情報処理学会論文誌 Vol.56 No.2 555–568 (Feb. 2015)

安定状態と優先イベント規定によるコントローラ生成

紫合 治[1,a)]

受付日 2014年5月19日, 採録日 2014年11月10日

概要：環境の規定に基づき，その環境を適切に制御するコントローラを生成する方式として，Uchitel らは，シナリオと FLTL（Fluent Linear Temporal Logic）による環境規定からコントローラ生成する手法を提案している．しかし，一般のソフトウェア技術者にとっては，FLTL の適切な規定は難しく，Uchitel らの例題でも，何度かの試行錯誤を経て解に行きついている．ここではより簡単で分かりやすい手法として，プロブレムフレームの枠組みでコントローラ生成をとらえ，ドメイン規定の LTS（Labeled Transition System）と，安定状態と優先度による要求規定からコントローラの LTS を生成する手法について提案する．

キーワード：コントローラ生成，状態マシン，プロブレムフレーム，要求分析

Controller Synthesis by Stable States and Event Priorities Rules

Osamu Shigo[1,a)]

Received: May 19, 2014, Accepted: November 10, 2014

Abstract: To synthesize the controller adequately managing the environment from the constraint description of the environments, Uchitel et al., proposed an automated method based on the scenario and FLTL (Fluent Linear Temporal Logic) descriptions. However, FLTL seems to be too difficult for normal software engineers to completely describe the environment constraints. The authors of the method also needed to repeatedly describe and modify the FLTL descriptions to obtain the final result. This paper proposes more understandable and simpler method of the controller synthesis based on the problem frames model. The environment constraints are described in LTSs (Labeled Transition Systems) as domain properties, stable states and event priorities as the problem requirement.

Keywords: controller synthesis, state machine, problem frames, requirement analysis

1. はじめに

信頼性の高いソフトウェアを開発するには，事前に十分問題を分析することが重要である．特に組込みソフトウェアの場合，制御の対象となるセンサや装置，さらにそれらに関連するガスや液体等の特性まで分析し，どんな場合にどんなことが起こるかについて漏れなく調べあげることが重要になる．これらシステムの環境の特性を厳密に規定することによって，その環境を適切に制御するための条件が明確になる．

環境の規定から，環境を適切に制御するコントローラを自動的に生成する手法が研究されている [1], [2], [3], [4]．そこでは，環境の規定として，環境からのイベント（センサ入力等）と環境へのイベント（アクチュエータ出力等）を設定し，そのイベントの性質（property）を規定することにより，それらの性質を条件とし，その条件を満たすコントローラを見い出す．このとき，コントローラはイベントをラベルとした LTS（Labeled Transition System）で表される．

Uchitel ら [4] は，イベントの性質の規定として，イベントのシーケンス図による動作例のシナリオで，最低限必要な遷移規定を，また，イベントによって定義された環境の状態（Fluent）を使った線形時相論理 FLTL（Fluent Linear Temporal Logic）による安全規定（safe property）で，許される最大限の遷移規定を定め，これらの規定によって

[1] 東京電機大学情報環境学部
School of Information Environment, Tokyo Denki University, Inzai, Chiba 270–1382, Japan
[a)] o-shigo@mub.biglobe.ne.jp

徐々に正しいコントローラを生成する手法を示している．この手法の中で最も難しいところは，FLTL によるプロパティ規定と考えられる．特に，通常のソフトウェア技術者にとって，時相論理式で性質を規定することは簡単ではない [5]．Uchitel らの論文 [4] のケーススタディでも，何度かの規定の追加修正を繰り返し，最終的なコントローラ生成に至ったことが述べられており，FLTL 記述が簡単ではないことを示している．

環境の性質を調べ，さらに環境の振舞いに対する要求を規定し，それからシステムの仕様を求める方式として，Jackson のプロブレムフレーム [6] がある．その中で，組込みシステムのモデルとして適用される，必要とされる振舞いフレーム（Required Behavior Frame）においては，環境となるセンサやアクチュエータをドメインとし，それらのプロパティと，環境に対する制約としての要求規定から，システムであるマシンの仕様を求める手法 [7], [8] は，コントローラ生成ととらえることができる．この場合，コントローラ生成問題における環境の規定は，所与で直説法記述（どうなっているか）の**ドメインプロパティ**と，願望法記述（どうしたいか）の**要求**に分けられ，規定すべき内容のガイダンスにもなる．ただ，これらの要求・仕様変換の研究 [7], [8] は，ドメインプロパティや要求を述語論理等で形式的に記述する必要があり，一般のソフトウェア技術者が実務で適用するのは簡単ではないと思われる．

ここでは，プロブレムフレームの必要とされる振舞いフレームのモデルに沿ったコントローラ生成の手法を提案する．まず，ドメインプロパティとしてドメイン（センサやアクチュエータ等）ごとに LTS でイベントのプロトコルを規定する．多くの組込み技術者にとって，LTS のような状態マシンは使い慣れたもので，理解しやすいといえる [9]．次に，ドメインの振舞いに対する要求として，コントローラの安定状態で，ドメインがどうなっているべきかを規定する．さらに，不安定状態での優先すべき出力イベントを規定する．これらの規定は，時相論理の規定より簡単で，かつ多くの組込みシステムで使えるものである．ここで安定状態とは，環境からの入力イベント（センサの反応，操作者による操作等）を待っている状態で，入力イベントが発生しなければ何もしない．ただし，タイマの時間切れイベントは，環境からの入力イベントととらえる．安定状態でない状態を不安定状態（または一過性状態）といい，ここでは時間が経過しないと考える [3]．UML [10] の状態マシンでは，状態はつねに入力イベント待ちであり，我々の安定状態に対応する．不安定状態は，いわば遷移中の処理の途中の状態で，UML では状態としては扱われない．

一般には，安定状態と優先イベントだけで，十分な要求を規定できるわけではない．しかし，これによって従来 FLTL 等で規定していた多くの条件が比較的簡単に規定できる．さらに，生成されたコントローラの状態に対して，環境状態が明示されることにより，その意味が理解しやすくなるという効果も期待できる．

以下に，2 章で関連研究について述べ，3 章で提案手法の基本的なアイデアである状態指向の状態マシンと，手法の概要を説明し，4 章で基本モデルを形式的に述べ，5 章で提案手法によるコントローラ生成の例を Uchitel らのケーススタディと対比して説明し，6 章では提案方式の利点や問題点について議論し，最後に 7 章でまとめと今後の問題について述べる．

2. 関連研究

コントローラ生成とは，環境の規定や環境のあるべき振舞いから，そのような振舞いをさせるように環境を制御するコントローラを自動生成することである．これには，シナリオ等，具体例をいくつか与えて，それらを満たす振舞いの一般化規則を状態マシンとして生成する方法と，環境の制約を時相論理（FLTL）等で規定し，その規定を満たす状態マシンを生成する方法がある．

シナリオからの生成では，シーケンス図によるシナリオ規定からのイベント列を取り出し，イベント列中のイベントとイベントの間を状態として，イベント列から状態遷移（(状態，イベント，状態）の組）を生成する．このとき，同じ状態の識別方法が問題になるが，Whittle ら [1] は OCL（Object Constraint Language）によって，状態ベクタの値に対するイベントの影響を規定し，その規定から各状態での状態ベクタの値を計算し，状態ベクタの値が等しい状態を同じ状態と判定する手法を提案している．この場合，状態ベクタの定め方や，OCL 記述の適切さが，生成されたコントローラの正確さを決めることになるが，適切な OCL 記述は簡単ではない．

Sibay ら [2] は，シナリオが始まるときの状態とイベントをトリガとし，その後のシナリオを存在規定と全称規定に分けて表す方法を提案している．ここで存在規定では，示されたシナリオがありうることを，全称規定では，示されたシナリオでなければならないことを規定する．前者は動作例を，後者は動作ルールを示すことができる．これらのシナリオ方式では，提示するシナリオが多いほど，より正しいコントローラが生成されるが，どれだけシナリオを示せば十分であるかの判断は難しい．

Letier ら [3] は，Deontic Input-Output Automata（DIOA）の概念によって，制約からコントローラを得る方式を提案している．DIOA では，イベントを入力と出力に分け，かつ状態を安定（stable）状態と一過性（transient）状態に分けて，一過性状態からは出力遷移の繰返しで，安定状態に行けること（independence progress rule）を条件とする．ここで，一過性状態とは，そこで時間が経過しない状態で，安定状態とは時間が経過する状態とする．いい換えると，タイマの tick イベントを受け付けるのが安定状

態，受け付けない（tick でエラーになる）のが一過性状態である．なお，我々のモデルでは，一過性状態を不安定状態と呼ぶ．Letier らは，環境の規定を DIOA で記述し，それに制約条件を FLTL で記述する方法を提案している．ここで，FLTL は Fluent という状態規定を使った時相論理で，Fluent は真となるトリガのイベント集合と偽となるトリガのイベント集合，さらに初期状態が真か偽かを規定することで定義する．FLTL の記述から，その制約を表現する Buch Automata を生成し，これらのオートマトンの合成によって，FLTL による制約を満たした DIOA（コントローラ）を生成する．

