムをシステムの形式仕様と見なす．

定義 2.3 状態遷移システムは，4 項組 $M = \langle Q, \Sigma, \rightarrow, q_0\rangle$ である．ここで，

(1) Q は状態の集合である；

(2) Σ は入力の集合である；

(3) $\rightarrow$ は遷移関係であり，$\rightarrow \subset Q\times\Sigma\times Q$ で与えられる；

(4) $q_0\in Q$ は初期状態である． □

以下では，$(p,a,q)\in\rightarrow$ のことを $p\stackrel{a}{\rightarrow}q$ と書き，状態 p において a が入力されたならばシステムの状態は q に遷移することを表す．一般性を失わず，すべての状態は初期状態から到達可能であると仮定する．

本論文で扱う状態遷移システムにおいて，素命題 A の解釈が状態 $q\in Q$ で成り立つか $(q\models A)$，成り立たないか $(q\not\models A)$，あるいは未定義であると仮定する．以下では，簡単化のため $q\models A(q\not\models A)$ と書いた場合，q における A の真偽が定義されていて，かつ成り立つこと（成り立たないこと）を表すものとする．以上より，与えられた状態のもとで任意の命題論理式の真偽を与えることができる．状態遷移システム M において，状態 $q(\in Q)$ に付随した素命題 A に対する解釈 $\mathcal{I}_q$ を，以下のように定義する．

$$\mathcal{I}_q(A)=\begin{cases}\textbf{true} & \text{if } q\models A\\ \textbf{false} & \text{if } q\not\models A\\ \textbf{undefined} & \text{otherwise}\end{cases}$$

ここで，解釈 $\mathcal{I}$ は，部分関数 $\mathcal{I}:\mathcal{P}\rightarrow\{\textbf{true},\textbf{false}\}$ と定義する．以下本論文では，状態遷移システム M に対して次の条件を仮定する：p，q を M の任意の状態とすると，$p=q$ であることと $\mathcal{I}_p=\mathcal{I}_q$ であることは同値である．つまり，すべての素命題についてその解釈が同じ状態は，同一視するという仮定である．状態に付随した解釈を定義したと同様にして，リテラルの論理積 γ に対応した解釈 $\mathcal{I}_\gamma$ を定義することができる．

3. 健全性と完全性

この章では，システム要求と形式仕様間の関係を与えるため，健全性と完全性を定義する．健全性とは，システム要求 $\mathcal{R}$ を満足する状態遷移システム M が存在することを意味する．$\mathcal{R}$ に関して健全な状態遷移システム M が $\mathcal{R}$ に関して完全であるとは，M が $\mathcal{R}$ によって記述された機能の振舞いをもらすことなく表現していることを意味する．

健全性を定義するために，システム要求の機能要求と状態遷移システムの遷移間の関係を定義する．

定義 3.1 状態遷移 $t=\langle p\stackrel{a}{\rightarrow}q\rangle$ が機能要求 $\rho: f_{in}\stackrel{b}{\Longrightarrow}f_{out}$ を満たすとは以下が成り立つときであり，$t\models\rho$ と記述する．

(1) $p\models f_{in}, a=b$，かつ $q\models f_{out}$ である；

(2) f_{out} に出現しない素命題 A に対して，$\mathcal{I}_p(A)=\mathcal{I}_q(A)$ である． □

この定義は，

(1) 現在の状態が前提条件を満たし，入力による機能の実行後，遷移した状態が後提条件を満足しなければならない；

(2) f_{out} に存在しない素命題 A の真理値は，現在の状態 p と遷移後の状態 q において変化しない；

ことをそれぞれ意味している．

機能要求と遷移間の関係を用いて，健全性を定義する．

定義 3.2 状態遷移システム $M=\langle Q,\Sigma,\rightarrow,q_0\rangle$ がシステム要求 $\mathcal{R}=\langle R,\gamma_0,\mathcal{C}\rangle$ に関して健全であるとは，以下の条件を満たすときである．

(1) $\mathcal{I}_{q_0}=\mathcal{I}_{\gamma_0}$；

(2) 任意の遷移 $t\in\rightarrow$ に対して，$t\models\rho$ となるような機能要求 $\rho\in R$ が存在する；

(3) 任意の状態 $q(\in Q)$，任意の禁止条件 $c(\in\mathcal{C})$ に対して，$q\not\models c$ である． □

この定義は，

(1) 状態遷移システムの初期状態とシステム要求の初期条件の解釈は同じである；

(2) 状態遷移システムのどの遷移であっても満足される機能要求が存在する；

(3) 状態遷移システムの状態の中には，禁止条件を満たす状態は存在しない．

ことをそれぞれ意味している．

健全性を利用することで，完全性を定義する．$\mathcal{R}$ に関して M が完全であるとは，$\mathcal{R}$ に関して健全な任意の状態遷移システム M' から準同型写像 $\xi: M' \to M$ が存在し，ξ が遷移によって満足される機能要求を保存することとして定義する．つまり，必要となる状態，遷移がすべて存在することを意味する．

完全性の定義に必要な準同型写像を定義する．

定義 3.3 $M=\langle Q, \Sigma, \to, q_0\rangle$，$M'=\langle Q', \Sigma, \to', q_0'\rangle$ を共通の入力記号を持つ状態遷移システムとする．M' から M への準同形写像は，以下を満たす写像 $\xi: Q' \to Q$ である．

(1) $\xi(q_0') = q_0$；

(2) M' において $p \xrightarrow{a} q$ ならば M において $\xi(p) \xrightarrow{a} \xi(q)$ である；

(3) M' における任意の状態 p と命題 f に対して，$p \models f$ ならば $\xi(p) \models f$ である． □

M' から M への準同型写像 ξ を便宜的に $\xi: M' \to M$ と書く．M' から M への準同型写像 $\xi: Q' \to Q$ が全単射であるとき，ξ を M' から M への同型写像といい M と M' は同型であるという．準同型写像の定義を用いて，完全性を定義する．

