情報処理学会論文誌 Vol. 52 No. 12 3205–3220 (Dec. 2011)

ゴール指向分析に基づくモデル検査のための外部環境の抽象化手法

乾 道孝[†1] 吉岡信和[†2] 落水浩一郎[†1]

本稿では，時系列の変化量に対して制約を設ける，外部環境モデルの抽象化手法を提案する．センサ・アクチュエータの組み込みシステムにおいて，複数の外部環境の情報を内部に保持するシステムの場合，静的な観点で外部環境の情報の範囲を削減する抽象化方法には限界がある．また，動的な観点で外部環境の情報の変化に制約を設ける抽象化方法は，制約の分析に必要なセンサとアクチュエータの振舞いを，検査記述から必要十分に抽出するのは困難である．さらに，センシング対象である水や色や音など，外部環境に関する必要十分な情報が検査記述にないため，その変化の分析が困難である．本稿では，この動的な観点での抽象化の 2 つの問題に対して，制約の分析に必要なセンサとアクチュエータの振舞いをゴール指向分析[10] を用いて網羅的に抽出し，外部環境の情報に対する時系列の変化量をドメイン知識を用いて分析する，外部環境の情報の変化に制約を設ける抽象化手法を提案する．さらに，事例をもとにその有効性を示す．

An External Environment Abstraction Method for Model Checking with a Goal-Oriented Analysis

Michitaka Inui,[†1] Nobukazu Yoshioka[†2] and Koichiro Ochimizu[†1]

In this paper, we propose an abstraction method of the external environment model with a limit on the time series variation. This paper is intended for embedded systems with sensors and actuators. When the system uses multiple external environment information, the abstraction method reducing the scope of the information from the static viewpoint might not work well. Also, the abstraction method giving a limit on the time series variation from the dynamic viewpoint is difficult from the description for model checking. Because it is hard to extract necessary sensors and actuators behaviors required to analyze for the limit. In addtion to, it is difficult to analyze the time series variation, because of the lack of necessary information in the description for model checking, The external environment information is water, color, sound, and etc. In this paper, to solve such problems with the dynamic viewpoint of this abstraction, we propose the abstraction method giving a limit on the time series variation, using a Goal-Oriented Analysis for extracting the behavior comprehensively, using domain knowledge for analyzing the time series variation. Also, we show validity of our proposal method with case studys.

1. はじめに

近年，モデル検査技術は，ソフトウェアシステムの信頼性を保証する有効な手段として産業界で注目を集めている．特に，自然現象（外部環境）の情報を得て判断動作するセンサ・アクチュエータシステムは，得られる外部環境の情報が膨大であり，レビューによる設計検証が困難である．そのため，モデル検査による設計検証が試みられている．

モデル検査は，モデル検査器へのインプットとなる，システムを検査するための記述と，そのシステムの入力となる外部環境の情報を生成させる記述が必要である．特にセンサ・アクチュエータシステムにおける外部環境は，次の理由で非常に多くの状態を持つことになる．すなわち，システムがセンサから得られる環境が連続的な値をとり，かつアクチュエータの動きや物理現象に合わせてその値が刻一刻と変化するためである．そのため，状態爆発の要因となりやすく抽象化が必要である．一般に，その抽象化は，静的な観点で外部環境の情報の範囲を削減する方法と，動的な観点で外部環境の情報の変化に制約を設ける方法が考えられる．

しかし，外部環境の情報を格納する複数の変数が相互に依存関係を持っている場合，外部環境の情報の範囲を削減する抽象化手法は，複数の変数の関係を考慮する必要があり抽象化に限界がある（複数の変数の組合せ問題）．また，外部環境の情報の変化に制約を設ける抽象化は，外部環境の情報の時系列における変化の分析を必要とするが，分析に必要なセンサとアクチュエータの振舞いを，検査記述から必要十分に抽出するのは困難である．加えて，センシング対象である水や色や音など外部環境に関する必要十分な情報が，検査記述にないため分析が困難である．

そこで本稿では，外部環境の情報の変化を分析するため，ゴール指向分析[10] を用いて，

†1 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology
†2 国立情報学研究所アーキテクチャ科学研究系
Information Systems Architecture Reserch Division, National Institute of Informatics

検査に必要なセンサとアクチュエータの振舞いを網羅的に抽出する．さらに，ドメイン知識を用いて，外部環境の情報の変化量（変化値の最大と最小）を分析し，外部環境の情報を生成させる記述の振舞いに制約を設ける抽象化手法を提案する．本提案手法を用いることで，複数の変数の組合せ問題になる場合でも，さらなる抽象化が可能となり，状態爆発の発生確率を従来より軽減できる．

まず，本稿で対象とするシステムの特徴を 2 章で説明する．次に，外部環境の情報を生成させる記述の抽象化における問題点を 3 章で述べる．その問題点を解決する手法を 4 章で提案し，5 章でその評価を行う．そして，6 章で現存の関連手法と本提案手法の違いについて述べ，最後の 7 章でまとめと今後の課題を述べる．また，モデル検査器には，SMV [11]，LTSA [12]，SPIN [9] など様々なモデル検査器が存在するが，本提案手法はモデル検査器に依存しない．そのため，本稿での事例では SPIN を用いる．

2. 対象とするシステム

まず，本稿で対象とするシステムの特徴を 2.1 節で説明する．次に，本稿で扱う事例システムを 2.2 節で述べる．

2.1 対象とする検査記述の特徴

本稿で対象とするシステムは，**図 1** に示すとおり，外部環境の情報をシステムが取得し，内部に状態を持って判断，そしてアクチュエータを用いて外部環境を操作する，センサ・アクチュエータベースのシステムである．ここで，外部環境の情報とは，「外部環境の要素」の現象をセンサで取得した情報である．この「外部環境の要素」は，水や音や色などのセンシングする対象である．また，システムに対してモデル検査を行ううえで，システムの振舞いの記述をシステム記述，そのシステムの入力となる外部環境の情報を生成させる記述を外部環境記述といい，その 2 つの記述をあわせて検査記述とする．この検査記述は，システムの機能を実現させる詳細な仕様，つまり設計仕様を入力として記述される．

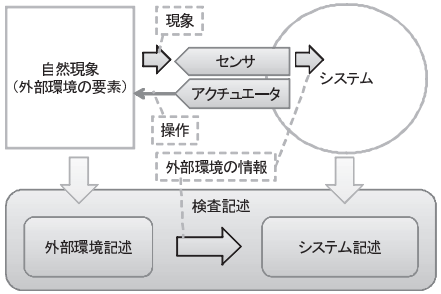

図 1　検査記述の概要
Fig. 1　Discription overview for model checking.

本稿で対象とするシステム記述は，自然現象などの外部環境の情報の入力を必要とするため，外部環境記述は一般的に非常に多くの状態が存在する．この外部環境記述の抽象化を行う手法として，静的な観点で外部環境の情報の範囲を削減する方法と，動的な観点で外部環境の情報の変化に制約を設ける方法が考えられる．ここでの外部環境の情報の変化への制約とは，外部環境記述が生成する情報に対する，時系列の変化量の制限のことである．たとえば，外部環境記述が 1〜10 までの範囲の値（外部環境の情報）をシステム記述に入力する外部環境記述があり，入力値が 1 つ前の入力値より 5 以上大きくはないとの制限を設けたとする．その場合，外部環境記述からの 1 つ前の入力値が 1 であれば，次は 1〜5 の 5 通りとした抽象化である．

2.2 事例システム

本提案手法を説明するための事例として，ET ロボコン[8] で用いられる「ライントレーサロボット」（ライントレーサ）を例にあげる．ライントレーサを**図 2** に記載する．また，以下に基本仕様を記載する．

- 左右それぞれのモータ出力でタイヤを制御する．

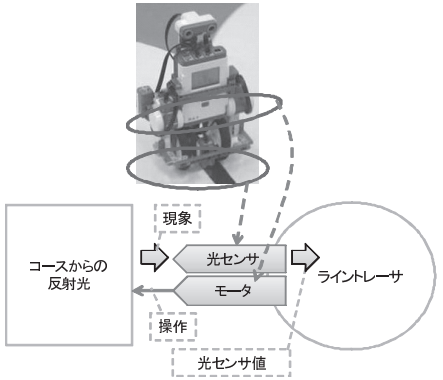

図 2　ライントレーサ
Fig. 2　Line tracer.