シナリオによる方法と制約記述による方法を組み合わせたものとして，Uchitel ら [4] の提案がある．ここでは，シナリオ（拡張シーケンス図）によって，コントローラが最低限やらなければならないことを規定する．コントローラとして，最下限は何もできないもの，最上限は何でもできるものを考えると，シナリオを追加するごとに，やるべきことが増え，下限を上げることになる．また，FLTL による制約記述によって，コントローラがやってはならないことを規定し，上限を下げていく．シナリオ規定では必要な遷移（required）が，FLTL 制約ではやってはならないことの規定から，できること，すなわち可能な遷移（maybe）が定まる．このように，LTS の遷移を required と maybe に分けた MTS（Modal Transition System）を使って，シナリオや FLTL 記述を徐々に追加・修正しながら，徐々に maybe 遷移を削除または required 遷移に変えていき，最後にすべてが required 遷移になる MTS（つまり LTS）をコントローラとして得る．この方法で最も難しいのは，適切な FLTL の記述である．まず，適切な Fluent を設定しなければならない．多くの場合，環境の要素の状態が Fluent の候補になるが，複数の要素の状態の組合せや，いくつかの状態の和の状態等を Fluent にすると，1 つの環境要素の性質を規定するだけでも複雑な FLTL が必要になる場合が生じる．

3. プロブレムフレームとコントローラ生成

ここでは，まずコントローラ生成問題の枠組みを定めるためにプロブレムフレーム，特に「必要とされる振舞いフレーム（Required Behavior Frame）」について説明し，その枠組みに沿って，振舞いを規定するための基礎となる「状態指向状態マシン」について述べたあと，それらに基づいたコントローラ生成手順を提案する．

3.1 プロブレムフレーム

プロブレムフレーム [6] は，システムの開発において，開発すべきシステムについて考察するまえに，システムによって制御される部品や装置等の環境について十分に調べ，それらの環境がどのように振る舞うべきかを規定するという考え方である．これらの環境やその振る舞い方は，元の問題そのものに属し，問題の解である開発すべきシステムの外にあると考える．つまり，解を考える前に問題を十分分析するという方法である．Jackson はさまざまな問題を 5 つの型（Problem Frames）に分類し，それぞれについて考察している [6]．ここでは，このうち，組込みシステムによく現れる必要とされる振舞いフレーム（Required Behavior Frame）に限定して考察する．

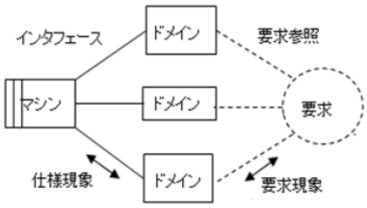


図 1 プロブレム図
Fig. 1 Problem diagram.

プロブレムフレームで基本となるものがプロブレム図である．**図 1** にプロブレム図の構成を示す．図で，2 本の縦線を持つ要素をマシンと呼び，問題の解決策を担当する．マシンと関連する四角の要素をドメインと呼び，環境となる部品や装置を表す．また，破線の楕円で，これらのドメインに対する要求を表す．基本的なプロブレム図では，マシンと要求がそれぞれ 1 つ存在し，ドメインには，マシンと直接関係するもの（モータやセンサ等）と間接にしか関係しないもの（水槽の水，ガス，信号制御システムでの車，運転手等）がある．間接的なドメインは，問題世界の深いところにあり，マシンに近くなるほど浅いドメインといわれる．ここでは，簡単のため間接的なドメインは記述しないという制限を加える．この制限により，要求はマシンが直接制御するドメインに対して記述することになる．

マシンとドメインをつなぐ実線をインタフェースと呼びマシンがドメインと関連していることを表す．また，要求とドメインをつなぐ破線を要求参照と呼び，要求がドメインに対して要請することを表す．ここでは，要求とマシンは直接的な関連を持たないという制限を設ける．つまり，要求は解決策に対しては何も要請しないものとする．

インタフェースや要求参照にそって，それらの線でつながれた要素間で共有する現象が規定される．インタフェースには，マシンが直接感知したり発生させたりできる仕様現象を，また要求参照には，分析者が要求を表現するための要求現象を規定する．仕様現象は，マシンが決めてドメインに渡す出力現象（マシンにとっての出力）と，ドメインが決めてマシンに渡す入力現象（マシンにとっての入力）がある．出力は，アクチュエータへの指令，入力は操作者の指令やセンサのイベント等からなる．一方，要求現象は，装置がどうなっているかの状態を示すことが多い．**表 1** に，Jackson の本 [6] に現れるいくつかのドメインの

表 1 仕様現象と要求現象
Table 1 Specification phenomena vs. Requirement phenomena.

問題	ドメインの例	仕様現象	要求現象
片側交互通行用信号	信号機	赤と緑のランプへのパルス	進め状態，止れ状態
走行距離計表示	走行車	車の回転毎に出るパルス	スピードと走行距離
周期と範囲の入力	周期＆範囲	編集操作（追加，修正，削除，読出等）	データ値
炉の操作	炉設備	炉の操作（点火，送風等）	燃焼中，停止等の状態
水門制御	門扉＆モータ	逆（順）回転，On/Off，位置	閉鎖状態，開門状態等

例について，仕様現象と要求現象を示す．この例では，すべて仕様現象はイベント，要求現象は状態となっている．つまり，マシンの処理にとってはドメインとやりとりするイベントが，また分析者による要求記述にとってはドメインがどうなっているかを示す状態が扱いやすい現象であるといえる．

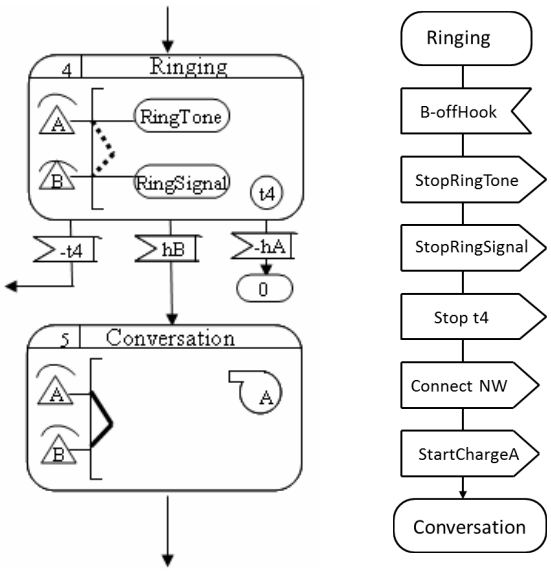
各ドメインに対して，その仕様現象と要求現象の関連をドメインプロパティとして規定する．仕様現象がイベントで要求現象が状態の場合，ドメインプロパティはイベントによる状態の変化の様子を LTS 等の状態マシンの形で表す．

コントローラ生成の観点からは，マシンがコントローラでドメインが環境になり，ドメインプロパティと要求に対する規定が環境の規定となり，それからマシンの仕様を生成することがコントローラ生成となる．つまり，仕様現象（イベント）による振舞いを，ドメインプロパティや要求で定められた条件を満たすように規定することが，コントローラ生成となる．プロブレムフレームのプロブレム図は，コントローラ生成問題の枠組みの規定に使うことができる．

3.2 状態指向状態マシン

SDL（Specification and Description Language）[11] は通信システム等の厳密な仕様記述のための図式言語として広く使われている．その初期の版 [12] では，状態の中に電話機，トーンやベルの発生器，回線，課金装置，タイマ等の絵（Pictorial Elements）を描き，その状態における制御対象装置の状況を表示した，状態指向の状態マシンが規定されていた（現在の版の SDL [11] では，Pictorial Elements や状態指向の概念は削除されている）．**図 2** (1) に，電話交換機の呼処理の一部の**状態指向**の状態マシンを，また図 2 (2) に (1) と等価な**遷移指向**の状態マシンを示す．

図 2 (1) で，遷移にともなう処理は状態間の Pictorial Elements の差分によって処理の順序を除いて暗黙的に示される．図 2 (2) のように，遷移指向の状態マシンでは状態の



(1)状態指向の状態マシン　　(2)遷移指向の状態マシン

図 2 SDL における状態指向と遷移指向の状態マシン
Fig. 2 State oriented and transition oriented state machines.