定義 3.4 M をシステム要求 $\mathcal{R}$ に対して健全な状態遷移システムとする．M が $\mathcal{R}$ に関して完全であるとは，$\mathcal{R}$ に関して健全な任意の状態遷移システム M' に対して，準同形写像 $\xi: M' \to M$ が存在し，次の条件を満たすことである．M' の状態遷移 $p \xrightarrow{a} q$，$\mathcal{R}$ の機能要求 ρ に対して $p \xrightarrow{a} q \models \rho$ ならば $\xi(p) \xrightarrow{a} \xi(q) \models \rho$ である． □

定義 3.5 システム要求 $\mathcal{R}$ に関して健全かつ完全な状態遷移システムを $\mathcal{R}$ の標準システムと呼ぶ．

4. 形式仕様の導出

この章では，システム要求 $\mathcal{R} = \langle R, \gamma_0, \mathcal{C}\rangle$ から状態遷移システム $M = \langle \Gamma, \Sigma, \to, q_0\rangle$ への変換 $\mathcal{T}$ を定義し，変換 $\mathcal{T}$ により $\mathcal{R}$ の標準システム $\mathcal{T}(\mathcal{R})$ を導出できることを証明する．変換 $\mathcal{T}$ を定めるために，システム要求 $\mathcal{R}$ の正規形 $\hat{\mathcal{R}}$ を定義する．この $\hat{\mathcal{R}}$ を用いて状態遷移システムを導出する．また，$\mathcal{T}(\mathcal{R})$ の健全性と完全性を証明することで，状態遷移システム $\mathcal{T}(\mathcal{R})$ がシステム要求 $\mathcal{R}$ の同型の場合を除いて唯一の動作モデルであることを示す．

定義 4.1 システム要求 $\mathcal{R} = \langle R, \gamma_0, \mathcal{C}\rangle$ の R に対して可能な限り以下の変換規則を適用した後の機能要求の集合を $\hat{R}$ とする．システム要求の正規形は，$\hat{\mathcal{R}} = \langle \hat{R}, \gamma_0, \mathcal{C}\rangle$ で定義される．

変換規則：機能要求の集合 $R' \uplus \{\gamma_1 \wedge l \wedge \gamma_2 \overset{a}{\Longrightarrow} \gamma_3\}$ を $R' \uplus \{\gamma_1 \wedge l \wedge \gamma_2 \overset{a}{\Longrightarrow} \gamma_3 \wedge l\}$ に変換する．ここで，l はリテラル（$l = A$ または $l = \neg A$）を表し，リテラルを構成する原子論理式 A は γ_i $(1 \le i \le 3)$ の中に部分論理式として現れないとする．また，$X \uplus Y$ は互いに素な集合 X と Y（$X \cap Y = \emptyset$）の和集合，つまり disjoint union を表す．

定義 4.1 は，任意の機能要求 ρ に対して，前提条件に現れ後提条件には現れないリテラルがあった場合，そのリテラルを後提条件に付け加えることを示している．

命題 4.1 変換ルールに関して次の結果が成り立つ．t を状態遷移システム M 中の任意の遷移，$\rho: \gamma_1 \wedge l \wedge \gamma_2 \overset{a}{\Longrightarrow} \gamma_3$ をシステム要求 $\mathcal{R}$ 中の任意の機能要求とし，$\rho': \gamma_1 \wedge l \wedge \gamma_2 \overset{a}{\Longrightarrow} \gamma_3 \wedge l$ と置く．ここで，$l = A$ あるいは $l = \neg A$ であり，A，$\neg A$ のいずれも γ_i $(1 \le i \le 3)$ 中に現れないとする．このとき $t \models \rho$ であることと $t \models \rho'$ であることは同値である． □

証明：定義 4.1 より明らかである． □

系 4.1 $\mathcal{R}$ をシステム要求，M を状態遷移システムとする．M が $\mathcal{R}$ の標準システムであることと M が $\hat{\mathcal{R}}$ の標準システムであることは同値である． □

例 4.1 システム要求例 $\mathcal{R} = \langle R, \gamma_0, \mathcal{C}\rangle$ を以下に示す．

$$
\begin{aligned}
R = \{ & \\
& \mathbf{a}: B \wedge C \overset{a}{\Longrightarrow} \neg A \wedge \neg C, \\
& \mathbf{b}: A \overset{b}{\Longrightarrow} \neg C, \\
& \mathbf{c}: B \overset{c}{\Longrightarrow} \neg A \\
& \}, \\
\gamma_0 = {} & A \wedge B \wedge C, \\
\mathcal{C} = {} & \emptyset.
\end{aligned}
$$

システム要求 $\mathcal{R}$ の正規形 $\hat{\mathcal{R}} = \langle \hat{R}, \gamma_0, \mathcal{C}\rangle$ は，以下で与えられる．

$$
\begin{aligned}
\hat{R} = \{ & \\
& \mathbf{a}: B \wedge C \overset{a}{\Longrightarrow} \neg A \wedge B \wedge \neg C, \\
& \mathbf{b}: A \overset{b}{\Longrightarrow} A \wedge \neg C, \\
& \mathbf{c}: B \overset{c}{\Longrightarrow} \neg A \wedge B \\
& \}, \\
\gamma_0 = {} & A \wedge B \wedge C, \\
\mathcal{C} = {} & \emptyset.
\end{aligned}
$$

次に，システム要求から状態遷移システムへの変換 $\mathcal{T}$ を定義する．

定義 4.2 変換 $\mathcal{T}$ によって，システム要求 $\mathcal{R}$ から得られる状態遷移システム $\mathcal{T}(\mathcal{R}) = \langle \Gamma, \Sigma, \rightarrow, q_0 \rangle$ を以下のように定義する．