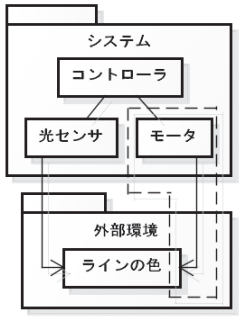

図 3　ライントレーサの設計の概略図
Fig. 3　Design overview of line tracer.

- 白地に黒色のラインの色を光センサを用いて認識し，タイヤ制御によりラインをトレースする．
- 外乱光は考慮しない．
- ジャイロセンサを用いて倒立振子制御により 2 輪倒立を行う．

このソフトウェアはラインをトレースするために，光センサを用いて光の反射量を観測し，反射量によって光センサ下にある色を認識する．色の認識によりライントレーサは，駆動モータを制御し，ラインをトレースする走行を行う．また，ラインをスムーズにトレースする一般的な技術である PID 制御[15] を用いる．この機能の設計仕様（2.1 節を参照）は次のとおりである．また，その設計の概略図を**図 3** に示す．

- PID 制御は，ライントレーサの車体がラインに沿う旋回と前進を行うことで実現される．
- 旋回および前進はモータ制御によって行われる．
- ラインに沿う旋回は，光センサによるラインの色に相当する光センサ値を取得し，旋回の向きと力を計算，そしてその計算結果を基にモータ制御による旋回を行う．
- 光センサから値を取得する周期は 1 ms である．
- その計算は，P 成分，I 成分，D 成分の総和であり，ライントレーサのコントロール部が行う．
- P = (今回の光センサ値 − 前回の光センサ値) × 係数（36）
- I = (今回の光センサ値 − 閾値（575)) × 係数（1）
- D = ((今回の光センサ値−前回の光センサ値)−(前回の光センサ値−前々回の光センサ値)) × 係数（4）

ここで，この PID 制御に関してモデル検査を行う場合を想定する．PID 制御はフィードバック制御を行う技術である．そのため検証項目は，「直線をトレースするとき，ライントレーサが旋回を行う力はいつかなくなる」や「デッドロックはつねに発生しない」などが考えられる．このシステムにおいて，システム記述はライントレーサ自体の振舞いとなり，外部環境記述から光センサ値を入力する．また，ラインの色の変化のモデル化は，様々な物理現象を考慮する必要があるため困難である．そこで，モデル検査器の網羅的な探索機能を用いて，全変化パターンの構築を行う．そのため，この PID 機能のシステム記述と外部環境記述（ソースコード 1 [⋆1]）は，図 3 の点線部分の記述を行っていない．

ソースコード 1　etrb_pid.pml

```
 1  chan light = [0] of { int }
 2  /* 外部環境記述 */
 3  active proctype external()
 4  { int value = 500;
 5    /* ライントレーサが取得する光センサ値 */
 6    /* (500〜700)の全組合せパターンを */
 7    /* 構築する */
 8    do::do::if::value++;
 9                ::value−−;
10              fi;
11              if::(value==499)−>value++;
12                ::(value==701)−>value−−;
13                ::else−>skip;
14              fi;
15            ::break;
16        od;
17      /* システムモデルへ送信 */
18      light!value;
19    od;
20  }
21  /* システム記述 */
22  active proctype system()
23  { /* 今回取得した光センサ値変数 */
24    int now = 0;
25    /* 設計した白黒の閾値 */
26    int th = 575;
27    /* 旋回値変数 */
```

⋆1 Promela の記法
if 式　::後の条件式を満たすものをランダムに 1 つだけ実行する．
do 式　繰返し if 式と同じ処理を行う．ただし，break 式で抜けられる．
active proctype　自動的に生成されるプロセスの宣言．
chan　通信用チャネル宣言．「!」は送信，「?」は受信．

```
28    int turn = 0;
29    /* 前回取得した光センサ値変数 */
30    int pre = 0;
31    /* 前々回取得した光センサ値変数 */
32    int prepre = 0;
33    /* P 成分変数:ラインとライントレーサの進行 */
34    /* 方向の角度 (光センサ値の今回と前回の */
35    /* 取得値の差を角度) */
36    int p_value = 0;
37    /* I 成分変数:ラインとライントレーサが離れ */
38    /* た距離 (光センサ値の前々回・前回・今回の */
39    /* それぞれの値と,閾値との差の積分)*/
40    int i_value = 0;
41    /* D 成分変数:ラインとライントレーサとの */
42    /* 角速度 (光センサ値の,前々回・前回と */
43    /* 前回・今回の角度の差) */
44    int d_value = 0;
45    /* 外部環境モデルから受信 */
46    do::light?now;
47      d_step{
48      /* P 成分計算(36はP 成分係数) */
49      p_value = (now−pre)∗36;
50      /* I 成分計算(1はI 成分係数) */
51      i_value = (now−th)∗1;
52      /* D 成分計算(4はD 成分係数) */
53      d_value = ((now−pre)−(pre−prepre))∗4;
54      /* 旋回値決定 */
55      turn = p_value + i_value + d_value;
56      /* 前々回取得した光センサ値を更新 */
57      prepre = pre;
58      /* 前回取得した光センサ値を更新 */
59      pre = now;
60      }
61    od;
62  }
```

3. 外部環境記述の抽象化での問題点

本稿で対象とするシステムにおいて，複数の変数の組合せ問題の場合，静的な観点での抽象化では限界があることを 3.1 節で説明する．また，動的な観点での抽象化は，設計仕様と検査記述（2.1 節を参照）からでは難しいことを 3.2 節で述べる．

3.1 静的な観点での抽象化の限界

設計仕様に含まれる外部環境記述の静的な情報は，一般的に検査記述に反映される．そのため，検査記述の抽象化方法に着目する．静的な観点で外部環境の情報の範囲を削減する方法において，データマッピング法による抽象化[4),7)] とプログラムスライシング[21)] の技術に類似した抽象化[4)] が，比較的に適用が容易なため一般的に用いられている．そのうち，複数の変数の組合せ問題の場合，つまり外部環境の情報をシステム上で保存しておくシステム記述の変数が複数あり，システム記述の振舞い結果にそれぞれが関連を持っている場合，データマッピング法[4),7)] が適用できない．データマッピング法は，変数に格納されるデータで処理結果が同じデータを，ひとまとまりにして扱う（離散化する）抽象化である．そのため，変数ごとにしか適用できない．

ライントレーサの事例において，ソースコード 1 の検査記述に対しデッドロックの検査を行うと，states は 4065259，transitions は 7838824 であるが，out of memory との表記がでて状態爆発が発生した．この記述へのプログラムスライシングの技術に類似した抽象化の適用は，削減できるコードがないためこれ以上難しい．そのため，状態数が多いと考えられる外部環境記述をデータマッピング法を用いた抽象化の適用を試みる．この PID 制御では，PID 成分それぞれの計算（47〜60 行目）には今回（now）前回（pre）前々回（prepre）の外部環境の情報が用いられ，旋回値（55 行目）を算出している．そのため，外部環境記述から入力される情報 now，pre，prepre を考慮して，データマッピング法を用いて，入力される光センサ値の情報をひとまとまりにできるかを検討する．もし，白（now < th（26 行目：閾値））のときは右旋回，黒（now ≧ th）のときは左旋回とした On/Off 制御であれば，白と黒の 2 値とひとまとまりにして抽象化が可能である．しかし，これら now，pre，prepre の変数は相互に依存関係を持っているため，白と黒の 2 値だけでは PID 制御が行えない．そのため，光センサ値の情報をひとまとまりにすることが困難である．