直後に入力イベント（図では B-offHook）が表れ，その次に出力イベント（StopRingTone 等装置への命令）の列が続き，次の状態（Conversation）に至る．状態は入力イベントを待ち，入力イベントが発生しなければ何もしない．我々のモデルや前節で述べた DIOA に対応すると，SDL の状態はすべて安定（Stable）状態になる．SDL では一過性の不安定状態は状態としては扱われないで，状態を遷移する途中，たとえば図 2 (2) で，B-offHook と StopRingTone の間，StopRingTone と StopRingSignal の間，ConnectNW（回線接続）と StartChargeA（A の課金開始）の間等が，不安定状態となる．不安定状態では時間が経過しないものとし，その間では要求としては正しくないことも許す．たとえば回線接続の直後かつ課金開始の直前では「回線接続中は発呼電話 A に課金すること」という要求は満たされないが，この間は不安定状態で時間は経過しないので問題なしとする．

図 2 (1) を検討すると，状態中の Pictorial Elements 等から以下のようなことが分かる．ただし，事前に Pictorial Elements を状態とした各装置（電話機 A，電話機 B，トーン・ベル発生器，回線，タイマ，課金装置等）に対するプロパティが LTS 等の形で規定されているものとする．以下で，入力イベントによる状態遷移を**入力遷移**，出力イベントによる状態遷移を**出力遷移**と呼ぶ．

(1) 安定状態の Pictorial Elements から，そこで発生しうるすべての入力イベントが分かる[*1]．たとえば，図 2 (1) の Ringing では，電話機 A の onHook (-hA)（電話機 A

[*1] ここでは，各ドメインの状態が変われば，その状態を表す Pictorial Element は変化するものとする．これによって，Pictorial Element を見れば（描かれていない場合も含めて）各ドメインの状態が一意に定まる．

がオンになっているから），電話機 B の offHook (hB)（電話機 B がオフになっているから），タイマ t4 のタイムアウト (-t4)（t4 がオンだから）の 3 つの入力イベントが発生しうることが分かる（このほかのトーン発生器や回線等からは，それぞれのプロパティ規定より入力イベントは発生しない）．Ringing 状態では，これらの入力イベントのすべてを受け付ける必要がある．つまり，Pictorial Elements を調べれば，入力遷移の漏れを発見することができる．

(2) 安定状態で入力イベントが発生した直後は，一般的にある不安定状態になる．たとえば，Ringing 状態で B offHook（電話機 B をオンにする）が発生した場合，A，B 両電話機がオンになっているのに，まだリングトーンやベルが鳴っており，回線は予約のまま等になる．A，B 両電話機がオンになる安定状態は通話中（Conversation）のみであるので，交換機はこの不安定状態を速やかに脱し，通話中に至るように制御する．このため，トーンやベル，タイマの停止，回線の接続，課金の開始を行い（それぞれのドメインに出力イベントを出すことによる），通話中に至る（図 2 (2)）．これらの処理は，順序を別にして Ringing と Conversation の Pictorial Elements の差分から自動的に生成できる．つまり，交換機（コントローラ）の処理とは，安定状態で入力イベントが発生し不安定状態になったとき，そこから次の安定状態に遷移する目的で行われると考えることができる．このため，不安定状態からは出力遷移の 0 回以上の繰返しによって，つまりコントローラの意思による制御によって，いずれかの安定状態に遷移できるようになっていなければならない（Deontic Input-Output Automata の Independent progress rule に相当する [3]）．

さらに，SDL における安定状態の定義から，以下のことがいえる．

(3) すべての安定状態は，初期状態（交換機の場合はアイドル（Idle））から到達可能であること．

(4) 安定状態からの出力遷移は存在しない．つまり，安定状態ではつねに入力イベントを待ち，入力がなければ永遠に何もしない．多くの場合，タイマによるタイムアウトイベントが働くので永遠に待つということはない．我々のモデルではタイマも環境のドメインととらえ，タイムアウトは入力イベントとするため，すべての安定状態では出力遷移は存在しないものとする．

(5) (2) の条件を少し強化すると，不安定状態からの遷移はいずれかの安定状態に向かうものに限ると考えてもよい．つまり，不安定状態はあくまでも安定状態に向かう一過性の状態であるとして，そこにとどまることを許さないものとする．たとえば，Ringing から Conversation に至る途中の不安定状態では，Pictorial Elements から回線は A-B 接続状態になるため，回線の予約をキャンセルする出力イベントの遷移は安定状態に向かうものではなく，これらの不安定状態でコントローラが使える出力遷移から除くことができる．

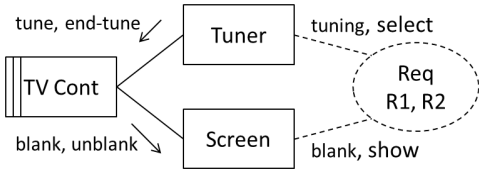


図 3 簡単なテレビジョンシステムの問題図
Fig. 3 Problem diagram of simple television system.

3.3 コントローラ生成の概要

ここでは，プロブレムフレームの枠組みと状態指向の状態マシンによる振舞い規定を使ったコントローラ生成の概要を，手順に従って述べる．提案手法によるコントローラ生成手順は以下のようになる．

① 問題の構造をプロブレム図で表現
② 各ドメインに対してドメインプロパティ（LTS）記述
③ ドメインプロパティの合成（直積）を生成
④ 安定状態を規定
⑤ 安定状態の条件を満たすコントローラを生成
⑥ その結果を見て（必要なら）優先条件を規定
⑦ 優先条件を満たすコントローラを生成
⑧ 必要ならコントローラの遷移にガード条件を規定

このうち，③，⑤，⑦はツールにより自動化できる．また，⑧はここでは取り上げない（6 章 (4) で議論する）．なお，実際の適用では①から⑧に順に進むだけでなく，たとえば⑤でエラーが発生し，そのために②に戻ってドメインプロパティを修正したり，④に戻って安定状態を追加修正したりとの手戻りが発生することも多い．この場合は，文献 [4] のケーススタディ同様に，手順を繰り返しながら徐々に最終版を得ることになる．

以下，説明のため，簡単なテレビジョンシステム [3], [13] を例に使う．**図 3** にテレビジョンシステムのプロブレム図を示す．ドメインには Tuner と Screen があり，Tuner から tune と end-tune の入力イベントが，Screen へ blank と unblank の出力イベントが規定され，要求現象として Tuner は tuning（選局中）と select（選局済），Screen は show（表示）と blank（暗転）の状態が規定されているものとする．

3.3.1 ドメインプロパティの規定

環境のドメインごとにドメインプロパティを規定する．ドメインプロパティは，ドメインの仕様現象（イベント）をラベル，要求現象（状態）を状態とした LTS で規定する（4.1 節参照）．ドメインプロパティの LTS は決定的とする．これによって，初期状態とイベント列から，遷移後

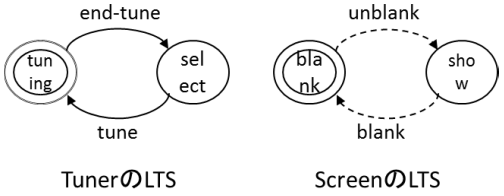


図 4 テレビジョンシステムのドメインプロパティ
Fig. 4 Domain properties of the television system.

の状態が一意に定まる．また，ドメインプロパティの規定では，ドメイン間の直接的な関連はないものとする（関連は要求に記述）．これによって複数のドメインを合成した場合も，イベント系列の履歴から各ドメインの状態が一意に定まることが保証される．

イベントは，センサ等からの入力イベントとアクチュエータ等への出力イベントに分けられる．ある状態から入力遷移がある場合，ドメインがその状態になった場合にその入力イベントが発生しうることを示し，コントローラはその入力を受け取って処理する体制になっている必要がある．つまり，入力イベントはコントローラの義務を示す．一方，ある状態から出力遷移がある場合，ドメインがその状態になった場合にその出力イベントをドメインに送ることができる（ドメインはそれを受け取る準備が整っている）ことを示す．つまり，出力イベントはコントローラができること（権利）を規定する．1 つの状態から入力遷移と出力遷移の両方がある場合，入力イベントが来る前であれば出力イベントを出すことができると考える．

テレビジョンシステムの Tuner と Screen のドメインプロパティ LTS を**図 4** に示す．図で 2 重の円は初期状態，実線は入力遷移，破線は出力遷移を示す．ここでは，文献 [3] とは異なり，tune や end-tune はいつでも発生するのでなく，tune は select 状態のとき，end-tune は tuning 状態のときのみ発生すると規定している．つまり，ここでは文献 [3] よりも多くの情報をドメインプロパティに持たせている．

3.3.2 ドメインプロパティの合成

コントローラ生成の準備としてドメインプロパティを合成し，環境全体の規定を作成する．プロブレム図でドメイン間の直接的な関連はないとした場合，合成は LTS の直積となる．**図 5** にテレビジョンシステムの Tuner と Screen を合成した LTS を示す．ここで状態は，ドメイン状態の直積になっている．

ドメインを合成した LTS は環境全体の規定となり，コントローラ生成に対する条件を示している．最終的なコントローラは，すべてのドメインの入力イベントと出力イベントをラベルとした LTS で表される．また，コントローラ LTS の状態は，すべてのドメイン状態の直積で表す．この場合，コントローラはドメイン状態が同じ 2 つ以上の状態を持つことができない．そこで，ドメイン状態が同じものを 2 つ以上必要になる場合は，特別にそれを区別するためのドメイン（変数に対応）を追加することで対応する．たとえば，「片側交互通行信号制御」の問題 [6] では，左向き通行後の全停止と右向き通行後の全停止は，ドメイン状態は同じ（2 つの信号がともに停止）だが，別状態とした方が制御しやすいので，新たに右向きと左向きの 2 状態を持つ方向ドメインを導入し，2 つの全停止状態を区別する．

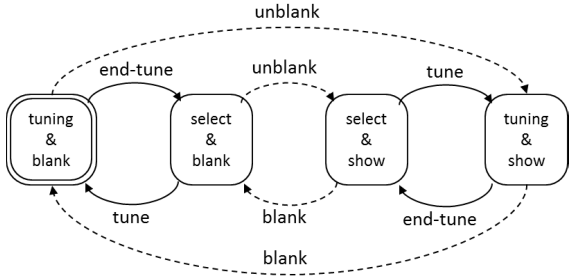


図 5 ドメインプロパティの合成 LTS
Fig. 5 LTS synthesized from the domain properties.