(1) $\mathcal{R} = \langle R, \gamma_0, \mathcal{C} \rangle$ の正規形を $\hat{\mathcal{R}} = \langle \hat{R}, \gamma_0, \mathcal{C} \rangle$ とする．

(2) $\Gamma' = \{\gamma \mid \gamma$ は素命題の集合 $\mathcal{P}$ 上の同じリテラルの重複を許さないリテラルの無矛盾な論理積であり，任意の $c \in \mathcal{C}$ に対して，$I_\gamma \not\models c$ である．$\}$

(3) $\Sigma = \{a \mid \rho : f_{in} \overset{a}{\Longrightarrow} f_{out} \in R\}$

(4) q_0 を初期条件 γ_0 とする．

(5) 状態 $\gamma, \gamma' \in \Gamma$，入力 $a \in \Sigma$ について $\gamma \overset{a}{\rightarrow} \gamma'$ であるとは，ある機能要求 $f_{in} \overset{a}{\Longrightarrow} f_{out} \in \hat{\mathcal{R}}$ が存在して，次の条件を満たすことでありそのときのみである．
   (a) $\mathcal{I}_\gamma \models f_{in}$;
   (b) $\mathcal{I}_{\gamma'} \models f_{out}$;
   (c) f_{out} に出現しない素命題 A について，$\mathcal{I}_\gamma \models A$ ならば $\mathcal{I}_{\gamma'} \models A$ でありかつ逆も成り立つ．

(6) $\Gamma = \{\gamma \in \Gamma' | \gamma$は$\gamma_0$から到達可能$\}$ □

変換 $\mathcal{T}$ により導出される状態遷移システム $\mathcal{T}(\mathcal{R})$ は，システム要求 $\mathcal{R}$ の標準システムであることを証明する．この証明では，$\mathcal{R}$ に関する任意の健全な状態遷移システム M から，$\mathcal{T}(\mathcal{R})$ への準同型写像 ξ が存在することが要求される．以上を証明する前に，以下の補題を証明する．

補題 4.1 M をシステム要求 $\mathcal{R} = \langle R, \gamma_0, \mathcal{C} \rangle$ に関して健全な状態遷移システムとすると，M から $\mathcal{T}(\mathcal{R})$ への準同型写像が存在する． □

証明： $M = \langle Q, \Sigma, \rightarrow, q_0 \rangle$ を $\mathcal{R}$ に関して健全な状態遷移システムとする．写像 $\xi : Q \rightarrow \Gamma$ を

$$\xi(q) = \gamma \quad \Leftrightarrow \quad I_q = I_\gamma$$

と定める．以下，ξ が M から $\mathcal{T}(\mathcal{R})$ への準同型写像であることを示す．

(1) M は $\mathcal{R}$ に関して健全であることから $I_{q_0} = I_{\gamma_0}$ である．したがって $\mathcal{T}$ の定義より，

$$\xi(q_0) = \gamma_0$$

である．

(2) $q \overset{a}{\rightarrow} q'$ を M の任意の遷移とする．M は健全であるので，ある $\rho : f_{in} \overset{a}{\Longrightarrow} f_{out} \in R$ が存在して，次が成り立つ．
   (a) $q \models f_{in}$；
   (b) $q' \models f_{out}$；
   (c) f_{out} に出現しない $A \in \mathcal{P}$ について
   $$q \models A \quad \Leftrightarrow \quad q' \models A.$$
   $\xi(q) = \gamma$，$\xi(q') = \gamma'$ とすると，$I_q = I_\gamma$，$I_{q'} = I_{\gamma'}$ である．以上より，
   (a) $\gamma \models f_{in}$;
   (b) $\gamma' \models f_{out}$;
   (c) f_{out} に出現しない $A \in \mathcal{P}$ について，
   $$\gamma \models A \quad \Leftrightarrow \quad \gamma' \models A.$$
   したがって，$\xi(q) \overset{a}{\rightarrow} \xi(q')$ が $\mathcal{T}(\mathcal{R})$ で成り立つ．

(3) $q \in Q$ について，$\xi(q) = \gamma$ とすると，任意の命題 f について，
   $$q \models f \quad \Leftrightarrow \quad \xi(q) \models f$$
   が成り立つのは，$I_q = I_\gamma$ より明らかである．

よって，写像は M から $\mathcal{T}(\mathcal{R})$ への準同型写像である． □

定理 4.1 $\mathcal{T}(\mathcal{R})$ は，$\mathcal{R}$ の標準システムである． □

証明： 健全性については，$\mathcal{T}$ の定義から明らかに成り立つ．完全性については，補題 4.1 およびその証明で与えた準同型写像 $\xi : M \rightarrow \mathcal{T}(\mathcal{R})$ が M の状態遷移 $q \overset{a}{\rightarrow} q'$，$\mathcal{R}$ の機能要求 σ に対して，ξ の構成法より $q \overset{a}{\rightarrow} q' \models \sigma$ ならば $\xi(q) \overset{a}{\rightarrow} \xi(q') \models \sigma$ が成り立つことによる． □

$\mathcal{T}(\mathcal{R})$ は $\mathcal{R}$ の標準システムであることが証明できた．次に，この標準システムが同型の場合を除いて一意であることを示す．

定理 4.2 システム要求 $\mathcal{R}$ に関する標準システム M は同型な場合を除いて唯一である． □

証明： $M = \langle Q, M, \rightarrow, q_0 \rangle$ をシステム要求 $\mathcal{R}$ に対する標準システムとする．定理 4.1 より，$\mathcal{T}(\mathcal{R})$ は $\mathcal{R}$ の標準システムであるから，定理を証明するためには，M と $\mathcal{T}(\mathcal{R})$ が同型であることを示せばよい．そのためには，補題 4.1 の証明で与えた準同型写像 $\xi : M \rightarrow \mathcal{T}(\mathcal{R})$ が同型写像であることを示せば十分である．