このことから，静的な観点で外部環境の情報の範囲を削減する方法には限界がある．

3.2 動的な観点での抽象化の問題点

外部環境の情報の変化に制約を設ける抽象化には，外部環境の情報がどのように変化するかを，次の項目を考慮して判断する必要がある．

- 「システムの動作に必要な外部環境の情報」
- 「外部環境の要素」
- 外部環境の要素に影響を及ぼす「アクチュエータの操作」
- 外部環境の要素に影響を及ぼす「外部環境の別の要素」
- センサやアクチュエータを制御する「システムの性能」

上記 5 つの項目を考慮する理由は次のとおりである．外部環境の情報の変化に制約を設ける抽象化は，「システムの動作に必要な外部環境の情報」に着目し，その情報の時系列の

変化を分析することで，制約条件の導出および抽象化を行う．ここで，「システムの動作に必要な外部環境の情報」とは外部環境の情報の中で，システムが動作するために扱う情報である．また，外部環境の情報は「外部環境の要素」の現象をセンサで取得した情報である．さらに，「外部環境の要素」は「アクチュエータの操作」と「外部環境の別の要素」が影響する[6]．この「アクチュエータの操作」とはアクチュエータを用いたシステムの制御であり，「外部環境の別の要素」とは「外部環境の要素」に影響を与える別の外部環境の要素である．そのため，「システムの動作に必要な外部環境の情報」の時系列の変化の分析は，「外部環境の要素」と「外部環境の別の要素」そして「アクチュエータの操作」が必要である．また，センサやアクチュエータの制御タイミングなどの「システムの性能」は，「システムの動作に必要な外部環境の情報」の時系列の変化に影響する．これらより，外部環境の情報の変化に制約を設ける抽象化は，上記 5 つの項目から分析する必要がある．

しかしながら，次の 2 つの問題がある．

問題 1　検査のための「システムの動作に必要な外部環境の情報」と「アクチュエータの操作」を必要十分に設計仕様と検査記述（2.1 節を参照）から整理して確認することは難しい．

問題 2　ドメイン知識から得られる「外部環境の要素」と「外部環境の別の要素」の情報が設計仕様と検査記述には含まれないため，本節記載の 5 つの項目の分析が困難である．

設計仕様（2.2 節を参照）とソースコード 1 の検査記述に対して，外部環境の情報の変化に制約を設ける抽象化を試みる．しかし，検査に必要な「システムの動作に必要な外部環境の情報」である PID 制御に必要な光センサ値と，「アクチュエータの操作」である PID 制御に必要なモータ制御を必要十分に整理して確認することは，設計仕様とソースコード 1 からでは難しい（問題 1）．また，「外部環境の要素」となるコースの色や，「外部環境の別の要素」となるコースの色に影響を及ぼす要素が，設計仕様とソースコード 1 上に情報が載っていない（問題 2）．そのため，PID 制御に必要な光センサ値の，時系列の変化の分析が困難である．

4. 提 案 手 法

本稿では，ゴール指向分析[10] とドメイン知識を用いて外部環境の情報の変化を分析し，外部環境の情報の変化に制約を設ける抽象化手法を提案する．まず，提案手法の特徴を 4.1 節で述べ，その後に具体的なアルゴリズムの説明を 4.2 節で行う．

4.1　提案手法の特徴

外部環境の情報の変化に制約を設ける抽象化は，3.2 節で述べた 2 つの問題が存在する．そこで本節ではそれぞれの問題解決に対する，本提案手法の取組みと手順概要を述べる．

まず問題 1 に対して述べる．設計仕様（2.1 節を参照）には，複数の「システムの動作に必要な外部環境の情報」と「アクチュエータの操作」を扱う設計仕様が含まれている．そのため，検査に必要十分なこれらの情報を把握するため，関係する設計仕様を網羅的に抽出する必要がある．そこで，ゴール指向分析を用いる．検査するシステムの機能の実現状態をトップゴールとし，設計仕様を用いて，センサ・アクチュエータを扱う具体的な設計仕様が現れるまでゴール分解を行うことで，検査するシステムの機能が制御する「システムの動作に必要な外部環境の情報」と「アクチュエータの操作」を段階的かつ網羅的に抽出できる．

次に問題 2 に対して述べる．検査記述に記載されていない 3.2 節であげた項目「外部環境の要素」「外部環境の別の要素」「システムの性能」を列挙するために，ドメイン知識を用いる．これらの取り組みにより，3.2 節で述べた 5 つの項目を列挙する．さらに，「システムの動作に必要な外部環境の情報」が時系列に変化する最大もしくは最小変化量を分析することで，外部環境の情報の変化に制約を設け抽象化を行う．本提案手法は次の 3 つの手順で行う．

まず手順 1 では，検査に必要十分な「システムの動作に必要な外部環境の情報」と「アクチュエータの操作」を把握するため，設計仕様から関連する複数の仕様をゴール指向分析を用いて抽出する．作成したゴール木をシステムモデルとする．次に手順 2 では，前述で把握した複数の設計仕様をまとめることで，検査に必要十分な「システムの動作に必要な外部環境の情報」と「アクチュエータの操作」を抽出する．そして最後に手順 3 では，ドメイン知識や要求仕様から「外部環境の要素」「外部環境の別の要素」「システムの性能」を列挙する．そして，手順 2 で抽出した 2 項目と合わせた 5 項目を用いて，「システムの動作に必要な外部環境の情報」が時系列に変化する最大もしくは最小変化量を分析することで，外部環境の情報の変化に制約を設け抽象化を行う．これら手順 1〜3 の具体的なアルゴリズムを，4.2 節で示す．

4.2　提案アルゴリズム

本提案手法は，次の 3 つの手順で外部環境の情報の変化の分析を行い，外部環境記述の抽象化を行う．その手順のアルゴリズムを 2.2 節のライントレーサの PID 制御での事例を用いて説明する．

手順 1：システムモデルの構築

「システムの動作に必要な外部環境の情報」と「アクチュエータの操作」を抽出するため，設計仕様（2.1 節を参照）を用いて，システム記述に対するゴール指向分析の適用により，

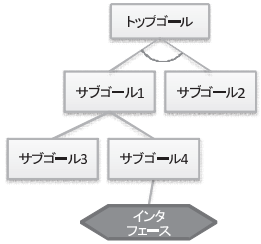

図 4　システムモデルの記法
Fig. 4　Notation of system model.

システムモデル（**図 4**）を構築する．

まず図 4 において，長方形に記載される文字列は機能の設計仕様であり，ゴール指向分析のゴールとする．また，一番上のゴールをトップゴールとする．トップゴールを実現するために必要なゴールをサブゴールとし，図 4 のサブゴール 1 とサブゴール 2 のように記載する．サブゴールの数に特に制限はない．また，ゴールを実現するために，サブゴール 1 とサブゴール 2 の両方が満たされている必要がある AND 表現を，トップゴールとサブゴール 1 とサブゴール 2 の関係のように表記する．また，ゴール 1 を実現するために，サブゴール 3 とサブゴール 4 のどちらかが満たされていればよい OR 表現を，サブゴール 1 とサブゴール 3 とサブゴール 4 の関係のように表記する．これら AND，OR 表記はゴール間で適用可能である．次に，センサ・アクチュエータのハードウェア（入出力インタフェース）を用いる設計仕様が現れた際，六角形のインタフェースで関連付けを明記する．

ライントレーサの事例では，まず**図 5** に示すように，「PID 制御によるライントレースが行えている」状態をトップゴールとし，光センサとモータのインタフェースを具体的に使用する仕様が現れるまで，同様のゴール分解を繰り返す．トップゴールを実現するためには，「ラインに沿える車体旋回ができている」設計仕様と「車体が前進できている」設計仕様の両方が満たされている必要があるため，AND 表記で表現し分解する．そして，インタフェースを具体的に使用した「車体が前進できている」「車体が旋回できている」「前回の光センサ値を保持している」「今回の光センサ値を保持している」「前々回の光センサ値を保持している」には，それぞれモータと光センサのインタフェースを接続する．

ここで，構築したシステムモデルには AND 表記のみ使用されているため，OR 表記を用いる事例を述べる．ライントレーサはコースの状況によって，PID 制御と On/Off 制御（3.1 を参照）を使い分けることがある．そのため，ライントレース機能に対するシステム

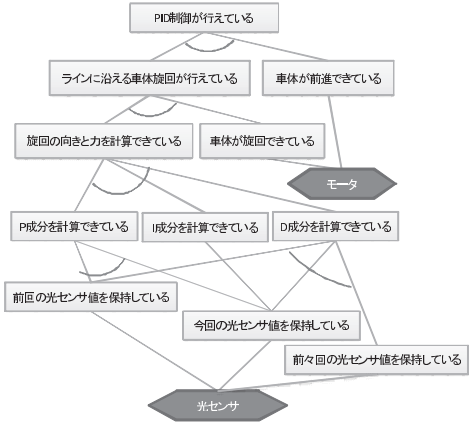

図 5　システムモデル
Fig. 5　System model.

