Vol.2016-SE-191 No.10
2016/3/14

プロブレムフレームに基づく状態マシン仕様の自動生成について

市川 杏子† 紫合 治†

概要：組み込みシステムの開発において，開発者はシステムの仕様を満たすだけでなく，それらの持つ外部環境についてもれなく把握しておく必要がある．本論文では，開発者への開発支援のために，プロブレムフレームの考え方を用いて，ドメインプロパティと要求を合成することで，開発するシステムの仕様を状態マシンとして自動的に生成する手法を提案する．

キーワード：プロブレムフレーム，状態マシン，仕様生成，要求分析

Synthesis of State Machine Specification based on the Problem Frames

KYOKO ICHIKAWA† OSAMU SHIGO†

Abstract: In development of embedded systems, developers have to not only meet the specifications of the system, but also comprehend all things of the environment which affect the system. This paper proposes the method of automatically synthesizing the state machine specification which consists of all possible transitions.

Keywords: Problem Frames, State machine, specification synthesis, requirement analysis

1. はじめに

近年の技術の発展に伴い，電子レンジや冷蔵庫といった電化製品，銀行の ATM 等，組み込みシステムを必要とする電子機器や装置が大幅に増加している．しかしながら，組み込みシステムの開発において，制御対象機器や装置の仕様を満たしながら，適切に動作するシステムを設計することは，多くの手間や時間が必要となる．加えて，機器や装置は想定外の振る舞いやエラーを起こす可能性があり，開発者は，システムが影響を受ける機器や装置の全ての事象について，もれなく把握しておくことが重要となる．

要求分析の手法には，UML やユースケース等，様々な手法が提案されているが，特に要求抽出型分析手法において，従来の手法では，問題そのものが十分に明確にならないと Jackson は指摘している[1]．問題そのものを明確にするためには，システムを設計するより前に，問題領域を分析すべきであるとして，Jackson はプロブレムフレーム[1]という要求分析手法を提案している．

本論文では，プロブレムフレームの考え方に基づいて，ドメインプロパティと要求を合成することにより，漏れのない状態マシンを自動的に生成する手法を考える．しかし一般に，自動生成される状態マシンには不要な遷移が含まれるため，本論文ではさらに，不要な遷移を自動的に排除する制限や条件を考える．また，提案手法に基づいた，システムの状態マシンの自動生成ツールの開発を行った．

我々の先行研究[2]として，プロブレムフレームに基づき，システムの仕様をユーザが手動で状態マシンとして設計し，作成した状態マシンシステムを分析する研究がある．しかしながら，特にシステムの規模が大きくなると，手動でシステムの状態マシンを作成することは，多くの手間や時間を費やさせ，かつ記述間違いといったヒューマンエラーを起こす可能性を増加させる．本研究では，システムの状態マシンを自動生成することにより，状態マシンの生成にかかる時間の短縮し，かつそれぞれの情報を共有することにより，記述間違いといったヒューマンエラーを防ぐことを可能とした．また，自動生成することにより，開発者が気付かなかった生成可能な遷移を発見することも期待できる．

また，別の先行研究[3][4]として，プロブレムフレームに基づき，要求を各ドメインの状態の組み合わせからなる安定状態という概念で定義することにより，ドメインプロパティと安定状態を合成することでシステムの状態マシンを自動生成する研究がある．しかしながら，生成される状態マシンには，一般に，不要な遷移が含まれるため，複雑なシステムになると，生成された遷移の一つ一つを不要な遷移か必要な遷移かを判断することは，困難な問題である．本研究では，不要な遷移の多くを自動的に排除する制限や条件を適用することにより，生成された遷移が必要な遷移であるか不要な遷移であるか判断する手間を軽減させることが可能である．

以下に，2 で関連研究について述べ，3 でプロブレムフレームについて説明し，4 で提案手法のアイディアを述べ，5 で提案手法に基づいた支援ツールについて説明し，6 で実例を用いた提案手法の評価を行い，最後に 7 でまとめと今後の課題について述べる．

† 東京電機大学大学院 情報環境学研究科
Graduate School of Information Environment, Tokyo Denki University

2. 関連研究

プロブレムフレームは，Jacksonによって提唱され，本[1]が出版されて以降，プロブレムフレームの拡張や進化についての研究がすすめられた．Cox[4]等は，これらの研究の全貌をまとめ，プロブレムフレーム研究のロードマップを発表している．

ツール化について，Ourusoff[5]は，教育の立場も踏まえて，現場の技術者がプロブレムフレームを適用する場合には，ツール支援が不可欠であると述べている．プロブレムフレームの研究者向けのツールとして，OpenPF[6]がある．これはテキスト表現からプロブレム図を生成したり，要求や仕様の規定をイベント計算(Event Calculus)で表現し，それから複数の副問題の合成の妥当性等をチェックしたりなどができるツールである．イベント計算は述語論理式をベースとした記述で，汎用性がありどのようなフレームに対しても適用できるが，反面，現場の技術者がそのまま使うのは難しいと思われる．Colombo 等[7]や Hatebur 等[8]は UML のクラス図によってプロブレムフレームのプロブレム図やコンテキスト図のメタモデルフレーム向けのツールを生成している．これは，プロブレム図に関するツールであり，個々のドメインプロパティや要求の記述に対しての支援まではない．