コントローラに対する要求の制約がまったくない場合，このドメインプロパティの合成結果がコントローラ LTS になる．これは，ドメインが許す可能なすべての遷移を含む最大限の LTS である．

3.3.3 要求の規定

要求は，安定状態と優先イベントによって規定する．前者は機能の概要を示し，後者はより詳細な要求に対応する．安定状態と優先イベントによって，すべての要求を規定できるわけではないが，システムによってはこの 2 つでほとんどの要求をカバーする（5 章のケーススタディ）．

(1) 安定状態

コントローラの状態を，入力待ちのみの安定状態と，出力を行うことができる不安定状態に分ける．システムの振舞いを考察する場合，分析者等，人間にとっては安定状態が考えやすいといえる．

UML [10] や SDL [11], [12] の状態マシンでは，原則として状態＝安定状態ととらえ，不安定状態は，状態遷移処理中の途中経過点になり状態として扱わないが，ここでは，Uchitel らのコントローラ生成手法に合わせて，不安定状態も状態として扱う．

プロブレムフレームの要求現象（状態）で規定する要求として，ドメインの直積状態から安定状態をリストアップする．テレビジョンシステムでは，Tuner が tuning 状態で Screen が black 状態のときと，select 状態で show 状態のときを安定状態とする（**図 6** の緑色の状態）．これらの安定状態は，テレビジョンシステムの次の 2 つの要求 [3],

R1：Screen は tuning 中のノイズを放映してはならない

R2：選局後はできるだけ速やかに Screen に放映する

に対応している．なお，放映中に tune イベントが発生すると，一時的に tuning 中で Screen が show となり要求 R1 を満たさないが，この状態は一過性の不安定状態であるの

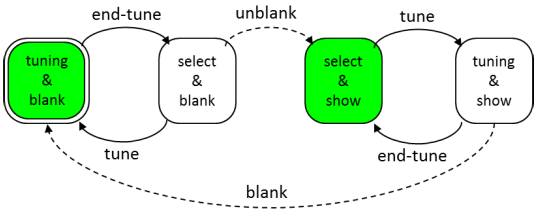


図 6 テレビジョンシステムのコントローラ LTS
Fig. 6 Controller LTS of the television system.

で問題なしとする [3].

3.2 節で述べたが，安定状態を定めることによって，遷移に関する以下の条件が成立する．

① 安定状態からの入力遷移はすべて必要，
② 不安定状態からは出力遷移の 0 回以上の繰返しでいずれかの安定状態に到達可能
③ 初期状態からすべての安定状態に到達可能
④ 安定状態は入力遷移のみ
⑤ 不安定状態からの出力遷移は，安定状態に向かうもののみとする

上で，①～⑤は，3.2 節の (1)～(5) をそれぞれまとめたものに対応している．このうち，①～③は，必要な遷移を示し，④と⑤は不要な遷移を示す．言い換えると，①から③は，コントローラ LTS の下限（最低限やるべきこと）を上げ，④と⑤は LTS の上限（最大限やってもよいこと）を下げる．コントローラ生成手順では，この①～⑤の条件を満たすようにドメインプロパティを合成した LTS を修正する．この修正はツールによって自動的にできる．

テレビジョンシステムの例では，④の条件より，安定状態からの出力遷移を削除した図 6 に示すコントローラ LTS が得られる．これは上の条件①～⑤をすべて満たす．図 6 の結果では，不安定状態からも入力遷移があるが，SDL や UML の状態マシンでは不安定状態からの入力遷移は扱わない．この結果，たとえば UML ではテレビジョンシステムのコントローラは，次の 2 つの遷移を持つ簡単な状態マシンになる（遷移中の “/” は入力/出力を表す）．

- tuning&blank—end-tune/unblank→select&show
- select&show—tune/blank→tuning&blank

(2) 優先イベント

不安定状態における出力遷移は，コントローラの意思で行うものであるが，ある不安定状態からの遷移が複数ある場合，どの出力イベントを優先するかを規定することができる．優先イベント規定は，不安定状態の条件（ドメイン状態の条件）と優先する出力イベントを規定することにより，条件に合った不安定状態から優先する出力遷移があった場合，その状態からの他の遷移をすべて削除する．

テレビジョンシステムの例では，出力イベントはそれぞれ 1 つしかなく，優先順位は関係ない．3.2 節で述べた電話交換機の例では，Ringing から Conversation に遷移する場合の出力イベントの優先順位は，トーンやベルの停止＞タイマ停止＞回線接続＞課金開始となる．要求によっては，収益を増やすために課金開始を最優先にするかもしれない．

4. 基本モデル

ここでは 3.3 節で述べたコントローラ生成手法を，形式的に定義する．なお，この基本モデルによって，手法を支援する自動化ツール（ドメインプロパティの設定，合成（直積），安定状態の設定，安定状態による条件を満たす条件の検査やその条件に合うように LTS を修正，優先イベント規定に基づく LTS の修正等）のアルゴリズムが明確になる．

4.1 環境の規定

制御すべき環境はいくつかのドメイン（センサやアクチュエータ等）から構成される．これらのドメインの間の関連は，明示的には分かっていないものとし，ドメインはそれぞれ独立したものとして扱う．

ドメインの規定をドメインプロパティと呼び，以下のような LTS で表す．

[定義 1. ドメインプロパティ]
ドメインプロパティは LTS $D = (Q, \Sigma, \delta, q_0)$ で表す．

- Q：状態の有限集合
- Σ：イベントの有限集合．イベントは互いに素な入力イベント Σ_I と出力イベント Σ_O よりなる．

$$\Sigma = \Sigma_I \cup \Sigma_O, \quad \Sigma_I \cap \Sigma_O = \phi$$

- $\delta \subset Q \times \Sigma \times Q$：遷移規則
- $q_0 \in Q$：初期状態

□

δ は入力イベントによる入力遷移 δ_I と出力イベントによる出力遷移 δ_O に分けられる．

- $\delta_I = \{(q, a, r) \in \delta | a \in \Sigma_I\}$ ：入力遷移
- $\delta_O = \{(q, a, r) \in \delta | a \in \Sigma_O\}$：出力遷移

δ は決定的である．つまり，$(q, a, r) \in \delta$ かつ $(q, a, t) \in \delta$ なら，$r = t$ となる．

δ の推移閉包 $\delta^* \subset Q \times \Sigma^* \times Q$ を以下のように定義する．

① $\forall q \in Q : (q, \varepsilon, q) \in \delta^*$（$\varepsilon$ は空列を表す）
② $(q, a, r) \in \delta$ かつ $(r, w, s) \in \delta^*$ なら $(q, a.w, s) \in \delta^*$

環境に n 個 $(n \geqq 1)$ のドメインが存在するとする．ドメイン k $(1 \leqq k \leqq n)$ のドメインプロパティを，$D^k = (Q^k, \Sigma^k, \delta^k, q_0^k)$ で表す．このとき，Σ^k は互いに素（$i \neq j$ なら $\Sigma^i \cap \Sigma^j = \phi$）とする．つまり，ドメイン間の直接的な関連はないものとする．

4.2 ドメインプロパティの合成

コントローラ生成の準備として，4.1 節で定めた n 個のドメインに対するドメインプロパティ $D^1, D^2, \ldots, D^n$ の合

成を作る．ここで，ドメイン間の関連はないため，ドメインプロパティの合成は直積と等しくなる．

[定義 2. ドメインプロパティの合成（直積）]

ドメインプロパティ $D^1, D^2, \ldots, D^n$ の合成は，次のような LTS $C_B = (Q_{CB}, \Sigma_{CB}, \delta_{CB}, q_{CB0})$ である．ここで，

- $Q_{CB} = Q^1 \times Q^2 \times \ldots \times Q^n$：ドメインの状態の直積集合
- $\Sigma_{CB} = \Sigma^1 \cup \Sigma^2 \cup \ldots \cup \Sigma^n$：ドメインのイベントの和
  ドメインイベントは互いに素（$i \neq j$ なら $\Sigma^i \cap \Sigma^j = \phi$）とする．
  イベントは入力と出力に分けられる．
  $\Sigma_{CBI} = \Sigma_I{}^1 \cup \Sigma_I{}^2 \cup \ldots \cup \Sigma_I{}^n$　：入力イベント
  $\Sigma_{CBO} = \Sigma_O{}^1 \cup \Sigma_O{}^2 \cup \ldots \cup \Sigma_O{}^n$：出力イベント
- $\delta_{CB} = \{((q^1, q^2, \ldots, q^n), a, (r^1, r^2, \ldots, r^n))|$
  $q^i, r^i \in Q^i (1 \leqq i \leqq n), a \in \Sigma^k (1 \leqq k \leqq n)$ のとき，$(q^k, a, r^k) \in \delta^k$ かつ $q^j = r^j$ $(1 \leqq j \leqq n$ で $j \neq k$ のとき$)\}$
- $q_{CB0} = (q_0{}^1, q_0{}^2, \ldots, q_0{}^n)$：初期状態はドメインの初期状態の組