はじめに ξ が単射であることを示す．いま，M の状態 $p, q \in Q$ について，$\xi(p) = \xi(q)$ と仮定する．ξ の定義より，$\mathcal{I}_p = \mathcal{I}_q$ が成り立つ．また，本論文においては，状態遷移システムに関し，すべての素命題に対する解釈が同じ状態を同一視するのであったから，$p = q$ が成り立つ．したがって，ξ は単射である．

次に，ξ が全射であることを示す．M は標準システムであり，$\mathcal{T}(\mathcal{R})$ は $\mathcal{R}$ に関して健全であることより，準同型写像 $\xi' : \mathcal{T}(\mathcal{R}) \rightarrow M$ が存在し，$\mathcal{T}(\mathcal{R})$ の状態遷移 $\gamma \overset{a}{\rightarrow} \gamma'$，$\mathcal{R}$ の機能要求 ρ について，$\gamma \overset{a}{\rightarrow} \gamma' \models \rho$ ならば $\xi'(\gamma) \overset{a}{\rightarrow} \xi'(\gamma') \models \rho$ が成り立つ．

いま，$\gamma \in \Gamma$ を $\mathcal{T}(\mathcal{R})$ の任意の状態とする．γ は

初期状態から到達可能であったから，次なる状態遷移の系列が存在する．

$$\gamma_0 \overset{a_1}{\to} \gamma_1 \overset{a_2}{\to} \cdots \overset{a_n}{\to} \gamma_n (= \gamma).$$

M の状態を，$p_i = \xi'(\gamma_i)$ と定めると ξ' は準同型写像であることより

$$p_0 \overset{a_1}{\to} p_1 \overset{a_2}{\to} \cdots \overset{a_n}{\to} p_n.$$

が M において成り立つ．そこで，i $(0 \le i \le n)$ に関する帰納法により，

$\mathcal{I}_{p_i} = \mathcal{I}_{\gamma_i}$

すなわち

$\xi(p_i) = \gamma_i$

が成り立つことを示す．$i = 0$ の場合は，明らかに $\mathcal{I}_{p_0} = \mathcal{I}_{\gamma_0}$ が成り立つ．ここで，$p_0 = q_0$ であることに注意されたい．$i \ge 0$ とし，$\mathcal{I}_{p_i} = \mathcal{I}_{\gamma_i}$ が成り立つとし，$\mathcal{I}_{p_{i+1}} = \mathcal{I}_{\gamma_{i+1}}$ が成り立つことを示す．

$\mathcal{T}(\mathcal{R})$ の作り方より，状態遷移 $\gamma_i \overset{a_{i+1}}{\to} \gamma_{i+1}$ を生成する機能要求を $\rho : f_{in} \overset{a_{i+1}}{\Longrightarrow} f_{out}$ とおくと，

$$\gamma_i \overset{a_{i+1}}{\to} \gamma_{i+1} \models \rho \qquad (1)$$

が成り立つ．したがって，

$$p_i \overset{a_{i+1}}{\to} p_{i+1} \models \rho \qquad (2)$$

が満足される．A を任意の素命題とする．A が f_{out} に出現するならば，式 (1)，(2) より

$\mathcal{I}_{\gamma_{i+1}}(A) = \mathcal{I}_{p_{i+1}}(A)$

が成り立つ．A が f_{out} に出現せず，$\mathcal{I}_{\gamma_i}(= \mathcal{I}_{p_i})$ において真偽値が定義されていれば，帰納法の仮定より

$$\begin{aligned}\mathcal{I}_{\gamma_{i+1}}(A) &= \mathcal{I}_{\gamma_i}(A)\\ &= \mathcal{I}_{p_i}(A)\\ &= \mathcal{I}_{p_{i+1}}(A)\end{aligned}$$

が成り立つ．それ以外の場合は，A は $\mathcal{I}_{\gamma_{i+1}}$ および $\mathcal{I}_{p_{i+1}}$ において未定義である．以上より $\mathcal{I}_{p_{i+1}} = \mathcal{I}_{\gamma_{i+1}}$ が成り立つ．以上より，$p_n \in Q$ について，$\xi(p_n) = \gamma$ が成り立つ．よって，ξ は全射である． □

定義 4.3 変換 $\mathcal{T}$ のアルゴリズムを以下に示す．

(1) システム要求 $\mathcal{R}$ から正規形 $\hat{\mathcal{R}}$ を求める．

(2) 初期状態 q_0 を設定する．定義 4.2 より $q_0 = \gamma_0$ である．

(3) $q = q_0$ とする．

(4) $\mathcal{I}_q \models f_{in}$ となる任意の機能要求によって，q から到達する状態 q' を定め，q' を Γ，そのときに実行する入力 a を Σ にそれぞれ追加する．また，(q, a, q') を $\to$ に追加する．ここで，$q' \models c$, $c \in \mathcal{C}$ であるときは，q' を生成しない．

(5) 生成される任意の状態 q に対して，(4) の操作を行う． □

5. 述語論理を利用したシステム要求

この章では，2 章で定義したシステム要求を，制限された述語論理を用いた記述に拡張する．この拡張による利点は，少ない記述でより複雑なシステム要求を記述することができることにある．また，欠点は意図するシステムの種類により，記述法を変える必要がある点である．以上について議論を行う．