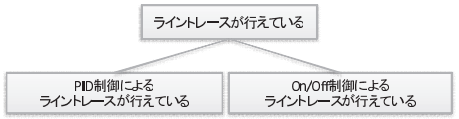

図 6　OR 表記の事例
Fig. 6　Example of OR notation.

モデルの構築を想定した場合，分解の過程において**図 6** のような OR 表記を用いる．

手順 2：「システムの動作に必要な外部環境の情報」と「アクチュエータの操作」の抽出

「システムの動作に必要な外部環境の情報」と「アクチュエータの操作」を，手順 1 の結果からボトムアップにより抽出する．ボトムアップにて行う理由は，手順 1 で網羅的に抽出した入出力インタフェースを用いた複数の設計仕様から，インタフェースごとに，システムの操作を整理するためである．ボトムアップでの抽出手法として，**図 7** の 3 階層の木構造を用いて行う．

まず，1 階層目は，システム機能が達成されているという抽象的な設計仕様とする．3 階層目は，手順 1 で抽出したインタフェースを使用する設計仕様とする．次に，2 階層目は，1 階層目の設計仕様を実現するためにインタフェースをどのように扱っているかの観点にて，3 階層目の設計仕様をボトムアップしひとまとまりにした設計仕様を抽出する．この 2 階層

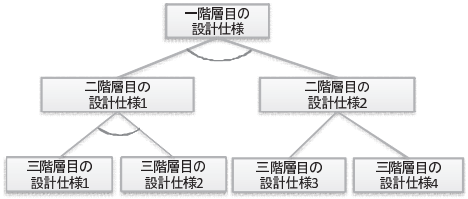

図 7　3 階層の木構造の記法
Fig. 7　Three hierarchical tree structure notation.

目の設計仕様は，インタフェースごとに存在する．このとき，2 階層目の設計仕様のインタフェースがセンサである場合は「システムの動作に必要な外部環境の情報」となる．また，インタフェースがアクチュエータである場合は「アクチュエータの操作」とし，外部環境の情報の時系列変化が最大となる状況を分析するために，アクチュエータの操作がどのように「外部環境の要素」に影響するのかという観点を用いてひとまとまりにする．

さらに，1〜3 それぞれの階層の設計仕様間の AND OR 表記を，システムモデルの結果を用いて機械的に関連付ける．第 1 と第 2 階層の関係は，対象であるシステムがセンサとアクチュエータを用いたシステムであるため，AND の関係となる．そして，システムモデルの設計仕様でもある「システムの動作に必要な外部環境の情報」の設計仕様の間の AND OR 関係を，システムモデルから抽出する．具体的には，システムモデルのトップゴールから，それらの設計仕様にいたるまでのゴール分解結果の間に OR 関係が 1 以上含まれていれば OR 表記を，そうでなければ AND 表記とする．

ライントレーサの事例では，図 5 で構築したシステムモデルから抽出する．まず，1 階層目の設計仕様は，検査する機能が PID 制御であるため，「PID 制御によるライントレースが行えている」とする．次に，3 階層目の設計仕様を，図 5 のインタフェースに接続されているサブゴール「車体が前進できている」「車体が旋回できている」「前回の光センサ値を保持している」「今回の光センサ値を保持している」「前々回の光センサ値を保持している」と抽出できる．次に，光センサとモータごとに 2 階層目の設計仕様を作成する．光センサの場合，「PID 制御によるライントレースが行えている」ために光センサ値をどのように扱っているかの観点で，「前回の光センサ値を保持している」「今回の光センサ値を保持している」「前々回の光センサ値を保持している」をひとまとまりにし，「光センサを用いて連続した過去 3 回分の値を保持している」という設計仕様を抽出できる．また，モータの場合，「PID 制御によるライントレースが行えている」ためにモータをどのように扱っているかの観点

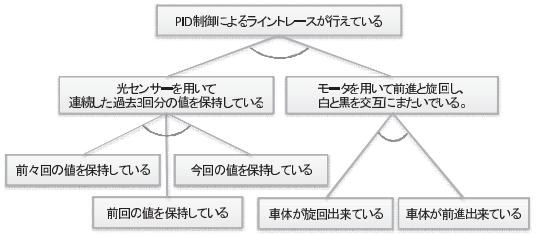

図 8　3 階層の木構造の事例
Fig. 8　The case of three hierarchical tree structure.

と，光センサ値の時系列変化が最大となる状況を分析するため，「外部環境の要素」である「コースの色」への影響を考慮して，「車体が前進できている」「車体が旋回できている」をひとまとまりにし，「モータを用いて前進と旋回し，白と黒を交互にまたいでいる」という設計仕様を抽出できる．さらに 1〜3 階層それぞれの設計仕様の関係を**図 8** のように AND 関係にした．

手順 3：外部環境の情報の変化の制約

3 つの項目「外部環境の要素」「外部環境の別の要素」「システムの性能」（3.2 節）をドメイン知識を用いて列挙し，手順 2 の 2 つの項目を加えて 5 つの項目を列挙する．**図 9** において，まず「外部環境の要素」の性質を，外部環境の情報（2.1 節）に影響する観点で列挙する．次に「外部環境の別の要素」の性質を，「外部環境の要素」に影響する観点で列挙する．次に「システムの性能」については，外部環境の情報に影響を及ぼす観点で列挙する．そして，手順 2 で抽出した「システムの動作に必要な外部環境の情報」「アクチュエータの操作」を列挙することで，5 つの項目を列挙する．

ライントレーサの事例では，まず「外部環境の要素」であるコースの色の性質は，ドメイン知識より，検査に必要十分であるコースの色の最大範囲（黒から白）の「光センサ値 500〜700」であるとした．その理由として，光センサ値はコース状況によって閾値（575）近辺での微小変動を行うこともあれば，大きく変動することもあるためである．次に「外部環境の別の要素」は，2.2 節で説明した要求仕様の「外乱光を考慮しない」ことから，「コースの色に影響を及ぼす性質はない」とした．次に「システムの性能」は，外部環境の情報である光センサ値に影響を及ぼす観点で「前進速度が 3（cm/s）」「センシング周期が 1（ms）」とした．そして，手順 2 のライントレーサの事例で列挙した「光センサを用いて連続した過去 3 回分の値を保持している」「モータを用いて前進と旋回し，白と黒を交互にまたいでい

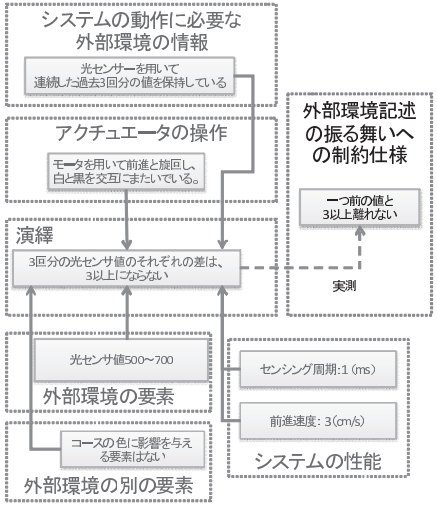

図 9　外部環境の情報の変化に対する制約の仕様抽出
Fig. 9　The line tracer case of giving a limit.