□

δ_{CB} は入力イベントによる入力遷移 δ_{CBI} と出力イベントによる出力遷移 δ_{CBO} に分けられる．

- $\delta_{CBI} = \{(q, a, r) \in \delta_{CB} | a \in \Sigma_{CBI}\}$　：入力遷移
- $\delta_{CBO} = \{(q, a, r) \in \delta_{CB} | a \in \Sigma_{CBO}\}$：出力遷移

[定義 3. もとの遷移]

$(q, a, r) \in \delta_{CB}$ に対して，$a \in \Sigma^k$（ドメイン k の信号）で，$q = (q^1, q^2, \ldots, q^n)$, $r = (r^1, r^2, \ldots, r^n)$ のとき，ドメイン k の遷移 $(q^k, a, r^k) \in \delta^k$ を (q, a, r) のもとの遷移という．

□

求めるコントローラ $C = (Q_C, \Sigma_C, \delta_C, q_{C0})$ は C_B から不要な遷移を取り除くことによって得られる．ここで，$Q_C \subset Q_{CB}$，$\Sigma_C \subset \Sigma_{CB}$，$\delta_C \subset \delta_{CB}$，$q_{C0} = q_{CB0}$ となる．Σ_{CI}（$\subset \Sigma_{CBI}$），Σ_{CO}（$\subset \Sigma_{CBO}$）を，コントローラ C の入力イベント，出力イベントとする．

4.3 要求の規定

ドメインプロパティの合成 C_B をもとに，コントローラを生成するために C_B に条件を加える必要がある．この条件をプロブレムフレームの枠組みに沿って「要求」と呼ぶ．ここでは，要求として安定状態と優先イベントを規定する．

(1) 安定状態

[定義 4. 安定状態]

安定状態 Q_S はドメイン状態の直積 Q_{CB} の部分集合である．Q_S は要求の一環として，分析者が定める．

$Q_S \subset Q_{CB}$：安定状態

□

[定義 5. 目標安定状態]

状態 q から出力遷移の 0 回以上の繰返しで到達できる安定状態を q の目標安定状態 $Q_T(q)$ と呼ぶ．

$Q_T(q) = \{r \in Q_S | \exists w \in \Sigma_{CBO}{}^*, (q, w, r) \in \delta_{CBO}{}^*\}$

□

[条件 1. 安定状態に対する条件]

安定状態に対して必要な遷移については以下の条件が成立する．なお，以下の①～⑤は，3.2 節の状態指向状態マシンでの考察 (1)～(5)，かつ 3.3.3 項 (1) の①～⑤に対応する．

必要な遷移に対して，以下の 3 つの条件が成り立つ．

① 求めるコントローラは，安全状態からの入力遷移をすべて含む．

$$\{(q, a, r) \in \delta_{CB} | q \in Q_S, a \in \Sigma_{CBI}, r \in Q_{CB}\} \subset \delta_C$$

② コントローラに含まれるすべての状態は目標安定状態を持つ．

$$\forall q \in Q_C : Q_T(q) \neq \phi$$

定義より，安定状態の目標安定状態は自分自身を含むので，これはすべての不安定状態は出力遷移の 0 回以上の繰返しで安定状態に遷移できること（Independent progress rule）を述べたものである．

③ 初期状態からすべての安定状態に到達可能である．

$$\forall q \in Q_S, \exists w \in \Sigma_C{}^* : (q_{C0}, w, q) \in \delta_C{}^*$$

また，不要な遷移については以下の条件が成立する．

④ 安定状態からは入力遷移のみ許される．

$$(q, a, r) \in \delta_C \text{ で } q \in Q_S \text{ なら，} a \in \Sigma_{CI}$$

⑤ コントローラの出力遷移は安定状態に向かう遷移に限る．

$\forall (q, a, r) \in \delta_{CO}, \exists s \in Q_T(q)$：
　$a \in \Sigma^k$，(q, a, r) のもとの遷移を $(q^k, a, r^k) \in \delta_k$，s のドメイン k の状態を $s^k \in Q^k$ としたとき，$(q^k, a, r^k) \in \delta_O{}^k[q^k \to s^k]$

ここで $\delta[q \to t]$ は，δ で q から t に向かう初めの遷移の集合を表す．

$\delta[q \to t] = \{(q, x, r) \in \delta |$
　$\exists (q_0, x_1, q_1), (q_1, x_2, q_2), \ldots, (q_{m-1}, x_m, q_m) \in \delta$ $(m \geqq 1)$, $q_0 = q, x_1 = x, q_1 = r, q_m = t$ かつ $q_i \neq q (1 \leqq i \leqq m)\}$

□

⑤の条件より，$(q, a, r) \in \delta_{CO} (a \in \Sigma^k)$ なら，$q^k \neq s^k$ となる q の目標安定状態 s が存在する．つまり，すべての目標安定状態でドメイン k の状態が変わらないなら，ドメイン k の出力遷移は不要ということになる．

(2) 優先イベント

優先イベント規定は，規定条件を満たす不安定状態から規定出力遷移がある場合，その遷移を優先することを要請する．

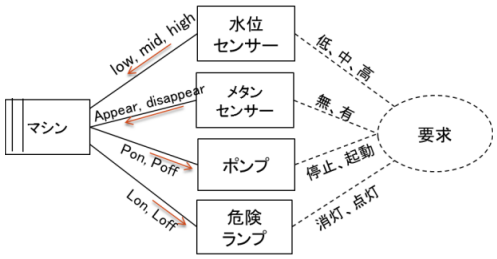


図 7 鉱山ポンプ問題のプロブレム図
Fig. 7 Problem diagram of mine pump system.

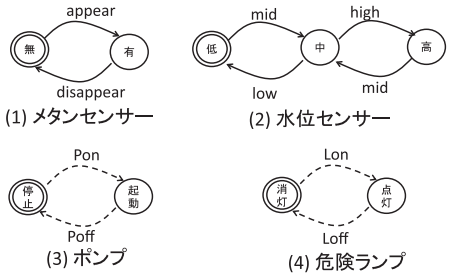


図 8 問題のドメインプロパティ LTS
Fig. 8 Domain property LTS of the mine pump system.

[定義 6. 優先イベント規定]
優先イベント規定 PE は，状態に対する条件と出力イベントのペアの集合である．

$$\mathrm{PE} \subset \text{条件} \times \Sigma_{\mathrm{CO}}$$

ここで，条件は状態に対する述語とする．

□

$(\mathrm{cond}, \mathrm{a}) \in \mathrm{PE}$ とすると，この優先イベント規定によって，

$$\{(\mathrm{q}, \mathrm{b}, \mathrm{s}) \in \delta_{\mathrm{C}} | \mathrm{cond}(\mathrm{q}) \wedge \exists \mathrm{r} \in \mathrm{Q_C} : (\mathrm{q}, \mathrm{a}, \mathrm{r}) \in \delta_{\mathrm{C}} \wedge \mathrm{a} \neq \mathrm{b}\}$$

が δ_{C} から削除される．つまり，条件 cond が成立する状態で a による遷移がある場合，その遷移を残して，他の遷移はすべて削除することが，優先イベントに対する処置となる．

5. ケーススタディ

前章で述べた方式を具体例に適用することで，方式の評価を行う．ここでは，Uchitel らのケーススタディの問題 [4] を取り上げ，Uchitel らの方式との比較を行う．

(1) 問題

鉱山におけるポンプ制御の問題を取り上げる．鉱山では坑内の出水を排除するための水抜き用ポンプを制御する．ポンプは水位が決められた限度を超えれば動作し，ある限度以下になれば停止する．ただし，坑内にメタンガスが発生しあるレベルを超えた場合は，爆発を避けるためにポンプを停止する．このとき坑内入口にある危険ランプを点灯する．コントローラは，坑内の水位とメタンガス濃度を検査するセンサからのイベントによって，ポンプや危険ランプを適切に制御する．**図 7** にこの問題のプロブレム図を示す．図で，マシンとドメイン間のインタフェースには仕様現象としてドメインのイベントを，ドメインと要求間の要求参照には要求現象としてドメインの状態を記述してある．

(2) 環境（ドメインプロパティ）の定義

プロブレム図に示すように，環境は水位センサ，メタンセンサ，ポンプ，危険ランプのドメインからなる．

① メタンセンサ

メタンセンサは坑内のメタンガス濃度を調べて安全か危険かを知らせる．イベントは appear と disappear の 2 種類で，ともにコントローラの入力となる．無（安全）と有（危険）を状態とした LTS を**図 8** (1) に示す．