また，実務者向けのプロブレムフレーム用のツールとして，UMLを活用する方式が提案されている[9][10]．これらは，既存のUMLツールを使って，問題をUMLのクラス図や状態マシン図で記述し解析する方法で，既存UMLツールの機能がそのまま活用できる．しかし，UMLは解の方式を規定するのが主な目的であり，それを問題そのものの記述に適用するのは難しい．たとえば，外部のドメインと解となるマシンがともにクラスとなってしまい，その区別が明確でなくなる．また，プロブレム図における要求要素は，クラス図では表せない．クラス図に出てくる要素は，あくまでも機器やプログラム要素などの実体を伴うものであり，要求のような実体を伴わない要素は表せない．さらに，プロブレムフレームでの共有現象が，クラスのメソッドやフィールドになり，実装を意識した表現となるため，プロブレムフレームに適用することは難しい．

要求からの仕様生成の研究としては，Seater等[11]のproblem transformations，Rapanotti等[12]のproblem reduction，Li[13]のproblem progressionなどがある．これらはともに，問題世界の深いドメイン(いくつかのドメインを経由した間接的な関係しかもたないドメイン)に対する要求の表現に現れる要求現象を，対応する仕様現象に置き換えることにより，要求記述を徐々にマシンに近い(浅い)ドメインの現象で表現するように変換していく方式である．ここで，要求現象から仕様現象を取り出すのは，ドメインプロパティの記述を用いる．要求やドメインプロパティの記述として，Seater等はAlloy[14]の記述を，Rapanotti等はGentzen風の定理証明記述を使っている．このため，変換の手法は汎用的で，どのフレームにも適用可能である．しかし，問題によっては，いつも深いドメインから始めるのが良いわけではないことも分かっている．

3. プロブレムフレーム

プロブレムフレームとは，システムの開発において，システムを設計するよりも前に，システムによって制御される部品や機器・装置といった外部環境を分析し，それらがどのように振る舞うべきかを規定するという要求分析手法である．

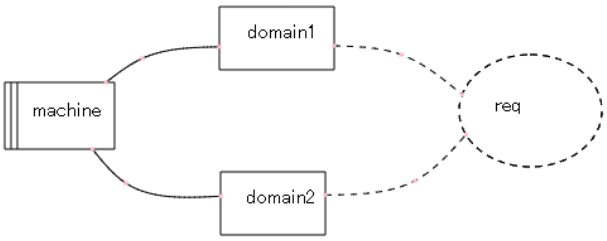


図 1　プロブレム図
Figure 1 The Problem Diagram.

プロブレムフレームにおいて基本となるものがプロブレム図（図 1）である．プロブレム図では，システムを，開発の対象である「マシン」，外部環境となる部品や装置等制御対象を「ドメイン」，ドメインの振る舞いを規定する「要求」の３つに分ける．それぞれの要素はプロブレム図上で，縦線を２本持つ四角，四角，破線の楕円で表現される．基本的なプロブレム図では，マシンと要求がそれぞれ一つずつ，そしてドメインが複数個存在する．本研究では，簡単のため，基本的なプロブレム図のみを扱うものとする．

マシンとドメインをつなぐ実線は「インタフェース」と呼ばれ，マシンとドメインの関連を表す．インタフェースは，マシンが直接感知したり発生させたりすることのできる仕様現象を規定する．マシンを取り巻くすべてのインタフェースの仕様現象に対する規定は「仕様」と呼ばれ，マシンがドメインをどのように制御するかを示す．

要求とドメインをつなぐ破線は「要求参照」と呼ばれ，要求とドメインの関連を表す．本論文では，要求とマシンが直接的な関連を持たないという制約を設ける．つまり，要求はマシンに対して直接要請することはなく，あくまでも問題に存在するドメインに対して要請するものとした．要求参照は，顧客が求める要求現象を規定する．要求を取り巻くすべての要求参照の要求現象に対する規定は「要求」と呼ばれ，ドメインに対する要請を示す．

また，ドメインごとの仕様現象と要求現象の関連に対する規定を「ドメインプロパティ」と呼ぶ．組み込みシステ

ムにおいて，ドメインプロパティの記述は，状態マシンが広く利用されていることから，本研究では，ドメインプロパティは状態マシンで表現する．問題の分析において，ドメインプロパティは考え出されるものではなく，事実として与えられるものである．ドメインプロパティの状態は，プロブレム図で要求現象として定義されたものが該当する．状態遷移を起こすイベントは，プロブレム図で仕様現象として定義されたもので，マシンの入力イベントと出力イベントに分けられる．入力イベントは，ドメインが発生させ，マシンが受け取るイベントである．出力イベントは，マシンが発生させ，ドメインが受け取るイベントである．