る」と合わせて，5 つの項目を列挙する．

列挙した 5 つの項目を用いて，「システムの動作に必要な外部環境の情報」の時系列の変化量を分析する．「システムの性能」「外部環境の要素」「外部環境の別の要素」を考慮し，「システムの動作に必要な外部環境の情報」が時系列の変化が最大もしくは最小となる「アクチュエータの操作」を検討する．そして，その状況での「システムの動作に必要な外部環境の情報」の時系列における，最大もしくは最小変化量をドメイン知識を用いて導きだす．

ライントレーサの事例では，まず，図 9 のシステムの性能が「前進速度が 3（cm/s)」「センシング周期が 1（ms)」，コースの色は「光センサ値 500〜700」を考慮し，「光センサを用いて連続した過去 3 回分の値を保持している」が時系列に最大に変化する「モータを用いて前進と旋回し，白と黒を交互にまたいでいる」状況をドメイン知識を用いて検討する．光センサ値の最小変化値は 0 であり，ラインに対して直角に前進する状況が，光センサ値の変化が一番大きいと検討した．その際の最大変化値を実測したところ，「3 回分の光センサ値のそれぞれの差は，3 以上にならない」との条件を得た．その制約でライントレースを行い実測し，その制約の妥当性確認を行った．この制約を外部環境記述に設けた結果をソースコード 2 に示す．

ソースコード 2　etrb_pid_m.pml

```
1  chan light = [0] of { int }
2  /* 外部環境記述 */
3  active proctype external() /* ← 外部環境モデルのプロセス */
4  { int value = 500;
5    int pre = 500;
6    /* ライントレーサが取得する光センサ値 */
7    /* (500〜700)の全組合せパターンを */
8    /* 構築する. */
9    /* 今回の送信値が前回の送信値より */
10   /* 3以上−3以下の時値を変更させない */
11   do::do::if::((value−pre)<=−3)||(3<=(value−pre))−>skip;
12               ::else −>
13                 if::value++;
14                   ::value−−;
15                 fi;
16           fi;
17           if::(value==499)−>value++;
18             ::(value==701)−>value−−;
19             ::else−>skip;
20           fi;
21         ::break;
22       od;
23     light!value; /* ← システムモデルへ送信 */
24     pre = value; /* ← 今回の送信値を前回の送信値として更新 */
25   od;
26 }
```

この検査記述に対しデッドロックの検査を行ってみると，本提案手法の適用前は，states は 4065259，transitions は 7838824 で out of memory との表記がでて状態爆発が発生した．適用後は，states は 2917781，transitions は 5487218 であり，状態爆発を起こさなかった．このことから，30%以上の抽象化が行えた．

5. 評　　価

まず，本提案手法の評価で用いる SESSAME[16] の「話題沸騰ポット」の適用事例を 5.1 節で説明し，次に本提案手法の評価を 5.2 節で行う．

5.1　話題沸騰ポットの適用事例

SESSAME[16] の「話題沸騰ポット」の温度制御機能に対する本提案手法の適用事例を述べる．ここで「話題沸騰ポット」の温度制御機能とは，蓋を閉じた際に，設定温度への保温を行う機能である．話題沸騰ポットの基本仕様の抜粋を以下に記載する．

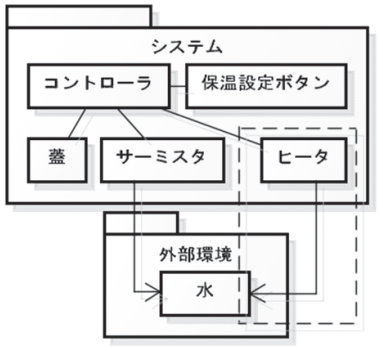

図 10　ポットの設計の概略図
Fig. 10　Design overview of pot.

- サーミスタによりポット内の水温を検出する．
- サーミスタは，−10°C〜150°C の範囲で小数第 1 位までの値の測定が可能．
- ヒータによりポット内の水を加熱する．
- 水の温度特性により決定される比例係数，微分係数，積分係数を使った PID 制御[15] で目標温度を制御する．
- 蓋センサにより蓋が閉じているかを検出する．
- 温度設定ボタンにより「98°C（初期値）」「90°C」「60°C」の保温温度設定が行える．
- 第 1〜4 の水位センサがすべて off のときには加熱しない．

また，話題沸騰ポットの要求仕様では未定である次の仕様を追加する．

- 第 1〜4 の水位センサがすべて off のときには，水の量は 1 リットル未満である．
- ヒータの最大出力は 500 W である．

このソフトウェアの温度制御は，設定された目標温度にするため，サーミスタによりポット内の水温を検出する．水温の検出で PID 成分が決定し，目標温度に制御する．この機能の設計仕様（2.1 節を参照）は次のとおりである．また，その設計の概略図を**図 10** に示す．

- 蓋を閉じた際，沸騰行為をし設定温度への保温を行う．
- 蓋の開閉は蓋センサを用いて監視する．
- 沸騰行為は，コントロール部がサーミスタから温度を取得し，PID 制御によりヒータを用いて加熱する．
- サーミスタから温度を取得する周期は 670 ms である．
- 保温の温度設定は，98°C・90°C・60°C の 3 種類である．
- ヒータで加熱を行う操作量の計算は，P 成分，I 成分，D 成分，そして 1 つ前操作量の総和であり，コントロール部が行う．
- P = (1 つ前の温度) − (現在の温度)
- I = (目標温度) − (現在の温度)
- D = (2 × 1 つ前の温度) − (現在の温度) − (2 つ前の温度)

このシステムにおいて，モデル検査の検証項目は「保温温度を 90°C に設定し，蓋を閉じていればいつかは 90°C となる」や「デッドロックはつねに発生しない」などが考えられる．そのうえで，外部環境記述が出力する外部環境の情報は水温となり，システム記述の振舞いはポットの振舞いとなる．また，モデル検査器の網羅的な探索機能を用いて，水温の全変化パターンの構築を行っている．そのため，検査記述[⋆1]には，図 10 の点線部分の記述はない．ここで外部環境記述には非常に多くのパターンが含まれたため状態爆発が発生した．そこで，外部環境記述の抽象化を行った．また，本稿の説明に必要な部分を抜粋した記述を，ソースコード 3 に示す．

ソースコード 3　potex.pml

```
1  /****************/
2  /* 外部環境記述 */
3  /****************/
4
5  /* 温度生成 */
6  inline getTemp(t0)
7  {
8    if
9    :: d_step{ if
10     :: (t0>=150)−>t0=150;
11     :: else−>t0++;
12     fi; }
13   :: d_step{ if
14     :: (t0<=−10)−>t0=−10;
15     :: else−>t0−−;
16     fi; }
17   :: break;
18   fi;
19 }
20 /* 温度操作量の算出 */
21 inline calcHeatPower(tg,t0,t1,t2,m)
22 {
```

⋆1 補足情報として，検査記述の全体をソースコード 5 に示す．

```
23    int dm;
24    int m0;
25    int m1;
26    getTemp(t0); /*温度取得*/
27    m1=m;
28    d_step{
29    dm=(t1−t0)+(tg−t0)+(2*t2−t0−t2); /*PID 制御アルゴリズム*/
30    m0=m1+dm;
31    if
32    :: (m0<=0)−>m0=0; /*加熱しない*/
33    :: (m0>=100)−>m0=100; /*加熱する*/
34    :: else−>skip; /*現状維持*/
35    fi;
36    t2=t1;
37    t1=t0;
38    m=m0;
39    }
40  }
```

次に，データマッピング法の適用を試みる．ソースコード 3 の t0，t1，t2（21 行目）は外部環境の情報が格納される変数である．それらは複数の変数の組合せ問題であるため，データマッピング法の適用は困難である（3.1 節を参照）．そこで，本提案手法の手順 1〜3 を適用する．