② 水位センサ

水位センサは，坑内水位を 3 段階（低，中，高）で計測する．センサイベントは水位が段階を変化したことをコントローラに知らせる low，mid，high の 3 つの入力イベントからなる．水位センサの LTS を図 8 (2) に示す．

③ ポンプ

ポンプはコントローラによって制御されるもので，状態として起動と停止を持ち，イベント（コントローラの出力）は Pon と Poff となる．図 8 (3) にポンプの LTS を示す．

④ 危険ランプ

危険ランプもコントローラによって制御され，消灯と点灯の 2 状態を持ち，コントローラの出力イベントとしては Lon と Loff を持つ．図 8 (4) に危険ランプの LTS を示す．

(3) ドメインプロパティの合成

環境の 4 つのドメインの LTS の直積を作成する．これは，4 つのドメインの状態数の積である 24 個（3×2×2×2）の状態を持ち，104 の遷移を持つ．**図 9** に，状態遷移表によるドメインプロパティを合成した LTS を示す．ここでは，各状態にドメイン状態を明示し，イベントによる遷移の有無が分かりやすいようにした．たとえば，No.0 の状態では，メタンは無，水位は低，ポンプは停止，危険ランプは消灯となっているので，発生する可能性のある入力イベントは，メタンの appear と水位の mid の 2 つ，また出力できるイベントは，ポンプの Pon と危険ランプの Lon の 2 つになる．これ以外の遷移は，ドメインの規定より削除する．

(4) 安定状態規定

安定状態は，メタン濃度の状態と水位の組合せにおいて，ポンプと危険ランプがどうなっているべきかを決めることで定まる．規則としては，次の 2 つが考えられる．

R1：危険ランプは，メタンが無なら消灯，有なら点灯

R2：ポンプは，メタンが安全でかつ水位が中以上なら起動，それ以外なら停止

これより，メタンと水位の状態の組ごとに 1 つの安定状態が決まり，合計 6 つの安定状態が規定される（図 9 の 0，6，10，13，17，21 の状態．**図 10** では薄緑色で太文字の状態）．

状態					遷移								
状態	ドメイン状態				入力イベント					出力イベント			
No	メタン	水位	ポンプ	ランプ	メタン		水位			ポンプ		危険ランプ	
					app	disap	low	mid	high	Pon	Poff	Lon	Loff
0	無	低	停止	消灯	12			4		2		1	
1	無	低	停止	点灯	13			5		3			0
2	無	低	起動	消灯	14			6			0	3	
3	無	低	起動	点灯	15			7			1		2
4	無	中	停止	消灯	16		0		8	6		5	
5	無	中	停止	点灯	17		1		9	7			4
6	無	中	起動	消灯	18		2		10		4	7	
7	無	中	起動	点灯	19		3		11		5		6
8	無	高	停止	消灯	20			4		10		9	
9	無	高	停止	点灯	21			5		11			8
10	無	高	起動	消灯	22			6			8	11	
11	無	高	起動	点灯	23			7			9		10
12	有	低	停止	消灯		0		16		14		13	
13	有	低	停止	点灯		1		17		15			12
14	有	低	起動	消灯		2		18			12	15	
15	有	低	起動	点灯		3		19			13		14
16	有	中	停止	消灯		4	12		20	18		17	
17	有	中	停止	点灯		5	13		21	19			16
18	有	中	起動	消灯		6	14		22		16	19	
19	有	中	起動	点灯		7	15		23		17		18
20	有	高	停止	消灯		8		16		22		21	
21	有	高	停止	点灯		9		17		23			20
22	有	高	起動	消灯		10		18			20	23	
23	有	高	起動	点灯		11		19			21		22

図 9 ドメインの直積 LTS

Fig. 9 Direct product LTS of domain properties.

状態					遷移									Uchitel の対応状態NO
状態	ドメイン状態				入力イベント					出力イベント				
No	メタン	水位	ポンプ	危険ランプ	メタン		水位			ポンプ		危険ランプ		
					app	disap	low	mid	high	Pon	Poff	Lon	Loff	
0	**無**	**低**	**停止**	**消灯**	12			4		2		1		0
1	無	低	停止	点灯	13			5		3			0	3
2	無	低	起動	消灯	14			6			0	3		13
3	無	低	起動	点灯	15			7			1		2	17
4	無	中	停止	消灯	16		0		8	6		5		21
5	無	中	停止	点灯	17		1		9	7			4	4
6	**無**	**中**	**起動**	**消灯**	18		2		10		4	7		12
7	無	中	起動	点灯	19		3		11		5		6	16
8	無	高	停止	消灯	20			4		10		9		11
9	無	高	停止	点灯	21			5		11			8	5
10	**無**	**高**	**起動**	**消灯**	22			6			8	11		7
11	無	高	起動	点灯	23			7			9		10	6
12	有	低	停止	消灯		0		16		14		13		1
13	**有**	**低**	**停止**	**点灯**		1		17		15			12	2
14	有	低	起動	消灯		2		18			12	15		なし
15	有	低	起動	点灯		3		19			13		14	なし
16	有	中	停止	消灯		4	12		20	18		17		20
17	**有**	**中**	**停止**	**点灯**		5	13		21	19			16	19
18	有	中	起動	消灯		6	14		22		16	19		14
19	有	中	起動	点灯		7	15		23		17		18	18
20	有	高	停止	消灯		8		16		22		21		9
21	**有**	**高**	**停止**	**点灯**		9		17		23			20	10
22	有	高	起動	消灯		10		18			20	23		8
23	有	高	起動	点灯		11		19			21		22	15

：安定状態　　n ：安定状態条件での削除

n ：優先イベント(Poff)規定　　n ：優先イベント(Pon)規定

図 10 要求規定による遷移の制約

Fig. 10 Delete transitions by requirement constrains.

安定状態の規定により，3.3.3 項 (1) で述べた条件に沿って不必要な遷移を削除する（必要な遷移の規定は後でチェックする）．まず条件④より，安定状態の出力遷移を削除する．また，条件⑤により，不安定状態の不要な出力遷移を削除する．図 9 で，メタンと水位の状態が等しい状態のグループについては，グループ内の状態の目標安定状態はそのグループ内の安定状態となる．条件⑤より，安定状態と同じドメイン状態を持つ場合は，そのドメインのイベントによる遷移は不要となる．たとえば，No.0 を安定状態とするグループ（0，1，2，3）では，ポンプは停止，ランプは消灯で，No.1 のポンプ，No.2 の危険ランプの状態は No.0 と等しい．よって，No.1 のポンプの遷移と No.2 の危険ランプの遷移は削除する．この結果，もとの 48 の出力遷移が，半分の 24 に減った（図 10 のオレンジ色の状態）．

(5) 優先イベント規定

次に優先イベントの設定を行う．鉱山の問題では，出力イベント中で優先すべきものはメタン発生時や水位が低になったときにポンプを停止させること（ガス爆発やポンプの空回りによる惨事を防ぐため），およびメタン無で水位が高になったときのポンプを起動させること（坑内水没を防ぐため）と考え，以下の 2 つの規定を定める．

A) Pon 規定：条件 [メタン無，水位高]，イベント [Pon]

B) Poff 規定：条件 [なし]，イベント [Poff]

A) により，メタン無で水位が高の状態で Pon による遷移を持つ 8，9 で，Pon による遷移以外の 4 つの入力遷移と 1 つの出力遷移を削除する（図 10 の青色の状態）．

B) のポンプ停止の条件は，メタン有またはメタン無かつ水位低であるが，安定状態条件による遷移の削除のあとではポンプ停止の場合はつねにこの条件が成立する（図 10 参照）ので，「条件なし」とした．B) より，Poff イベント遷移を持つ状態 2，3，14，15，18，19，22，23 で，Poff による遷移以外のすべての遷移（入力遷移も含む）を削除する．これによって，入力遷移 18，出力遷移 4 の計 22 の遷移を削除する（図 10 の緑色の状態）．

図 10 で削除した遷移は白抜き文字で示す．ここで，No.14，15 の状態への遷移が削除されており，その状態にくることはないので，それらの状態を削除できる．この結果，状態数が 22，遷移数が 51 の LTS が得られる．これは 4.3 節 [条件 1] の安定状態に対する条件①～⑤をすべて満たしている．この結果は Uchitel ら [4] の結果（**図 11**）と等しい LTS が得られた．図 10 の右に，各状態に対応する文献 [4] の状態番号を記した．なお，イベント名は，図 10 の app，disap，low，mid，high，Pon，Poff，Lon，Loff が図 11 の methAppears，methLeaves，lowWater，midWater，high-Water，switchOn，switchOff，switchDLOn，switchDLOff にそれぞれ対応している．また，図 11 で tick（タイマイベント）による遷移を持つ状態が安定状態になり，そこからは入力遷移しかないことが確かめられる．