4. プロブレムフレームに基づいたシステムの状態マシンの自動生成手法の提案

システムの仕様を確実に満たす状態マシンを生成するためには，状態マシンにおいて，起こりうるすべての遷移を総当たりで割り出し，不要な遷移を開発者が目視でチェックし，取捨選択することで確実に実現可能である．しかしながら，システムの規模が巨大化，複雑化すると，起こりうるすべての遷移は膨大な量となる．加えて，起こりうる全ての遷移の中には，システムの仕様を満たしていない遷移も多く含まれるため，それらの中から，開発者が一つ一つチェックし，取捨選択することは現実的ではない．

この問題を解決するために，本論文では，起こりうるすべての遷移を生成するのではなく，システムの仕様を満たす遷移のみを自動生成する手法を考える．システムの仕様は，プロブレムフレームにおいて，ドメインプロパティと要求を正しく定義することで表現できることが分かっている．つまり，ドメインプロパティが許し要求を満たす，生成可能なすべての遷移を自動的に生成すれば良い．

本手法では，開発者が，システムがどのような機能を持つべきか，どのような処理を行うべきかを理解していることを前提とする．

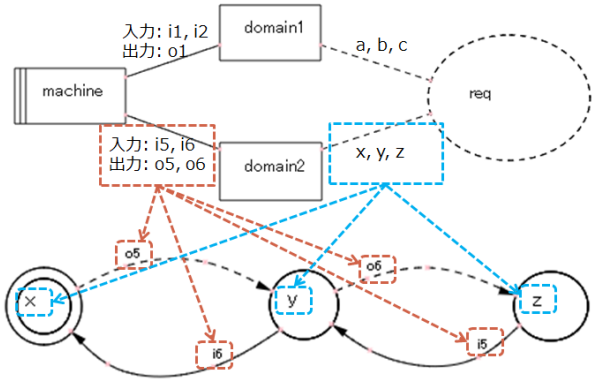


図 2　ドメインプロパティの作成
Figure 2 How to create a Domain Property

4.1 システムの状態マシンの自動生成手順

以下に，システムの状態マシンを自動生成するための手順を示す．

(1) 開発するシステムのプロブレム図を作成する．
(2) プロブレム図で定義されたそれぞれのドメインについて，ドメインプロパティを定義する．
(3) 要求を安定状態として定義する．
(4) 安定状態と入力遷移のみからなる，システムの状態マシンを生成する．
(5) (4)で得られた状態マシンに対し，制限や条件を付加することにより，不要な遷移を自動的に排除する．
(6) (5)で得られた状態マシンに出力遷移を加え，システムの状態マシンを完成させる．

4.1.1 プロブレム図の作成

開発するシステムの問題をプロブレム図で表現することで，システムの概要を把握する．このステップは，開発者が手動で行う．

4.1.2 ドメインプロパティの作成

4.1.1 で作成したプロブレム図より，ドメインプロパティを状態マシンとして定義する（図 2）．このステップは，開発者が手動で行う．

4.1.3 安定状態の作成

安定状態[15]とは，システムの仕様が許す，各ドメインの状態の組み合わせからなる，複合的な状態（図 3）である．安定状態は，入力イベントが発生しない限り，次の安定状態に遷移しないという特徴を持つ．つまり，安定状態は常に入力待ちの状態であり，これはいわば，システムの機能を意味する．開発者は前提より，システムがどのような機能を持つかを既に知っているため，その機能が，各ドメインがどの状態の時の組み合わせかを定義する．このステップは，開発者が手動で行う．

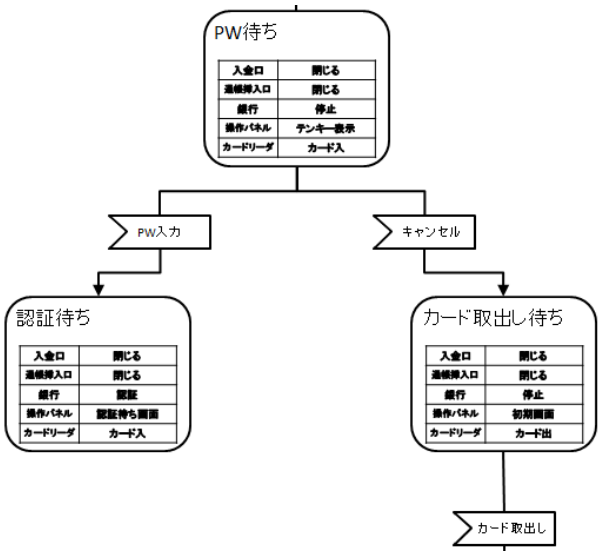
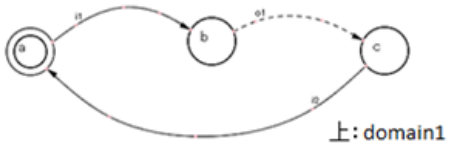
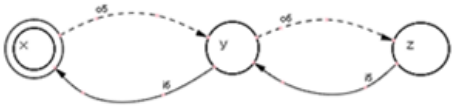