手順 1 の適用

手順 1 の適用結果を**図 11** に示す．ここでは「蓋を閉じた際に，設定温度への保温が行えている」機能を，設計仕様から，サーミスタやヒータを使用する仕様が現れるまでゴール分解を繰り返している．そのことで，システムモデルを構築し，「現在の温度の保持」「1 つ前の温度の保持」「2 つ前の温度の保持」「操作量を基に加熱できている」が現れている．

手順 2 の適用

次に，図 11 を用いて手順 2 の適用を行う．その結果を**図 12** に示す．サーミスタとヒータを用いた 3 階層目の設計仕様である「現在の温度の保持」「1 つ前の温度の保持」「2 つ前の温度の保持」「操作量を基に加熱できている」を図 11 から選択する．そして，「蓋を閉じた際に，設定温度への保温が行えている」機能達成のために，サーミスタに関して「サーミスタを用いて連続した過去 3 回の温度を保持している」という 2 階層目の設計仕様を抽出する．この設計仕様が「システムの動作に必要な外部環境の情報」となる．また，水温の時系列変化が最大となる状況を分析するため，ヒータに関して，水温にどのように影響するかの観点を用いて「ヒータを用いて操作量を基に加熱し，水温を上昇できている」という 2

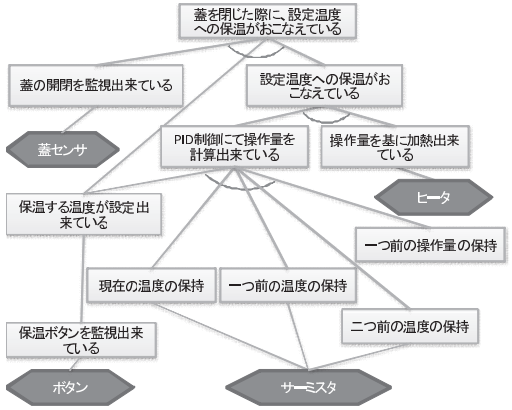

図 11　ポットのシステムモデル
Fig. 11　The pot case of system model.

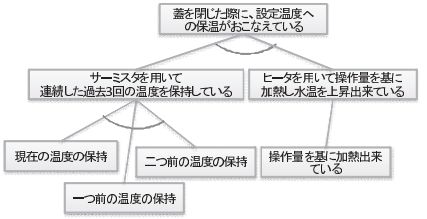

図 12　3 階層の木構造のポットの事例
Fig. 12　The pot case of three hierarchical tree structure.

階層目の設計仕様を抽出する．これが「アクチュエータの操作」となる．

手順 3 の適用

次に，図 12 を用いて手順 3 の適用を行う．その結果を**図 13** に示す．図 13 では，「外部環境の要素」である水温の性質は，要求仕様より「−10°C〜150°C の範囲」「小数第 1 位までの値」となる．次に，要求仕様の「気圧は考慮しない」から，「外部環境の別の要求」はなしとした．そして，「システムの性能」は，ドメイン知識を用いて「水温検出周期は 670 ms」「最大 500 W で加熱」とした．これら 3 つの項目に加えて，手順 2 で抽出した 2 つの項目を加えて 5 つの項目を列挙した．次に，システムの性能が「水温検出周期は 670 ms」「最大 500 W で加熱」，水温が「−10°C〜150°C の範囲」「小数第 1 位までの値」を考慮して，

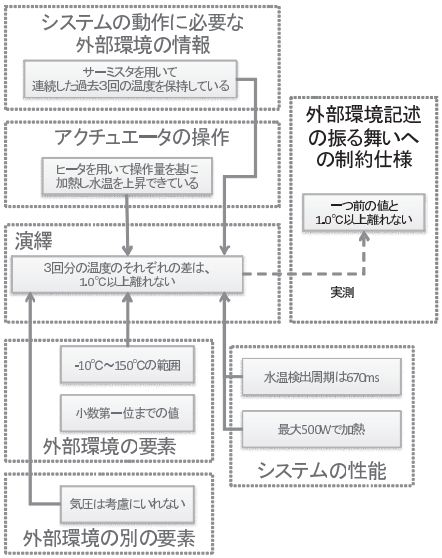

図 13　ポットの水温の変化に対する制約の仕様抽出
Fig. 13　The pot case of giving a limit.

「サーミスタを用いて連続した過去 3 回の温度を保持している」の変化が最大となる「ヒータを用いて操作量を基に加熱し，水温を上昇できている」状況を検討した．その結果，第 1〜4 水位のセンサがすべて off（加熱しない．基本仕様より）になる直前の 500 W による加熱が，水の量が最も少ない（1 リットル）ため，変化量が最大であると分析した．なお，最小変化は 0°C であるため考慮しない．ここで，分析した状況での温度の実測を行う．その結果が「1 つ前の水温と 1.0°C 以上離れない」という条件であったとする[⋆1]．

この制約を外部環境記述に設けた結果をソースコード 4 に示す．

ソースコード 4　pot2.pml

```
1  /* 温度生成 */
2  inline getTemp(t0)
3  {
4    int loop = 10;
5    do
6    ::if
7      ::(loop==0)->break;
8      ::else->skip;
9      fi;
10     if
11     ::d_step{ if
12       :: (t0>=1500)->t0=1500;
13       :: else->t0++;
14       fi; }
15     ::d_step{ if
16       :: (t0<=-100)->t0=-100;
17       :: else->t0--;
18       fi; }
19     fi;
20     loop--;
21   :: break;
22   od;
23 }
```

⋆1 この条件は次の物理法則から妥当であると考える．また，この条件が成立しない状況でも本提案手法の評価には影響しない．
0.804 (°C) = 0.24 (係数) × 500 (W) × 0.67 (s)/1000 (g)

このモデルに対しデッドロックの検査を行ってみると，本提案手法の適用前は，states は 77629548，transitions は 1.0927509e+08 で，適用後は，states が 20735770，transitions は 50644960 であり，53%以上の抽象化が行えた．

5.2　事例を用いた評価

本提案手法である「外部環境の情報の変化に制約を設ける抽象化」を評価するために，3.2 節で述べた次の 2 つの問題点が解決されていることを評価する．

(1)　検査のための「システムの動作に必要な外部環境の情報」と「アクチュエータの操作」を必要十分に設計仕様と検査記述から整理して確認できていること
(2)　5 つの項目（3.2 節）から，外部環境の情報の変化の制約を分析できていること

問題 1 の解決の評価

まず，ライントレーサの例（図 5）の手順 1 の適用で，「PID 制御によるライントレースが行えている」トップゴールを，光センサとモータを用いる設計仕様が現れるまでゴール分解を行った．その結果，ゴール指向分析の特徴である網羅的かつ段階的詳細化によって，「車体が前進できている」「車体が旋回できている」「前回の光センサ値を保持している」「今回の光センサ値を保持している」「前々回の光センサ値を保持している」という設計仕様を抽出した．そして，手順 2 でそれらの設計仕様を，インタフェースごとにボトムアップで整

理することで，すべての設計仕様と検査記述から抽出が困難である「光センサを用いて連続した過去 3 回分の値を保持している」と「モータを用いて前進と旋回し，白と黒を交互にまたいでいる」という，検査に必要十分な「システムの動作に必要な外部環境の情報」と「アクチュエータの操作」の抽出が可能となった．

次に，「話題沸騰ポット」の例においても同様に「サーミスタを用いて連続した過去 3 回の温度を保持している」と「ヒータを用いて操作量を基に加熱できている」と抽出が可能となった．そのため，本提案手法では，1 つ目の問題点が解決できている．