6. 議論

(1) Uchitel ら [4] との比較

図 11 の LTS は，かなり詳細に見ないとその妥当性を理解することは難しい．我々が理解したやり方は，初期状態 0 から始めて，遷移のイベントを見ながら遷移先の状態の様子（ドメイン状態）をメモし，ドメイン状態の把握によってそこから出るべきイベントを理解するというやり方を繰り返した．たとえば，図 11 で，1 の状態は水位低でメタン発生，ポンプもランプもオフ状態であることが，初期状態 0 から methAppears による遷移先であることより分かる．これより，1 からの入力遷移は，（水位低だから）midWater か（メタン発生だから）methLeaves，必要な出力遷移は（メタン発生なので危険ランプを点灯する必要が

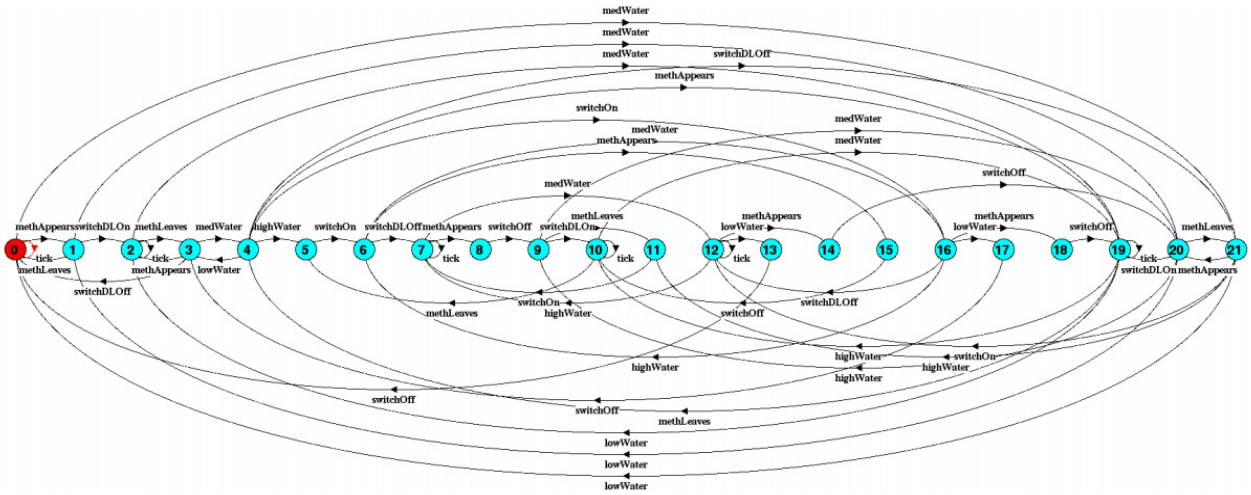


図 11 Uchitel らの結果（文献 [4] より）
Fig. 11 Finally synthesized LTS by Uchitel, et al, appeared in Ref. [4].

あるので）switchDLOn であることが理解できる．図 10 では，各状態のドメイン状態が明示されているため，そこからどんな遷移がありうるか，遷移によるドメイン状態の変化より遷移先状態がどうなるか等が比較的簡単に理解できる．

Uchitel らの方法では，制約として 13 個の FLTL を規定しているが，我々の方式に対応させると，このうち 5 つはドメインプロパティの規定，6 つが安定状態の規定，2 つが優先イベントの規定と考えられる．たとえば，ドメインプロパティの規定の例として，①ポンプオフのときは switchOff（我々の例では Poff）ができない，②ポンプオンのときは switchOn（我々の例では Pon）ができないという規定に対して，以下の FLTL が書かれる（FLTL についての詳細は文献 [4] を参照されたい）．

① □(¬PumpOn ⇒ X(¬switchOff W PumpOn))
② □(PumpOn ⇒ **X**(¬switchOn **W** ¬PumpOn))

これは，我々のドメインプロパティである図 8 (3) のポンプの規定を FLTL で表したものである．

また，安定状態の規定として，③メタン発生中は危険ランプがオンで，それでなければオフになること，④水位低でなくかつメタンがないならポンプオンであることの規定として，以下のような FLTL が書かれる．

③ □(tick ⇒ (MethanePresent ⇔ DangerLightOn))
④ □(tick ⇒ ((¬LowWater ∧ ¬MethanePresent) ⇒ PumpOn))

これは，③が 5(4) の安定状態の条件 R1 に，④が R2 の一部（R2 の「それ以外は停止」が抜けている）に対応している．

優先度の例として，⑤水位高でかつメタン無なら，直後はポンプオンでならなければならない（ポンプオンを優先），⑥水位低またはメタン有なら，直後はポンプオフでなければならない（ポンプオフを優先）規定の FLTL 記述を示す．

⑤ □((HighWater ∧ ¬MethanePresent) ⇒ **X** PumpOn)
⑥ □((LowWater ∨ MethanePresent) ⇒ **X**¬PumpOn)

これらは，⑤が我々の Pon イベント優先規定（5(5)A)），⑥が Poff イベント優先規定（5(5)B)）に対応している．

このように，文献 [4] のケーススタディの FLTL による制約記述は，我々のモデルでのドメインプロパティ，安定状態，優先イベント規定のいずれかになっていることが分かる．コントローラ生成における FLTL 記述がつねにこの 3 つに分類できるとはいえないが，この分類は分析者にとって制約を決める重要なガイダンスとなる．

Uchitel らのケーススタディでは，シナリオや FLTL による条件を記述し，それから MTL を生成し，さらに条件を加えて徐々に正しい解に近づけるやりかたを 6 回繰り返して，結果が得られたとしているが，これらの条件が十分かどうかの判断が非常に難しいと思われる．我々の方法では，ドメインプロパティの直積をもとに安定状態を定め，結果の状態表を見ながらさらなる制約として優先イベントを規定するので，もしケーススタディのように結果の状態数がそれほど多くない場合，これらの各段階で状態表を見て結果をチェックすることができ，必要な制約についても理解しやすくなる．また，制約の十分性についても，状態表のサイズが大きくなければ，状態表の残った遷移について個別に検討すれば漏れなくチェックすることができる．ただしコントローラの状態をドメイン状態の直積として扱うことができない場合は，逆に Uchitel らの方法が汎用的で優れているといえる（(5) 適用範囲についての考察参照）．

(2) 状態数が多い場合

コントローラ生成はモデル検査と従妹問題であるとされる [14]．つまり，モデル検査はモデルが条件を満たしているか否かをチェックするのに対し，コントローラ生成は条件を満たしたモデルを生成する．ただし，モデル検査はモ

デルができた後に行うが，コントローラ生成はモデルが未定の状態，つまりより上工程（要求分析等）で使われるため，抽象度がモデル検査よりも高くなることが多い．したがって，モデル検査で重要な状態爆発についても，そこまで至らないで適用できる問題が多くあると考えられる．それでも，ドメインの数が多くなってくると，その直積で得られるすべての可能な状態数は分析者の手におえないほど多くなることがある．我々の経験では，少し複雑な ATM の分析で，ドメイン数が 6 個（カード (4)，入出金口 (6)，テンキー (3)，機能選択ボタン (8)，キャンセルボタン (2)，銀行 (11) で () 内は状態数）で，状態の組合せ数は 12,672 になった．このうち，安定状態は 15，不安定状態は 98 になり，残りの 12,559 の状態は考慮しなくてもよかった．最初に安定状態を 15 個定め，次に安定状態からの入力イベント (25) のすべてにつき次の安定状態の候補をツールによって自動生成し [15]，候補が複数の場合は分析者が意味のあるものを選択し，なお複数個選択された場合は遷移のガード条件を規定するという手順で進めた．これにより，安定状態とそれから派生する不安定状態（安定状態から安定状態への遷移処理の途中に現れ，ツールによって自動的に生成される）だけに注目することができる．つまり，これ以外の状態には至らないことが，ドメインプロパティの規定と安定状態の条件より保証される．このように，組合せの状態数が多くなっても，提案手法では検討すべき状態は，安定状態（分析者が決定）とそこから派生する不安定状態（自動的に求めることができる）に限ることができるという効果がある．

(3) 不安定状態の扱い方

この論文では，不安定状態も状態としてあつかったが，UML や SDL の状態マシンにおいては，安定状態のみを状態とし，不安定状態は遷移での処理の中間点の状態を表し，状態マシンの状態とはしない方式になる．我々の方式で，「不安定状態は入力遷移を持たない」という条件を入れると，UML や SDL と同様の方式になり，それ以外の条件はそのまま利用できる．

SDL では，入力はイベントキューに入るので，処理途中の入力はキューに覚えて，処理後の次の状態に入ったところで見られることになる [11]．我々の方式では，たとえば図 10 で，状態 9 ではポンプ起動を優先するので，同時にメタンが発生（app イベント）した場合，まず 9 から 11 に移り，11 で app を受けて 23 に移り，そこでポンプを停止して 21 の安定状態に移ることになる．これを，優先を考えないと，9 で app を受けるとすぐに 21 に移ることになり，最終的な結果は同じになる．この場合，優先イベントによって，Pon の直後に Poff を行うという無駄な処理が起こってしまう．出力処理の速度が，センサ感知の速度に比べて十分に速い場合は，SDL のように不安定状態からの入力遷移をなくす方式（つまり不安定状態を状態として扱わない）でもよいが，そうでない場合は不安定状態での入力も含めた考察が必要となる．