例 5.1 は述語論理に基づいた拡張システム要求の例である．

例 5.1 ステープラのシステム要求 $\mathcal{R}_{stapler} = \langle R, \gamma_0, \mathcal{C}\rangle$ は以下のように示される．ここで，述語 $needles(n)$ はステープラ内に n 本の針が入っていることを．機能要求 **add_needle** は，針が 0 本以上 K 本未満の場合に，機能 add を実行することによって針の数が n 本ではなく $n+1$ 本の状態となる機能を表している．

$$\begin{aligned}R = \{&\\ &\textbf{add\_needle}:\\ &needles(n) \land n \ge 0 \land n < K \overset{add}{\Longrightarrow}\\ &\qquad \neg needles(n) \land needles(n+1),\\ &\textbf{use\_needle}:\\ &needles(n) \land n \ge 1 \overset{use}{\Longrightarrow}\\ &\qquad \neg needles(n) \land needles(n-1),\\ \}&\\ \gamma_0 &= needles(0)\\ \mathcal{C} &= \emptyset.\end{aligned}$$

拡張システム要求には，述語論理に基づいているため変数の概念が導入されている．この拡張により，徐々に針の数が減少したり針が補給されたりといったパラメータ変更による状態変化を持つシステムの記述が容易となる．

以下，$\mathcal{P}_{pre}, \mathcal{F}$ をそれぞれシステム記述に必要な述語記号の集合，関数記号の集合とする．この章では，すべての記述は述語論理に基づくものとする．したがって，特に断わらない限り，項，リテラルなどは 1 階述語論理の項，リテラルを意味するものとする．

定義 5.1 拡張機能要求は 5 項組 $\rho = \langle id, a, f_{in}, o, f_{out}\rangle$ である．ここで，

(1) id は機能名である；

(2) a は機能の入力であり，項として与えられる；

(3) f_{in} は機能の前提条件であり，リテラルの論理積で表現される；

(4) o は機能の出力であり，項として与えられる；

(5)　f_{out} は後提条件であり，リテラルの論理積で表現される．□

命題論理の場合と同様，述語論理の場合も便宜上機能名と出力を省略することにする．拡張機能要求 $\rho: f_{in} \stackrel{a}{\Longrightarrow} f_{out}$ は自由変数を含んでもよい．自由に出現する変数を明示するために，拡張機能要求を

$$\rho : f_{in}(\tilde{x}) \stackrel{a(\tilde{x})}{\Longrightarrow} f_{out}(\tilde{x})$$

と表したりする．ここで，$\tilde{x}$ は自由変数のベクトル表現である．拡張機能要求 $\rho: f_{in}(\tilde{x}) \stackrel{a(\tilde{x})}{\Longrightarrow} f_{out}(\tilde{x})$ は，その基礎例（ground instance）全体の集合

$$\{f_{in}\theta \stackrel{a\theta}{\Longrightarrow} f_{out}\theta \mid \theta\text{は基礎代入}\}$$

と意味的に等価である．

定義 5.2　拡張システム要求は 3 項組 $\mathcal{R} = \langle R, \gamma_0, \mathcal{C}\rangle$ である．ここで

(1)　R は，拡張機能要求の集合である；

(2)　γ_0 は初期条件であり，基礎リテラルの論理積で表現される；

(3)　$\mathcal{C}$ は禁止条件の集合であり，個々の禁止条件はリテラルの論理積で表現される．□

前述の例 5.1 が，拡張システム要求の例である．拡張システム要求により仕様記述する場合に注意しなければならない点は，データベースシステムのようにデータが随時蓄積されていくような状態変化を表す場合と，パラメータの変更により状態変化を表す場合では区別して記述する必要があることである．以下では，この 2 つの記述法の違いについて述べる．

データベースシステムのように，データが蓄積されていくような状態変化を持つシステム要求の場合は，特に考慮せずに記述することができる．その理由として，定義 4.2 (5) により，標準システム中の状態遷移 $\gamma \stackrel{a}{\to} \gamma'$ について，状態 γ に現れ遷移に対応する機能要求 $f_{in} \stackrel{a}{\Longrightarrow} f_{out}$ によって影響を受けないリテラル，つまり f_{out} と矛盾しないリテラルは状態 γ' に保存されるからである．例 5.2 が，データベースシステム的なデータ蓄積を表現した拡張システム要求である．

例 5.2　データ蓄積を考慮した拡張システム要求 $\mathcal{R}_a = \langle R, \gamma_0, \mathcal{C}\rangle$ を次に示す．

$$R = \{\mathbf{func1}:\ P(n) \stackrel{a}{\Longrightarrow} P(n+1)\},$$
$$\gamma_0 = P(1),$$
$$\mathcal{C} = \emptyset.$$

標準システムの構成方法より，$\mathcal{R}_a$ から標準システムを生成する際，機能要求 **func1** を次々に実行していくことによって，次に示すように新しい状態が漸増的に作られていく．

状態 1 : $P(1)$
状態 2 : $P(1) \wedge P(2)$
⋮
状態 n : $P(1) \wedge P(2) \wedge \cdots \wedge P(n)$
⋮
状態 m : $P(1) \wedge P(2) \wedge \cdots \wedge P(n) \wedge \cdots \wedge P(m)$

ここで，たとえば

状態 n : $P(1) \wedge P(2) \wedge \cdots \wedge P(n)$

は，状態 n においては述語 $P(1), \cdots, P(n)$ が成り立つ必要があり，その他の述語については未定義であることを表す．この例では，初期条件 $P(1)$ だけを満たす初期状態からはじまり，次々に機能の実行によって新しい述語が成り立つ状態に遷移していく．つまり，状態 n で成り立つ述語は，状態 m $(m \geq n)$ でも必ず成り立つ状態遷移システムである．