問題 2 の解決の評価

ライントレーサの例（図 9）において，1 つ目の問題点の評価で抽出した 2 つの項目に加え，ドメイン知識を用いて「光センサ値は 500〜700 である」「外乱光は考慮しない」「前進速度は 3（cm/s）」「センシング周期は 1（ms）」との，5 つの項目を列挙できている．また，「光センサを用いて連続した過去 3 回分の値を保持している」ことを考慮すれば，対象とする検査記述の外部環境の情報において，必要十分であると手順 2 の結果より得られている．これらに加えて，一番大きく過去 3 回分の光センサ値が時系列に変わる状況をドメイン知識用いて検討し実測することで，「3 回分のセンサ値のそれぞれの差は，3 以上にならない」との制約を抽出できている．

次に，「話題沸騰ポット」の例においても同様に，1 つ目の問題点の評価で抽出した 2 つの項目に加え，「測定は −10°C〜150°C の範囲で可能」「小数第 1 位までの値の測定が可能」「気圧は考慮に入れない」「水温検出周期は 670 ms である」「最大 500 W で加熱」と 5 つの項目を列挙できている．また，「サーミスタを用いて連続した過去 3 回の温度を保持している」ことを考慮すれば，対象とする検査記述の外部環境の情報において，必要十分であると手順 2 の結果より得られている．これらに加えて，過去 3 回分の水温が時系列に一番大きく変化する状況をドメイン知識を用いて検討し，「1 つ前の水温と 1.0°C 以上離れない」との制約を抽出できている．

以上のように，両事例とも抽出した制約を外部環境記述に設けることで，状態爆発の解消が行えている．よって，本提案手法により，2 つ目の問題点も解決でき「外部環境の情報の変化に制約を設ける抽象化」を可能とした．

6. 関 連 研 究

この章では，外部環境記述の抽象化手法の関連研究を述べる．まず検査記述の抽象化の研究，次にモデル検査を用いた自動テスト生成の研究，ゴール指向分析とモデル検査を用いた研究，そして外部環境の分析によるモデル化の研究について述べる．

検査記述の抽象化手法は，プログラムスライシングに類似した抽象化[4] と，データマッピング法を用いた抽象化[4),7)]，そして偽反例をなくす抽象化[1),2),5)] の研究がある．本提案手法は，外部環境記述の振舞いに制約を設ける抽象化に対して，文献 4)，7) は，外部環境の情報の範囲を削減する抽象化である．まず，プログラムスライシングに類似した抽象化[4] は，システム記述の振舞いに影響しない状態を削除する抽象化である．次に，データマッピング法を用いた抽象化[4),7)] は，システム記述の振舞い結果が同じとなる，外部環境記述の情報をひとまとまりにする抽象化手法である．そして，偽反例をなくす抽象化のうち文献 2)，5) は，モデル検査の結果が NG の場合に出力される反例より，「反例が出ない条件」を導き出し，反例の分析者へ提示する研究である．一方，偽反例をなくす抽象化のうち文献 1) は，自然現象などのリニアな値を含めた検査モデルを扱い，反例が出力される場合に「反例が出ない条件」を導き出し，検査モデルに付与しながら行える範囲で検査を行う．そして「反例が出ない条件」を最後にすべて提示するという研究である．

モデル検査を用いた自動テスト生成の研究[3),14)] は，仕様からテストシーケンスのスイートを構築する手法で，テストに必要な外部入力をモデル検査で抽出する手法である．一方，本提案手法は，設計検証に必要な外部入力を抽出する手法である点が異なる．

ゴール指向分析とモデル検査を用いた研究[13)] は，ゴール指向分析で要求の分析と要件定義を行い，その要件とその後に設計した設計の振舞いとを，モデル検査によって検証するプロセスの提案を行っている．しかし本稿では，「システムの動作に必要な外部環境の情報」と「アクチュエータの操作」を必要十分に設計仕様から整理して確認するためにゴール指向分析を用いており，分析する対象と観点が異なっている．

外部環境を分析しモデル化を行っている研究として文献 6)，17)–20) がある．この研究では，システムラインとコンテキストラインの切り分けを行うため，設計仕様の観点から，外部環境の分析とモデル化を行っている．しかし本稿では，モデル検査を行うため，設計仕様からだけでなく，設計の振舞いからも外部環境を分析しモデル化（システムモデルの構築）を行っている点が異なる．

7. お わ り に

本稿では，分析に必要なセンサとアクチェータの振舞いをゴール指向分析を用いて網羅的に抽出し，外部環境の情報に対する時系列の変化量をドメイン知識を用いて分析，そして外部環境記述の振舞いに制約を設けることで抽象化する手法を提案した．また，ライントレー

サと話題沸騰ポットの事例を用いて，本提案手法の評価とその効果の確認を行い，本提案手法の有効性を示した．さらに，関連する研究との違いについて述べることで，本提案手法の独自性を示した．

本提案手法は，3.2 節で述べた 5 つの項目が複数になる場合，複雑な条件のもとで，外部環境の情報の時系列における分析を行う必要があり困難となる．そのため，今後の課題となる．

参 考 文 献

1) Clarke, E.M., Fehnker, A., Han Z., Krogh, B.H., Ouaknine, J., Stursberg, O. and Theobald, M.: Abstraction and Counterexample-Guided Refinement in Model Checking of Hybrid Systems, *Int. J. Found. Comput. Sci.* (*IJFCS*), Vol.14, No.4, pp.583–604 (2003).
2) Giannakopoulou, D., Pasareanu, C.S. and Barringer, H.: Assumption Generation for Software Component Verification, *ASE 2002*, pp.3–12 (2002).
3) Angelo, G. and Constance, H.: Using model checking to generate tests from requirements specifications, *Proc. 7th European Engineering Conference Held Jointly with the 7th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp.146–162, Springer-Verlag (1999).
4) Bharadwaj, R. and Heitmeyer, C.: Model checking complete requirements specifications using abstraction, *Automated Software Eng. J.*, Vol.6, No.1 (January 1999).
5) Alur, R., Dang, T. and Ivancic, F.: Counter-Example Guided Predicate Abstraction of Hybrid Systems, *TACAS 2003*, pp.208–223 (2003).
6) 鵜林尚靖，金川太俊，瀬戸敏喜，中島　震，平山雅之：コンテキストベース・プロダクトライン開発と VDM++の適用，情報処理学会論文誌，Vol.48, No.8, pp.2492–2507 (2007).
7) Choi, Y. and Heimdahl, M.: Model Checking Software Requirement Specifications using Domain Reduction Abstraction, *18th IEEE International Conference on Automated Software Engineering* (*ASE 2003*), 6.-10.X.2003. Montreal, Canada, IEEE (2003).
8) http://www.etrobo.jp/.
9) Holzmann, G.: *The SPIN Model Checker*, Addison-Wesley (2004).
10) Dardenne, A., Lamsweerde, A.V. and Fickas, S.: Goal-Directed Requirements Acquisition, *Science of Cmputer Programming*, Vol.20, No.1-2, pp.3–50 (1993).
11) McMillan, K.: *Symbolic model Checking*, Springer (1993).
12) Magee, J. and Kramer, J.: *Concurrency: State Models and Java Programming*, John Wiley and Sons (2006).
13) Ogawa, H. and Kumeno, F. and Honiden, S.: Model Checking Process with Goal Oriented Requirements Analysis, *Software Engineering Conference 2008* (*APSEC '08*). 15th Asia-Pacific (2008).
14) Paul, A., Paul, B. and William, M.: Using model checking to generate tests from specifications, *Proc. 2nd IEEE International Conference on Formal Engineering Methods* (1998).
15) システム制御情報学会（編）：システム制御情報ライブラリー 6「PID 制御」，朝倉書店，東京 (1992).
16) 組込みソフトウェア管理者・技術者育成研究会．http://www.sessame.jp/
17) 瀬戸敏喜，金川太俊，鵜林尚靖，鷲見　毅，平山雅之：組込みシステムの外部環境分析のための UML プロファイル，組込み技術とネットワークに関するワークショップ ETNETf2007，2007-EMB-4, pp.65–70 (2007).
18) 鷲見　毅，平山雅之，鵜林尚靖：組込みシステムにおける外部環境の分析，情報処理学会ソフトウエア工学研究会 SE-146, pp.33–40 (2004).
19) 鷲見　毅，平山雅之，鵜林尚靖：組込みシステムにおける動作条件分析手法の提案，電子情報通信学会ソフトウエアサイエンス研究会 SIG-SS-2005-36, pp.19–24 (2005).
20) 鷲見　毅，平山雅之，鵜林尚靖：組込みシステムの動作環境の特徴に着目した仕様分析手法の提案，情報処理学会組込みシステム研究会 EMB-1, pp.7–12 (2006).
21) Weiser, M.: Program slicing, *IEEE Trans. Softw. Eng.*, SE-10(4), pp.352–357 (1984).