図 3　安定状態の概要
Figure 3 Concept of Stable State

4.1.4 状態マシンの自動生成

4.1.2 と 4.1.3 を合成することにより，システムの状態マシンを自動的に生成する．以下，簡単なプロブレムフレームの具体例を用いて，生成の手順を示す．





安定状態名 / ドメイン名	ss1	ss2	ss3
domain1	a	c	b
domain2	x	x	z

図 4　簡単なプロブレム図の例
Figure 4 Example: The simple Problem

(1) 安定状態 ss1 を構成する，domain1 の状態 a に対し，入力イベント i1 が発生すると，domain1 の状態は，b に遷移する．安定状態 ss1 は，domain1 の状態が b，domain2 の状態が x の状態の組み合わせである，中間状態へ遷移する（図 5）．

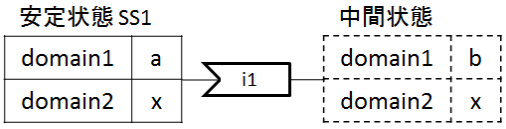


図 5　自動生成の手順 1
Figure 5 Step 1 of generating the state transition

(2) 中間状態を構成する，domain1 の状態 b から，0 回以上の出力遷移の使用で到達可能なすべての状態を SO_1 とおく．同様に，domain2 の状態 x から，0 回以上の出力遷移の使用で到達可能なすべての状態を SO_2 とおく．

$$SO_1 = [b, c], SO_2 = [x, y, z]$$

(3) SO_1 と SO_2 の積を SO_x とおく．

$$SO_x = [\{b, x\}, \{b, y\}, \{b, z\}, \{c, x\}, \{c, y\}, \{c, z\}]$$

SO_x の組み合わせの中から，同じ状態の組み合わせを持つ安定状態を探す．図 4 の例では，[c, x]の組み合わせ，[b, z]の状態の組み合わせが，それぞれ安定状態 ss2, ss3 の状態の組み合わせと一致した（図 6）．

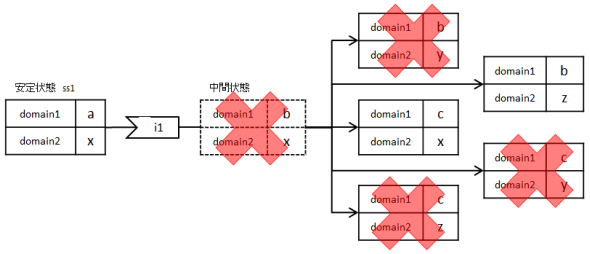


図 6　自動生成の手順 3
Figure 6 Step 3 of generating the state transition

(4) (3)より，安定状態 ss1 は，入力イベント i1 が発生すると，安定状態 ss2 と ss3 へ遷移するという状態マシンが生成される（図 7）．

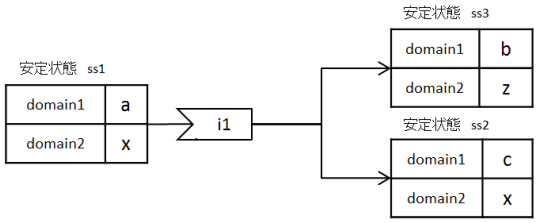


図 7　自動生成の手順 4
Figure 7 Step 4 of generating the state transition

(5) domain1 の状態 a から発生しうるすべての入力イベントと，domain2 の状態 x から発生しうるすべての入力イベントに対し，(1)から(4)の手順を繰り返す．

(6) (1)から(5)の手順を，すべての安定状態に対して実行する．

ただし，安定状態は，必ず 1 つ以上の安定状態に遷移しなければならない．遷移先のない安定状態が存在する状態マシンは誤りとし，ドメインプロパティもしくは安定状態に誤りがあると判断する．

4.1.5 不要な遷移の排除

4.1.4 で生成される状態マシンには，一般に，理論上遷移可能な遷移であっても，開発したいシステムにとっては不要な遷移が含まれる．これは，開発対象のシステムに必要とされる機能が，開発者によって一様ではないためである．不要な遷移を確実に取り除くためには，開発者が，生成された遷移を一つずつ目視でチェックし，取捨選択することで実現できるが，システムが大規模化したり、複雑化すると，目視でチェックすることは現実的ではない．不要な遷移の多くを自動的に排除する制限や条件として，以下の 5 つを提案する．

① 自動生成の際に使用する出力遷移は，idle 状態を経由しない（図 8）．

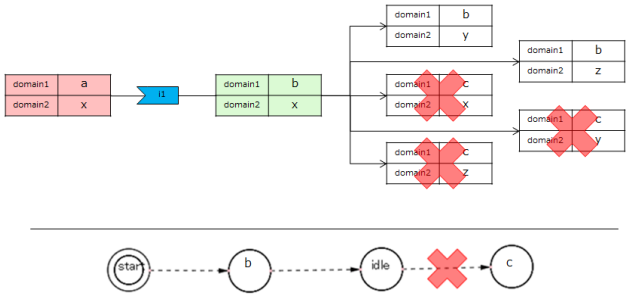


図 8　idle 状態を経由しない制限
Figure 8

② 自動生成の際に使用する出力遷移の回数は，0 回か 1 回のみとする（図 9）.