次に例 5.1 のような，徐々に針の数が減少したり，針を補給するといったパラメータ変更による状態変化を行うシステムの記述について議論する．例 5.1 は，パラメータ変更による状態変化を表した拡張システム要求の例である．$\mathcal{R}_{stapler}$ の機能要求の集合 R を，例 5.2 のように

$$R = \{$$
$$\mathbf{add}: needles(n) \wedge n \geq 0 \stackrel{add}{\Longrightarrow} needles(n+1),$$
$$\mathbf{use}: needles(n) \wedge n \geq 1 \stackrel{use}{\Longrightarrow} needles(n-1)$$
$$\}$$

と記述したのでは，針の数が n 本の状態を意図したとしても，「針の数が 1 本であり，2 本であり，$\cdots$ かつ n 本である状態」という不自然な状態が生成される．そこで，例 5.1 のように，たとえば針を加える場合には，前提条件に現れる $needles(n)$ を後提条件では $\neg needles(n)$ として成り立たなくなるように明示している．このように拡張システム要求では前提条件で満たす述語が後提条件で満たすか否かを注意深く記述する必要がある．次にシステム要求における部分解釈と全解釈の違いについて議論する．部分解釈と全解釈では，生成される形式仕様が異なる．この違いを例 5.1 を用いて説明する．初期条件が部分解釈である場合，状態遷移システムは以下のように状態 1 から順番に生成される．ここで，n は $needles$ の略記である．

状態 1 : $n(1)$
状態 2 : $\neg n(1) \wedge n(2)$
⋮

状態 K : $\neg n(1) \wedge \neg n(2) \wedge \cdots \wedge \neg n(K-1) \wedge n(K)$
全解釈の場合は，以下のように生成される．

状態 1 : $n(1) \wedge \neg n(2) \wedge \cdots \wedge \neg n(K-1) \wedge \neg n(K)$
状態 2 : $\neg n(1) \wedge n(2) \wedge \cdots \wedge \neg n(K-1) \wedge \neg n(K)$
$\vdots$
状態 K : $\neg n(1) \wedge \neg n(2) \wedge \cdots \wedge \neg n(K-1) \wedge n(K)$

このとき，各状態で入力 *use* が実行された場合を考える．全解釈の場合は，状態 n から状態 $n-1$ の状態に戻るような遷移をたどることになる．しかしながら，部分解釈の場合は，存在していない未定義のリテラルを順次追加しながら状態を生成しているため，入力 *use* の実行により新たな状態を生成することになる．この例の場合では，部分解釈による状態遷移システムよりも全解釈による状態遷移システムの方が，直観的に分かりやすい状態遷移システムとなる．

部分解釈と全解釈の大きな違いとして，記述能力の差がある．たとえば，例 5.2 のようなシステム要求を考える．簡単のために命題論理に基づくシステム要求での記述例であるが，ここでの議論は拡張システム要求にも適用される．

例 5.3 揮発性のメモリがあり，そのメモリの状態は電源を入れた時点では分からないとする．このシステムのシステム要求 $\mathcal{R}_{memory}$ は次のとおりである．

$$
\begin{aligned}
R = \{ & \\
& \textbf{switch\_on}: \neg power \overset{on}{\Longrightarrow} power, \\
& \textbf{reset\_memory}: power \overset{reset}{\Longrightarrow} good\_cond, \\
& \textbf{trouble}: power \overset{trouble}{\Longrightarrow} good\_cond \\
& \}, \\
\gamma_0 = & \neg power, \\
\mathcal{C} = & \emptyset.
\end{aligned}
$$

ここで **switch_on**，**reset_memory**，**trouble** は機能名を示す．*power*，*good_cond* は素命題を示し，それぞれ“電源が入っている”，“メモリは使用可能である”ことを示す．*on*，*reset*，*trouble* は，それぞれ“電源を入れる”，“メモリをリセットする”，“問題が発生する”という機能に対する入力を表している．導出される形式仕様は図 **3** である．

例 5.2 は，初期状態においてメモリの状態（使用可能・不可能）を把握することができないという場合の記述例であり，状態における命題の解釈が与えられない例である．例 5.2 のようなシステムの場合は，メモリの状態を示す命題の解釈を事前に定めることができないため，全解釈では仕様を記述することができない．そのため，部分解釈により仕様を記述することになる．

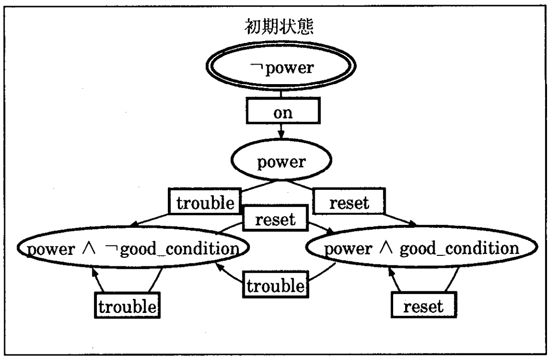


図 3 $\mathcal{R}_{memory}$ の状態遷移システム
Fig. 3 State transition system for $\mathcal{R}_{memory}$.

最後に両解釈による状態遷移システムの状態数の違いについて述べる．部分解釈による状態遷移システムの方が，命題が未定義である場合も考慮するため，全解釈の状態遷移システムよりも状態の数が多い場合が多い．しかし，全解釈の方が部分解釈よりも状態数の少ない状態遷移システムを生成できるとは必ずしもいえない．以下のシステム要求は，部分解釈の方が状態数が少なくなる例である．

$$
R = \{\mathbf{a}: \neg A \ \wedge \ \neg B \overset{a}{\Longrightarrow} B\},
$$
$$
\mathcal{C} = \emptyset
$$

ここで，部分解釈の場合の γ_0 は，

$\gamma_0 = \neg A$

全解釈の場合の γ_0 は，

$\gamma_0 = \neg A \ \wedge \ \neg B$

である．全解釈では，$\neg A \ \wedge \ \neg B$ を初期状態として，そこから機能要求 a により $\neg A \ \wedge \ B$ を生成するため，2 つの状態が生成される．一方，部分解釈の場合は，機能要求の前提条件が初期条件 γ_0 を満たさないため，初期状態だけの状態遷移システムとなる．