付　　録

A.1　話題沸騰ポットの検査記述

ソースコード 5　pot.pml

```
1  /****************/
2  /* 外部環境記述 */
3  /****************/
4
5  /* 温度生成 */
6  inline getTemp(t0)
7  {
8    do
9    ::d_step{ if
10     ::(t0>=1500)->t0=1500;
11     ::else->t0++;
12     fi;}
13   ::d_step{ if
14     ::(t0<=-100)->t0=-100;
```

```
      ::else->t0--;
      fi;}
   :: break;
   od;
}

/*** 状態 ****/
/* ステートタスク */
mtype = {s_idle,s_warming,s_boiling}
/* ヒータ制御タスク */
mtype = {s_off,s_pid,s_heating}
/* 蓋監視タスク */
mtype = {s_opn_cvr,s_cls_cvr,s_init}

/*** イベント ***/
mtype = {
    m_stopWrmCtrl /* 温度制御停止 */
   ,m_compBoiling /* 沸騰処理完了 */
   ,m_reqBoiling /* 沸騰要求 */
   ,m_doBoiling /* 沸騰行為 */
   ,m_doWarming /* 保温行為 */
   ,m_stopDoBoiling /* 沸騰行為停止 */
   ,m_stopDoWarming /* 保温行為停止 */
   ,m_setTemp98 /* 温度設定(98℃)*/
   ,m_setTemp90 /* 温度設定(90℃)*/
   ,m_setTemp60 /* 温度設定(60℃)*/
   ,m_tempSetBtnOn /* 温度設定ボタンON */
   }

/* メッセージ */
mtype = { st,ht,tm,ft }
chan msg = [5] of { mtype,mtype }

/****************/
/* システム記述 */
/****************/

/*** タスク定義 ***/
/* ステートタスク */
active proctype st_tsk()
{
   mtype state = s_idle;
   mtype event;
   do
   ::d_step{
      msg?st(event)->
      if
      ::(state==s_idle)->
      if
      ::(event==m_reqBoiling)->
         state=s_boiling;
         msg!ht(m_doBoiling);
      ::else->skip;
      fi;
    ::(state==s_warming)->
      if
      ::(event==m_stopWrmCtrl)->
         msg!ht(m_stopDoWarming);
         state=s_idle;
      ::(event==m_reqBoiling)->
         state=s_boiling;
         msg!ht(m_doBoiling);
      ::else->skip;
      fi;
    ::(state==s_boiling)->
      if
      ::(event==m_stopWrmCtrl)->
         msg!ht(m_stopDoBoiling);
         state=s_idle;
      ::(event==m_compBoiling)->
         state=s_warming;
         msg!ht(m_doWarming);
      ::else->skip;
      fi;
    fi;
    }
   od;
}

/* 温度捜査量の算出 */
inline calcHeatPower(tg,t0,t1,t2,m)
{
   int dm;
   int m0;
   int m1;
   getTemp(t0); /*温度取得*/
   m1=m;
   d_step{
   dm=(t1-t0)+(tg-t0)+(2*t2-t0-t2);
   m0=m1+dm;
   if
   :: (m0<=0)->m0=0; /*加熱しない*/
   :: (m0>=100)->m0=100; /*加熱する*/
   :: else->skip; /*現状維持*/
   fi;
   t2=t1;
```

```
111    t1=t0;
112    m=m0;
113    }
114  }
115
116  /* ヒータ制御タスク */
117  active proctype ht_tsk()
118  {
119    mtype state = s_off;
120    mtype event;
121    int tg=98;
122    int t0=0;
123    int t1=0;
124    int t2=0;
125    int t_m,m;
126
127    do
128    ::msg?ht(event)->
129      d_step{
130      if
131      ::(event==m_setTemp60)->tg=60;
132      ::(event==m_setTemp90)->tg=90;
133      ::(event==m_setTemp98)->tg=98;
134      ::else->
135        if
136        ::(state==s_off)->
137          if
138          ::(event==m_doWarming)->
139            state=s_pid;
140          ::(event==m_doBoiling)->
141            state=s_heating;
142          ::else->skip;
143          fi;
144        ::(state==s_pid)->
145          if
146          ::(event==m_stopDoWarming)->
147            state=s_off;
148          ::else->skip;
149          fi;
150        ::(state==s_heating)->
151          if
152          ::(event==m_stopDoBoiling)->
153            state=s_off;
154          ::else->skip;
155          fi;
156        fi;
157      fi;
158      }
159    ::(state==s_heating)->
160      if
161      ::calcHeatPower(tg,t0,t1,t2,t_m);
162        m=t_m;
163      ::state=s_off;
164        msg!st(m_compBoiling);
165      fi;
166    od;
167  }
168
169  /* 蓋監視タスク */
170  active proctype ft_tsk()
171  {
172    mtype state;
173    mtype preState=s_cls_cvr;
174
175    do
176    ::if
177      ::state=s_opn_cvr;
178      ::state=s_cls_cvr;
179      fi;
180      if
181      ::(preState==s_cls_cvr)&&(state==s_opn_cvr)->
182        msg!st(m_reqBoiling);
183      ::(preState==s_opn_cvr)&&(state==s_cls_cvr)->
184        msg!st(m_stopWrmCtrl);
185      ::else->skip;
186      fi;
187      preState=state;
188    od;
189  }
190
191  /* 保温設定ボタン監視タスク */
192  active proctype tm_tsk()
193  {
194    do
195    ::if
196      ::msg!ht(m_setTemp60);
197      ::msg!ht(m_setTemp90);
198      ::msg!ht(m_setTemp98);
199      fi;
200    od;
201  }
```

(平成 23 年 3 月 1 日受付)
(平成 23 年 9 月 12 日採録)

乾　　道孝

1978 年生．2002 年大阪工業大学工学部電子情報通信工学科卒業．2004 年大阪工業大学大学院工学研究科電気電子工学専攻修了．同年三菱電機マイコン機器ソフトウエア（株）入社．北陸先端科学技術大学院大学情報科学研究科博士後期課程在学中．

吉岡　信和（正会員）

1971 年生．1993 年富山大学工学部電子情報工学科卒業．1998 年北陸先端科学技術大学院大学情報科学研究科博士後期課程修了．博士（情報科学）．同年（株）東芝入社．2002 年より国立情報学研究所に勤務，現在，同研究所准教授，2007 年より総合研究大学院大学准教授を兼務，セキュリティ技術，エージェント技術，ソフトウェア工学の研究に従事，2011 年より日本ソフトウェア科学会理事，現在に至る．電子情報通信学会，日本ソフトウェア科学会各会員．

落水浩一郎（正会員）

1946 年生．1969 年大阪大学基礎工学部卒業．1974 年同大学院基礎工学研究科博士課程修了．工学博士．静岡大学工学部講師，助教授，教授を経て，1992 年より北陸先端科学技術大学院大学情報科学研究科教授．ソフトウェア工学，特に，オブジェクト指向開発方法論とその支援環境，分散共同開発のプロセスモデルと支援環境，ソフトウェアアカウンタビリティ機能の実現に関する研究に従事．著書に『ソフトウェア工学実践の基礎』（日科技連），『オブジェクトモデリング』（アジソン・ウェスレイ・パブリッシャーズ・ジャパン）等．IEEE，日本ソフトウェア科学会，教育システム情報学会各会員．