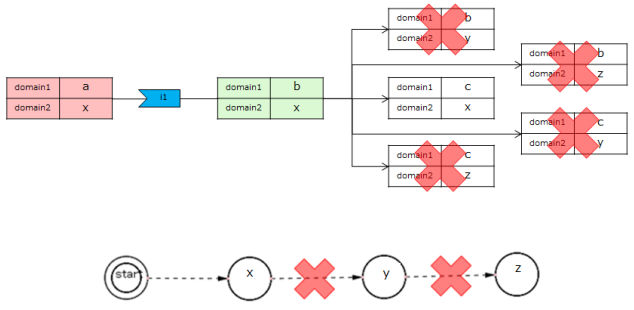


図 9　使用する出力遷移の回数の制限
Figure 9

③ 自己遷移は全て不要な遷移とみなす.

④ ドメイン間の入力イベントと状態の関係性(reset という入力イベントが発生したらカードドメインはカード排出状態になる等)を条件として規定する．ただし規定する条件は，システムが持つべき一般的な処理についてのみに限る．ドメインの関係性は，以下の 2 つの条件で規定する.

(1) ドメイン D1 で入力イベント I が発生したならば，直後の安定状態では，ドメイン D2 の状態は S でなければならない

(2) ドメイン D2 の状態が S ならば，直前の安定状態では，ドメイン D1 では入力イベント I が発生しなければならない

⑤ 安定状態の遷移先が複数個ある入力イベントのみを，開発者に提示する.

制限や条件によって排除しきれなかった不要な遷移は，自動排除した結果の状態マシンから，開発者が目視で取捨選択をするものとする. なお, 本研究では①から⑤のうち，①②④のみを採用した.

4.1.6 出力遷移を加える

4.1.5 で不要な遷移が排除された状態マシンに対し，遷移に使用した出力遷移を加えることで，システムの最終的な状態マシンを生成する．このステップは，すでに先行研究[2][3]で行われていることより，ここでは説明を省略する.

5. 提案手法に基づいた支援ツールの開発

提案手法に基づいた，支援ツールを開発した．支援ツールは，プロブレム図の作成（4.1.1），ドメインプロパティの編集（4.1.2），安定状態の作成（4.1.3），状態マシンの自動生成（4.1.4），及び条件の作成（4.1.5 の④）と，制限下（4.1.5 の①②）と条件下（4.1.5 の④）での状態マシンの自動生成を行う機能を持つ．作成，生成されたすべての情報は互いに共有しているため，状態名やイベント名の記述間違い等の，単純なミスを防ぐことが可能である．また，ツールにはセーブロード機能が搭載されており，プロジェクトを継続して編集することが可能である．さらに，プロジェクトの書き出し，読み込みが可能であるため，多数の分析者とプロジェクトを共有することが可能である.

6. 提案手法の評価と考察

5 で開発したツールを用いて，4 の提案手法を ATM と電子レンジの 2 つの実例に適用し，提案手法の評価と考察を行う.

6.1 ATM の問題

ATM は，利用者がお金の預け入れ，引き出し，残高照会を行う. 利用者はまず, カードをカード挿入口に挿入する. ATM はカードの挿入を確認すると, 銀行のネットワークにアクセスし，パスワードを要求する．利用者はテンキーを用いてパスワードを入力し, ATM は入力されたパスワードが正しいかどうか，銀行に認証確認を行う．認証が出来なければ，利用者にもう一度パスワードの入力を促し，認証が完了すれば，操作の選択を促す.

預け入れが選択されると，ATM は現金の入金口を開き，利用者はその中にお金を入れる. ATM は入金口にお金が挿入されたことを確認すると，入金口を閉じ，入金された金額を数える．入金された金額分を残高に追加すると，ATM は預け入れの処理を終え，カードを排出する.

引き出しが選択されると，利用者はテンキーで引き出したい金額を入力する. ATM は銀行のネットワークにアクセスし，入力された金額が残高を上回っていないかをチェックする．上回っていればエラー表示をし，引き出しの処理を終了し，カードを排出する．下回っていれば，入金口に入力された金額分を用意し，入金口を開ける．残高から金額分のお金を減らすと，ATM は引き出しの処理を終え，カードを排出する。

残高照会が選択されると，銀行のネットワークにアクセスし，残高を表示し，ATM は残高照会の処理を終了し，カードを排出する.