(4) その他の要求規定について

ケーススタディでは安定状態と優先イベントですべての要求を規定できたが，一般にはこれだけで要求を完全に述べることができるわけではない．ここでの方式の基本的な前提としては，コントローラの振舞いはドメインの状態によって決まり，環境を制御するとは，ドメインの状態を適切なものに変えることととらえる．そのうえでシステムは決められた安定状態を持ち，安定状態以外の状態に対しコントローラは適切な安定状態に速やかに移るように制御することを基本とする．

このとき，不安定状態からの目標安定状態が 1 つだけある場合は，4 章にも示したように，多くの場合，どのドメインのイベントを優先すべきかを規定することによって適切な制御が得られる．目標安定状態が 2 個以上ある場合，まずはどの安定状態に向かうべきかを定める必要がある．一般には複数の目標安定状態を選択してもよいが，その場合は 1 つの状態から複数の目標安定状態に向かう出力遷移が規定される．1 つの状態から 2 つ以上の出力遷移がある場合，そのどれを選ぶかについての条件を別途規定する必要がある．たとえば電話交換機で，ダイアル中状態で数字を入力した場合，ダイアル中状態，呼び出し状態，ダイアル誤り状態等の目標安定状態があるが，条件としてはそれぞれ，ダイアル未完，ダイアル完了，ダイアルエラー等が規定される．これらは遷移のガード条件となるが，問題独自の記述になることが多く，ドメインプロパティ，安定状態，優先イベント規定には含まれない要求規定になる．

我々の方式の適用手順では，安定状態の規定によって得られたコントローラ LTS において，安定状態から 1 つの入力遷移で移った状態に対し，その目標安定状態を調べ，それが複数個ある場合は，①採用，②ありうるが不採用，③ありえない，の 3 つに分類し，①の採用の目標状態に至る遷移のみを残した LTS を作る．これは，具体的な遷移をチェックしながら行うため，検討すべき遷移数が少ない場合は，検討漏れを防ぐ方法として有効である．上の (2) で述べた ATM の例では，安定状態と入力イベントを定めたときの目標安定状態数は，25 の入力イベントにつき，1 個が 9，2 個が 9，3 個が 6，4 個が 1 となり，最大でも 4 個の目標安定状態につき選択すればよいため，このような後付けでの条件設定が実用的で便利に適用できた．しかし，遷移数がもっと多い場合は FLTL 記述のような包括的な規定が優れているといえる．

(5) 適用範囲についての考察

提案手法は 5 のケーススタディに対してはかなりうまく適用できた．これまでの我々の実験的な適用では，電話交換機（呼処理）[12]，水門制御 [6]，自動販売機，ATM，簡単な洗濯機制御，電子レンジ等で便利に適用できた [16]．

これらはいずれも，事前にドメインプロパティが明らかで，制御が状態指向の考え方に沿っている，つまり安定状態が重要であり，制御はつねに安定状態に復帰するように振る舞うという性質がある．

これらの前提が満たされない場合，たとえばドメインプロパティが明確ではない場合，特に個別のドメインの識別が不明確である場合は，提案方式はうまく適用できない．我々の手法ではドメインプロパティを明確にすることは，コントローラ生成問題の対象ではなく，その前提と考えるためである．また，状態指向に合わない場合，たとえば環境の状態とは関係なく制御する必要がある場合や，環境の状態のとらえ方が明確ではない場合も，提案手法は適さない．一方 Uchitel らの方法だと，これらの前提は必要ないため，より汎用的に適用できるといえよう．その他，離散的な状態遷移で問題が規定できない場合，たとえば飛行船の制御 [17] では PID 制御等微積分を使った制御が必要になり，我々の手法は適用できなかった（この場合 FLTL も適用できない）．

提案手法は他の方法と排他的なものではないので，他の方法と混在させて使うこともできる．たとえば，直接安定状態を定める代わりに，安定状態についての条件を FLTL（Fluent はドメイン状態）で記述し，それを満たす安定状態を生成することができる．またドメインプロパティを設定後にいくつかのシナリオを規定し，安定状態の検討（シーケンス図でコントローラへの入力の直前が安定状態に対応する）や必要な遷移の分析を行う方法も考えられる．

7. おわりに

環境の規定からコントローラを生成する問題に対して，プロブレムフレームと状態指向状態マシンの概念を適用し，安定状態に加え優先イベント規定によって，解となるコントローラを得る手法を述べた．プロブレムフレームの概念によって，環境の規定は，所与であるドメインプロパティと願望である要求に分けて扱われる．たとえば鉱山ポンプの問題では，水位センサは低から高への遷移はない（必ず中を経る）というのは所与であり，メタンガスが発生すればポンプを停止するというのは願望である．状態指向の概念により，まずドメインの直積（所与となる）を環境が許す可能なすべての処理（たとえば「水位が低でもポンプを起動する」も可能）として求め，それに安定状態や優先イベントによって可能な処理から必要な処理，不要な処理を求め，可能な処理で必要か不要か判断できない処理を減らしていく．ケーススタディで，この方式により求める解が適切に得られたことを示した．また，求めるコントローラ LTS を状態遷移表で表すことにより，漏れのない検討が可能であった．

今後の課題として，まずは提案手法をいろいろな実際の問題で評価することが必要である．ケーススタディの問題では，ドメイン数が 4 つ，直積 LTS の状態が 24 と比較的少ないため，見通しがつきやすい．また，不安定状態の目標安定状態がつねに 1 つであり，コントローラの振舞いが決めやすい．目標安定状態が多く存在し，その選択や条件付けが複雑な場合は提案手法だけでは求めるコントローラを生成できない．この場合，さらなる要求をどのように定めるかが問題となる．それでも，安全状態と優先イベント規定によってできる未完成状態のコントローラは，分析者にとってさらに要求を絞り込むうえでの重要なガイドを与えることができる．6 章 (5) にも述べたが，より一般化して，FLTL も併用する方式も考えられる．さらに，組込みシステムの特徴をとらえて，限定的ではあるが使いやすい要求規定（FLTL のパターン化 [5] 等）を定めることも今後の課題となる．さらに，支援ツールの開発も今後の課題である．我々は提案方式で不安定状態や優先イベントを含まない場合については支援ツールを作成した [16] が，不安定状態も入れると状態数が何倍にも増えるため，多くの状態を適切に扱えるツールが必要になろう．

参考文献

[1] Whittle, J. and Schumann, J.: Generating Statechart Designs From Scenarios, *22nd International Conference on Software Engineering* (2000).

[2] Sibay, G.M., Braberman, V., Uchitel, S. and Kramer, J.: Synthesizing Modal Transition Systems from Triggered Scenarios, *IEEE Trans. SE*, Vol.39, No.7, pp.975–1001 (2013).

[3] Letier, E. and Heaven, W.: Requirements Modelling by Synthesis of Deontic Input-Output Automata, *ICSE 2013*, pp.592–601 (2013).

[4] Uchitel, S., Brunet, G. and Chechik, M.: Synthesis of Partial Behavior Models from Properties and Scenarios, *IEEE Trans. SE*, Vol.35, No.3, pp.384–406 (2009).

[5] 金井勇人，岸　知二：ソフトウェア設計に対するモデル検査のための検証パターン，情報処理学会論文誌，Vol.49, No.10, pp.3493–3507 (2008).

[6] Jackson, M.: *Problem Frames: Analyzing and Structuring Software Development Problems*, Addison-Wesley Publishing Company (2001).

[7] 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).

[8] 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).

[9] 紫合　治：ジャクソン法（JSP）による状態遷移設計，情報処理学会論文誌，Vol.50, No.12, pp.3041–3051 (2009).

[10] OMG: UML 2.0 Superstructure Specification (2004), available from ⟨http://www.omg.org/⟩.

[11] 若原　恭，長谷川晴朗：仕様記述言語 SDL，株式会社カットシステム (1996).

[12] Rockstrom, A. and Saracco, R.: SDL-CCITT Specification and Description Language, *IEEE Trans. Communications*, Vol.COM-30, No.6, pp.1310–1318 (1982).

[13] Giannakopoulou, D. and Magee, J.: Fluent model checking for event-based systems, *ESEC/FSE 2003* (2003).

[14] Kuijper, W. and Pol, J.: Compositional Control Synthesis for Partially Observable Systems, *CONCUR'09* (2009).
[15] 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 論文集 (2011).
[16] 紫合 治，横山 薫：プロブレムフレームに基づく組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 (2012).
[17] ESS ロボットチャレンジ 2012，入手先 ⟨http://sigemb.jp/rc/2012/⟩ (2012).

紫合 治 （正会員）

1971 年東京電機大学工学部電子工学科卒業．同年日本電気（株）入社．ソフトウェア生産技術研究所，NEC アメリカ等にて，ソフトウェア工学，インターネットセキュリティ等の研究開発に従事．2003 年より東京電機大学情報環境学部教授，現在に至る．博士（工学）．情報処理学会 20 周年記念論文賞（1980 年），大河内記念技術賞（1984 年），情報処理学会山下記念研究賞（2010 年）受賞．ソフトウェア科学会会員．