以上で述べた部分解釈と全解釈の比較より，開発対象となるシステムの性質によって部分解釈または全解釈を使い分ける必要がある．

6. 開発支援環境

この章では，本論文で提案したシステム要求から状態遷移システムを生成する方法論を基礎とした，システムの開発支援環境を紹介する．本論文におけるシステムの製作手順は以下のとおりである．

(1) ユーザ要求をシステム要求として記述する．
(2) 変換 $\mathcal{T}$ により形式仕様である標準システム（状態遷移システム）を自動生成する．
(3) 形式仕様を検証・診断し，形式仕様およびシス

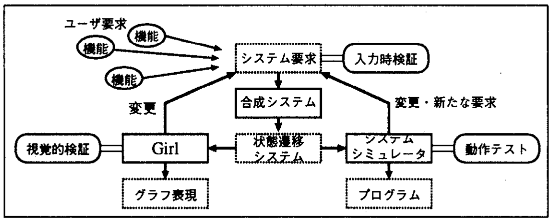


図4 統合環境システム
Fig. 4 A support system.

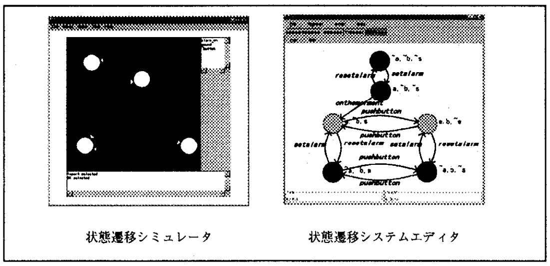


図5 システムの実行例
Fig. 5 An example of system interface.

テム要求を修正する．

(4) 最終的な形式仕様からプログラムを生成する．

この開発支援環境の特徴は，システム要求が設計者が意図するシステムの完全な仕様であるならば，それを支援システムに与えるだけで意図するシステムが生成されるという点である．開発工程が自動化されているので，システム開発期間の短縮が可能となる．しかしながら，製作者が意図している仕様を1回の記述により完全に網羅することは難しい．よって，初期のシステム要求を完全なシステム要求に修正していく方法が必要となる．

そこで本開発支援環境では，形式仕様を図的表現に変換することで動作を視覚的に検証・診断するツールや実際にシステム動作をシミュレートして動作を検証・診断するツールなどを提供する．検証により発見した不具合をシステム要求にフィードバックさせることで，完全なシステム要求に近づけていく．開発支援環境の構成図を図**4**に，実行例を図**5**に示す．

開発環境はシステム要求を入力するためのインタフェースおよび合成を行うシステム要求入力部，形式仕様を視覚的に検証・編集をする状態遷移システム表示インタフェース（Girl, Graphical Interface for Representation of Lts)，システム動作をアクションを入力することによりシミュレートして検証を行うシステムシミュレータとプログラム生成部の3つの主要なツールから成り立っている．

7. おわりに

本論文では，一般的なシステムの構築方法のような，動作に着目することで設計するのではなく，システムに必要とされる機能を宣言することで，動的モデルとなる状態遷移システムを生成する方法を提案した．この方法論には以下のような利点がある．

(1) 仕様記述は宣言的な表現であり，構造を持たないため記述しやすい．これは，複雑な構文や構造がなく，設計者が意図しているシステムに必要となる機能を宣言的に記述できるためである．

(2) 関連する部分のシステム要求の変更が容易である．この理由として，1つの機能要求は複数の状態遷移システムの遷移を表現しているため，ある機能要求の変更によって，それに関連するすべての箇所の変更がなされるからである．

(3) プロトタイプの迅速な製作が可能である．システム要求 $\mathcal{R}$ が記述できれば，変換 $\mathcal{T}$ により，状態遷移システムは自動的に生成されるため，短時間でプロトタイプが完成できる．

しかしながら以下のような欠点がある．

(1) システム要求に複雑な構文や構造がないため，記述は簡単であるが，記述の意味を理解することが難しい．

(2) 動作モデルが状態遷移システムのため，形式仕様の状態数が爆発的に増加する可能性がある．拡張システム要求を利用して仕様を記述すると，記述量自体は少なくなるが，生成される状態遷移システムの状態数は，場合によっては無限となる．支援システムでは，動作確認を図的表現するため，多数の状態を表現するのは困難である．

今後の課題として，以下で述べることについて取り組みたい．

(1) 拡張システム要求の問題点の解決を行う．制限 $\mathcal{C}$ において，様々な条件を記述できるようして，諸問題を解決する．

(2) 本論文の変換 $\mathcal{T}$ の逆変換となる状態遷移システムからシステム要求を生成する変換 $\mathcal{T}^{-1}$ を提案をする．この変換 $\mathcal{T}^{-1}$ により，状態遷移システムを操作した場合に，変更した状態遷移システムから元のシステム要求を変更できるようになる．

(3) 本論文を基礎として仕様記述言語を設計・開発をする．

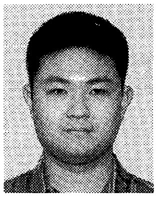
謝辞 本論文は，一部文部省科学研究費（基盤研究(C) 08680343，重点領域研究 09245214)，電気通信