キャンセルボタンが押されると，操作を中止し，カードを排出する．ただし，キャンセルボタンは，ATM が処理中の場合には，選択することができない．また，処理中にエラーが発生した場合は，速やかにカードを排出する.

表 1　ATM の安定状態

Table 1 Stable States of an ATM

Domain / Stable State	idle	waitingForCard	waitingForPW	waitingForCertification	waitingForChoice
Card	idle	waitingForCard	cardIn	cardIn	cardIn
PaperMoney	idle	idle	idle	idle	idle
Tenkey	idle	idle	inputing	inputFin	idle
Button	waitingForStart	waitingForCertification	waitingForCertification	waitingForCertification	waitingForChoice
ResetButton	idle	waitingForReset	waitingForReset	idle	waitingForReset
Bank	idle	idle	waitingForPW	waitingForCertification	certified

Domain / Stable State	waitingForInputingNum	waitingForAmountCheck	despendingCash	saitingForCashRemoval	waitingForDeposit
Card	cardIn	cardIn	cardIn	cardIn	cardIn
PaperMoney	Idle	idle	idle	openGetCash	openDepositNULL
Tenkey	inputing	inputFin	idle	idle	idle
Button	getCash	getCash	getCash	getCash	depositMoney
ResetButton	waitingForReset	idle	idle	idle	waitingForReset
Bank	waitingForInputNum	amountCheck	despensed	fin	waitingForCountMoney

Domain / Stable State	countCash	depositingCash	waitingForBalance	processFin	watingForTakingCard
Card	cardIn	cardIn	cardIn	cardIn	waitingForTakingCard
PaperMoney	countCash	idle	idle	idle	idle
Tenkey	idle	idle	idle	idle	idle
Button	depositMoney	depositMoney	dispBalance	fin	idle
ResetButton	idle	idle	idle	waitingForReset	idle
Bank	waitingForCountMoney	Depositing	waitingForDispBalance	fin	idle

表 2

Table 2 Stable states of a microwave

Domain / Stable State	Start*	Start	Time*	Time	Cooking	CookEnd
Door	open	close	open	close	close	close
StartButton	idle	idle	idle	on	pushed	pushed
CancelButton	idle	idle	on	on	on	on
Timer	idle	idle	waiting	waiting	working	idle
Beep	idle	idle	idle	idle	idle	beep
Light	idle	idle	idle	idle	working	idle
Table	idle	idle	idle	idle	working	idle
Tube	idle	idle	idle	idle	working	idle

6.1.1 ATM の問題の設計

問題をプロブレム図におこしたものを，図 10 に示す．また，作成した安定状態を表 1 に示す．自動生成する遷移の中から，不要な遷移を排除する条件として，以下の 9 つを規定した．

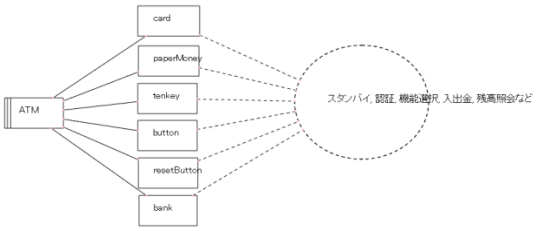


図 10　ATM のプロブレム図

Figure 10 Problem Diagram of an ATM

① button start → card waitingForCard（スタートボタンが押されたら，カードリーダはカード挿入待ちになる）

② bank certifiedOK → button waitingForChoise（PW の認証が完了したら，ボタンは操作選択の状態になる）

③ bank despensedNewBalance → paperMoney openGetCash（銀行の出金処理が完了し新たな残高が算出されたら，入金口は金を用意し入金口を開く）

④ bank depositedNewBalance → button fin（入金処理が完了し新たな残高が算出されたら，ボタンは処理の続行か終了を選択する状態になる）

⑤ resetButton reset → card waitingForTakingCard（リセットボタンが押されたら，カードリーダはカードを排出する）

⑥ resetButton reset ← card waitingForTakingCard（カードリーダがカードを排出したなら，事前にリセットボタンが押されていること）
⑦ paperMoney getCash → button fin（入金口から金がとり出されたら，ボタンは処理の続行か終了を選択する状態になる）
⑧ paperMoney sendAmount → bank depositing（入金口に入れられた金額を送信したら，銀行は入金処理を行う）
⑨ bank dispBalance → button fin（残高表示をしたらボタンは処理の続行か終了を選択する状態になる）

6.1.2 結果

生成された遷移の総数と，その内訳を表 3 に示す.「必要」は必要な遷移，「自己」は自己遷移，「可能」は不要な遷移ではあるが存在しても問題のないあり得る遷移，「不要」は不要な遷移を表している．排除対象の遷移は，「自己」「可能」「不要」な遷移である．遷移の生成は，以下の 6 つの方式のもとに行った．

① 制限なし・条件なし
② idle 状態を経由しない出力遷移の制限あり
③ 使用する出力遷移を 0 回か 1 回のみとする制限あり
④ 条件あり
⑤ ②の制限あり・条件あり
⑥ ③の制限あり・条件あり

すべての自己遷移は不要な遷移であった．また，24 個の入力イベントのうち，23 個は遷移先の安定状態が 1 つのみであった．

①より，提案手法はシステムに必要な遷移の全てを，自動的に生成した．②の idle 状態を経由しない制限は，必要な遷移を排除しないが，多くの排除も行わない．③の出力遷移の使用回数制限は，多くの排除を行うが，同時に必要な遷移も排除する．④の条件は，多くの排除を行い，かつ必要な遷移を排除しない. ⑤⑥の制限と条件の複合適用は，それぞれの制限と条件で排除した個数の和集合が，排除個数となった．

6.2 電子レンジの問題

電子レンジは，利用者が温めたいものを電子レンジに入れ，設定した時間だけ調理する．扉の開閉に関わらず，電子レンジはタイマを設定することができる．利用者がタイマを設定し，スタートボタンを押すと，電子レンジは庫内灯を点灯させ，テーブルを回転させ，調理を開始する．設定された時間が経つと，電子レンジはブザー音を鳴らし，調理を完了する．

タイマの設定時，調理時にキャンセルボタンが押されると，電子レンジは調理を中止し，タイマ等のすべてを初期化する．タイマの設定は，設定が完了するかキャンセルボタンが押されるまで，扉を開閉しても設定が継続される．

6.2.1 電子レンジの問題の設計

問題をプロブレム図に起こしたものを，図 11 に示す．

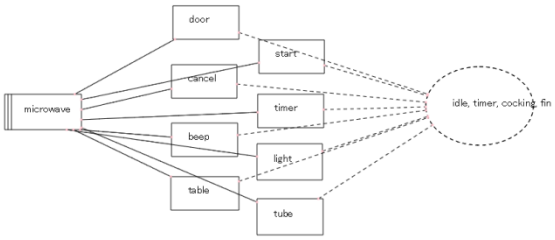


図 11 電子レンジのプロブレム図
Figure 11 Problem Diagram of a microwave

また，作成した安定状態を表 2 に示す．自動生成する遷移の中から，不要な遷移を排除する条件として，以下の 3 つを規定した．

① start push → light working（スタートボタンが押されたら，庫内灯が点灯する）
② cancel push → start idle（キャンセルボタンが押されたら，電子レンジは初期状態に戻る）
③ timer timeOut → beep beep（タイマで設定された時間が完了したら，ビープはビープを鳴らす）

6.2.2 結果

生成された遷移の総数と，その内訳を表 4 に示す．遷移の生成に使用した方式は，6.1.2 と同じである．

すべての自己遷移は不要な遷移であった．また，15 個の入力イベントのすべてに関して，遷移先の安定状態は 1 つのみであった．

①より，提案手法はシステムに必要な遷移のすべてを，自動的に生成した．②の idle 状態を経由しない制限は，必要な遷移を排除しないが，多くの排除も行わない．③の出力遷移の使用回数制限で排除された遷移は，idle 状態を経由しない制限と等しかった．④の条件は，多くの排除を行い，かつ必要な遷移を排除しない．⑤⑥の制限と条件の複合適用は，それぞれの制限と条件で排除した個数の和集合が，排除個数となった．

6.3 考察

ATM，電子レンジそれぞれの例において，システムの仕様として必要なすべての遷移を生成できた．ドメインプロパティと安定状態による要求を合成することにより，システムの仕様を満たす，漏れのない状態マシンを生成することができる．

使用する出力遷移が idle 状態を経由しないように制限すると，必要な遷移を排除せず，不要な遷移を排除することが可能だが，排除される遷移の数は多くない．使用する出力遷移の回数を制限すると，多くの不要な遷移を排除するが, 同時に必要な遷移を排除する可能性も高くなる. また，システムの規模が小さい場合は，出力遷移の回数を制限した場合と，idle 状態を経由しない制限の場合とでは，結果に差がないことから，出力遷移の制限としては，idle 状態を経由しない制限を採用するのが適切であると考える．

ドメイン間の入力イベントと状態の規定は，多くの不要な遷移を排除し，かつ必要な遷移を排除しなかったことから，不要な遷移を自動排除する要素として効果的であったといえる．ただし，規定する条件の数が多くなると，排除する不要な遷移の数は増えるが，同時に開発者への負担が増加する．少ない条件で，多くの不要な遷移を排除する条件付けの手法が望まれる．

制限と条件の複合適用で排除される遷移の数は，それぞれの制限下、条件下で排除される遷移の個数の和集合で求められる．よって，条件のみの適用と比べ，複合適用の排除個数は，必ず，条件のみの適用の結果と等しいかそれ以上となる．しかしながら，idle 状態を経由しても良い例(電話交換機等)も存在するため，条件のみの適用の方が，汎用性が高いと考えられる．