普及財団，東海産業技術新興財団，栢森情報科学新興財団の援助によります．また，多くの有益なコメントと誤りの指摘をしていただいた査読者の方に感謝いたします．

参考文献

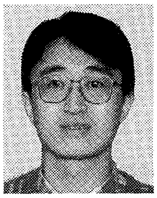
1) Abramsky, X., Gabbay, D.M. and Maibavm, T.S. (Eds.): *Handbook of Logic in Computer Science*, Vol.2 Background: Computational Structures (1992).
2) 小野寛晰：情報科学における論理，日本評論社 (1994).
3) 岡野浩三，今城広志，東野輝夫，谷口健一：拡張有限状態機械モデルを用いた分散システムの要求仕様から各ノードの動作仕様の自動導出，情報処理学会論文誌，Vol.34, No.6, pp.1290–1301 (1993).
4) Kobayashi, Y., Ohta, T. and Terashima, N.: A Requirement Description and Acquisition Method Based on Communication Service Knowledge, *7th International Conference on Systems Research Informations and Cybernetics* (Aug. 1994).
5) 原林利幸，河合敦夫，椎野 努，武内 惇：制限自然言語によるソフトウェア要求記述とその解析，ソフトウェア工学 95-3, pp.15–22 (1993).
6) Hirakawa, Y. and Takenaka, T.: Telecommunication Service Description Using State Transition Rules, *Proc. 6th Int. Work. Software Specification and Design*, pp.140–147 (1991).
7) Milner, R.: *Communication and Concurrency*, Prentice Hall (1989).
8) Stirling, C.: *Modal and temporal logics for processes*, Dept. of Computer Science, University of Edinburgh (1995).
9) Kozen, D.: Results on the propositional μ-calculus, *Theoretical Computer Science 27*, pp.333–354 (1983).
10) Fischer, C. and Smith, G.: Combining CSP and Object-Z: Finite Trace or Infinite Trace Semantics?, *FORTE/PSTV '97*, pp.503–518 (1997).
11) Evans, A.S.: Specifying and Verifying Concurrent Systems Using Z, *FME '94*, LNCS973, pp.365–380 (1994).
12) Spivey, J.: *Understanding Z*, Cambridge University Press (1988).
13) Spivey, J.: *The Z notation*, Prentice Hall (1992).
14) Jones, C.B.: *Systematic Software Development using VDM*, Second Edition, Prentice-Hall International (1990).
15) ISO: Estelle: A FormalDescription Technique based on the Extended State Transition Model, ISO 9074 (1989).
16) ISO: Information Processing Systems – Open System Interconnection – LOTOS – A Formal Descriptioin Technique based on the Temporal Ordering of Observational Behavior, ISO 8807 (1989).
17) Bolognesi, T. and Brinksma, E.: Introduction to the ISO Specification Language LOTOS, *Formal Description Technique LOTOS*, pp.23–73, Elsevier Sci. Pub. (1989).
18) Higashino, T.: Service Specification and its Protocol Specifications in LOTOS – A Survey for Synthesisand Execution, Invited Paper, 電子情報通信学会英論文誌（A），Vol.E75-A, No.3, pp.330–338 (1992).
19) 安本慶一，東野輝夫，谷口健一：LOTOS によるソフトウェアプロセスの記述とその実行，コンピュータソフトウェア，Vol.12, No.1, pp.16–30 (1995).
20) 馬淵博之，高橋 薫，白鳥則郎：LOTOS に基づいたプロトコル仕様の導出，信学技報，IN-91-110 (1991).
21) Gotzhein, R.: *Specifying Communication Services with Temporal Logic, Protocol Specification, Testing and Verification*, pp.295–309, Elsevier Science Publishers (1990).
22) Manna, Z. and Wolper, P.: Synthesis of Communicating Processes from Temporal Logic Specifications, *ACM Trans. Programming Languages and Systems*, Vol.6, No.1, pp.68–93 (1984).
23) Murata, T. and Petri Nets: Properties, Analysis and Applications, *IEEE Proc.* Vol.77, No.4, pp.541–580 (1989).
24) 木村成伴，富樫 敦，野口正一：様相論理式による基本プロセスの合成アルゴリズム，電子情報通信学会論文誌（D-I），Vol.J75-D-I, No.11, pp.1048–1061 (1992).
25) Song, K., Togashi, A. and Shiratori, N.: Verification and Refinement of System Requirements, *IEICE Trans. on Fundamentals of Elec., Comm. and Comput. Sci.*, E78-A, No.11, pp.1468–1478 (1995).
26) 宋 国煥，富樫 敦，白鳥則郎：命題論理に基づいた要求記述法と状態遷移システムによる意味記述，情報処理，Vol.37, No.4, pp.511–519 (1996).
27) Togashi, A., Kanezashi, F. and Lu, X.: A Methodology for the Description of System Requirements and the Derivation of Formal Specifications, *FORTE/PSTV '97*, pp.383–398 (1997).

(平成 10 年 5 月 8 日受付)
(平成 10 年 12 月 7 日採録)



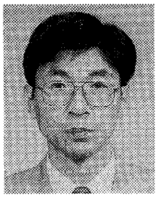
金指　文明

1974年生．1997年静岡大学工学部情報知識工学科卒業．現在，同大学大学院理工学研究科計算機工学専攻博士前期課程在学中．仕様記述方法，開発環境，プロセス計算，プログラミングに関する研究に従事．日本ソフトウェア科学会会員．



陸　　暁松

1968年生．1993年中国人民大学情報工学科卒業．同年北京光大システムインテグレーション会社入社．1996年中国オラクル勤務．現在，静岡大学大学院理工学研究科計算機工学専攻博士前期課程在学中．ソフトウェア工学，形式言語，開発環境に関する研究に従事．日本ソフトウェア科学会会員．



富樫　　敦（正会員）

1956年生．1984年東北大学大学院工学研究科博士課程修了．同年同大学通信研究所助手．1991年同助教授．1996年静岡大学情報学部助教授．1997年同教授．現在に至る．工学博士．現在，並行プロセス計算の意味論や型理論等の研究に従事．並列・分散処理基礎論，プログラム意味論や機能推論等の推論機構に興味を持つ．電子情報通信学会，日本ソフトウェア科学会，ACM各会員．