表 3　生成された遷移の個数（ATM）
Table 3 The number of generated transitions (ATM)

	①	②	③	④	⑤	⑥
全て	63	60	46	32	31	28
必要	25	25	23	25	25	23
自己	15	13	11	6	5	5
可能	17	17	8	1	1	0
不要	6	5	4	0	0	0

表 4　生成された遷移の個数（電子レンジ）
Table 4 The number of generated transitions (microwave)

	①	②	③	④	⑤	⑥
全て	29	23	23	21	21	21
必要	15	15	15	15	15	15
自己	8	3	3	3	3	3
可能	0	0	0	0	0	0
不要	6	5	5	3	3	3

7. まとめ

本研究では，プロブレムフレームに基づいたシステムの状態マシンを自動生成する手法を提案し，提案手法に基づいた支援ツールを開発した．また，開発したツールを実例に適用し，提案手法の効果の検証を行った．提案手法は，システムの仕様を満たし，かつ漏れのない状態マシンを自動生成し，提案手法が効果的であったことが確認できた．

さらに，自動生成される状態マシンには，一般に，不要な遷移が含まれることから，不要な遷移の多くを自動排除する制限と条件を提案し，同様に実例に適用し，効果を検証した．制限の適用は効果的でないが，条件の適用は効果的であることが確認できた．また，制限と条件を複合適用は，制限が適さない例も存在することから，不要な遷移の自動排除手法としては，条件のみの適用が，最も適切であることを確認した．

今後の課題として，本研究で採用しなかった，自己遷移を不要な遷移とみなす条件，複数の安定状態に遷移する安定状態のみを開発者に提示する手法の効果の検証を行いたい．また，電子レンジの例で，不要な遷移をすべて排除しきれなかったことから，より効果的な条件の調査，また別の条件や制限の調査と効果の検証を行いたい．さらに，ツール本体のユーザインタフェースの改善を予定している．

参考文献

[1] Jackson, M.: Problem Frames: Analyzing & Structuring Software Development Problems, Addison-Wesley (2001).
[2] 紫合治, 横山薫: プロブレムフレームに基づく組み込みシステムの状態遷移分析支援システム, 情報処理学会論文誌, Vol.53, No.2, pp.523-534 (2006).
[3] 和田遥, 紫合治: 組み込みシステムにおける仕様の自動生成, ソフトウェアエンジニアリングシンポジウム (2011).
[4] 市川杏子, 紫合治: プロブレムフレームに基づく状態マシン仕様の設計支援システム, ソフトウェアエンジニアリングシンポジウム (2015).
[5] Cox, K., Hall, Jon G. and Rapaotti, L.: Editorial: A roadmap of Problem Frames research, Information and Software Technology, Vol.47, Issue 14, pp.891-902 (2005).
[6] Ourusoff, N.: Towards a CASE tool for Jackson's JSP, JSD and Problem Frames, Proc. 1st International Workshop on Applications and Advances of Problem Frames, pp.69-73 (2004).
[7] Tin, T. T., Yu, Y., Haley, C. and Nuseibeh, B.: Modelbased Argument Analysis for Evolving Security Requirements, 4th International Conference on Secure Software Integration and Reliability Improvement, pp.88-97 (2010).
[8] Colombo, P., et.al: Towards a Meta-model for Problem Frames: Conceptual Issue and Tool Building Support, Fourth International Conference on Software Engineering Advance, pp.339-345 (2009).
[9] Hatebur, D., Heiselm M., and Schmidt, H.: A Formal Metamodel for Problem frames, MoDELS 2008, LNCS 5301, pp.69-82 (2008).
[10] Lavazza, L. and Bianco, V.D.: A UML-based approach for representing problem frames, Proc. 1st International Workshop on Applications and Advance of Problem Frames, pp.39-48(2004).
[11] Choppy, C. and Reggio, G.: A UML-based approach for problem frame oriented software development, Information and Software Technology, Vol.47, Issue 14, pp.929-954(2005).
[12] Seater, R. and Jackson, D.: Problem Frame Transformations: Deriving Specifications from Requirements, Proc. 2nd International Workshop on Applications and Advances of Problem Frames, pp.71-80 (2006).
[13] Rapanotti, L., Hall, L.G. and Li, Z: Deriving Specifications from Requirements through Problem Reduction, IEE Proc. Software, Vol.153, No.5, pp.183-198 (2006).
[14] Li, Z.: Progressing Problems from Requirements to Specifications in Problem Frames, Proc. 3rd International Workshop on Applications and Advances of Problem Frames, pp.53-50 (2008).
[15] Jackson, D.: Alloy: A Lightweight Object Modeling notation, ACM Trans. Software Engineering and methodology, Vol.11, No.2, pp.256-290 (2002).
[16] 紫合治: 安定状態と優先イベント規定によるコントローラ生成, 情報処理学会論文誌, Vol.56 (2015).