博 士 論 文

モデル検査におけるゴール指向分析を用いた外部入力値の時系列変化の制約による状態削減手法

指導教員　鈴木 正人 准教授

北陸先端科学技術大学院大学
情報科学研究科

乾 道孝

平成 26 年 9 月

要 旨

本研究の目的は，センサ・アクチュエータベースのシステムを対象とし，システムの振る舞いを検証する技術を適用する手法の提案である．振る舞いを検証する技術は，近年モデル検査技術が適用されることが多い．また，センサ・アクチュエータベースのシステムにおいては外部環境の情報が膨大であるため，外部環境の情報の組合せによるモデル検査での状態数が多くなり，その状態数の削減が必要となる．一般的な削減手法としては，抽象化手法と，外部環境からの入力値を限定する方法が考えられる．本論文では，入力値を限定する方法，特に今まで整理されていなかった入力値の時系列の変化幅を限定する方法に着目する．

本手法では，変化幅を限定するため，入力値の変化幅が最大となるシナリオ（ワーストケースシナリオ）を分析し，その時の最大の変化幅を得る必要がある．ところが，着目すべき入力値を格納する変数（入力変数）が検査する性質によって異なるため，その変数の抽出が困難である．また，分析したシナリオには，考慮不要な外部環境の要因が含まれることがあり，要求仕様書や設計書にその要因について一般的に明記されていないため排除が困難である．

これらの問題を解決するため本論文では，検査する性質に依存する入力変数を既存のゴール指向分析手法を用いて抽出する手法（ EIVP - Extraction method of Input Value related to a verification Property ）と，考慮不要な外部環境の要因を排除したワーストケースシナリオを分析できるゴール指向分析手法（ GWEU - Goal-oriented analysis of Worst case scenario with Eliminating Unnecessary contexts ）を提案する．また，これらの手法の有効性を２つの事例研究により示す．

本研究の成果により，センサ・アクチュエータベースのシステムの設計検証に対して，従来に比べて状態爆発の発生確率をより軽減でき，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貢献できる．

目 次

第 1章

はじめに

近年，モデル検査技術 [26] 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組み込みソフトウェア業界で注目を集めている．特に，自然現象（外部環境の要素の現象）をサンプリングし状況判断と共に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（図 1.1）では，得られる外部環境の情報が膨大であるため，デッドロックの検出，到達可能性の確認，そして変数値の範囲の逸脱防止の設計検証がレビューでは困難な状況である．そこで，モデル検査

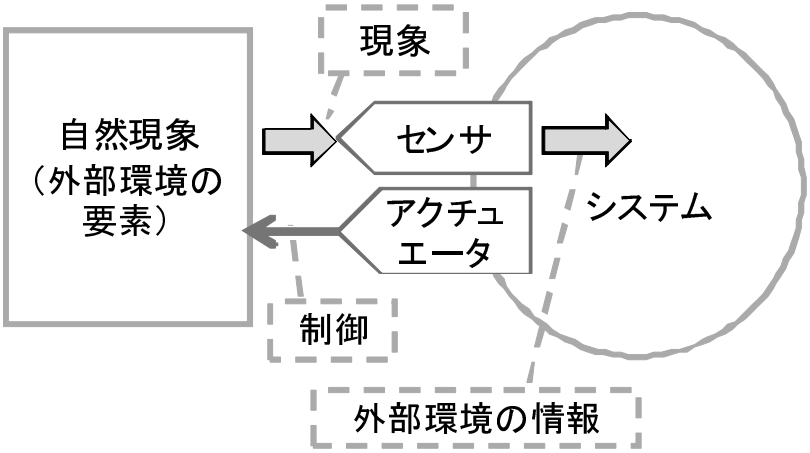

図 1.1: センサ・アクチュエータベースのシステム

の導入が試みられている．センサ・アクチュエータベースのシステムは，システムが自然現象をセンサを用いてサンプリングし，外部環境の情報を基にシステム

が外部環境の状況を把握し，外部環境をアクチュエータを用いて制御するシステムである．モデル検査を実施するためには，図 1.2 に示す，モデル検査器へ入力するための検査モデルが必要である．検査モデルには，システムの振舞い部のシステムモデル，外部環境の状態がモデル化されシステムへの入力値を生成させる外部環境モデル，そして検査する性質がある．この検査モデルを本論文では対象とし，対象の検査モデルと呼ぶ．

ところが，自然現象をサンプリングした値の組合せが膨大な数となり，状態爆発の要因となりやすく状態数の削減が必要である．特に連続的な値を入力とする

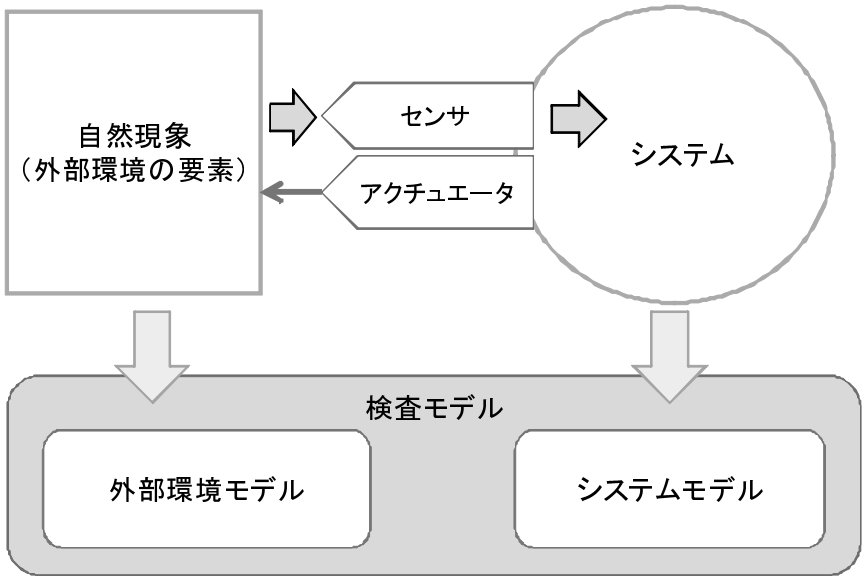

図 1.2: 検査モデルの概要

設計モデルを検査する場合，外部環境記述が生成する情報が膨大となり，モデル検査器で探査する空間が非常に大きくなる．理由として，入力値の変化時系列に対応する空間と，振る舞いを決定する状態変数の空間の直積が，モデル検査器上での探査空間となるためである．そのため，その設計モデルに対するモデル検査では，探査に必要なメモリ空間が不足する状態（状態爆発）が発生しやすい．

この状態数の削減には，抽象化手法と，外部環境からの入力値を限定する方法が一般的に考えられる．抽象化手法は，検査モデルに存在する状態集合に対して，検査する性質を保持しつつ抽象構造を小さく作り，全体的な状態数を削減す

る手法である．一般的な抽象化手法の概要を図 1.3 に示す．抽象化ではシステムモデルと外部環境モデルに対して，検査する性質を抽象化の前後で同じとした上で，状態削減を行う必要があり，主な手法としてデータマッピング法 [13][19]，プログラムスライシング [57] の技術に類似した抽象化法 [13]，そして述語抽象化法 [33] などがある．一方，外部環境からの入力値を限定する方法は，外部環境を

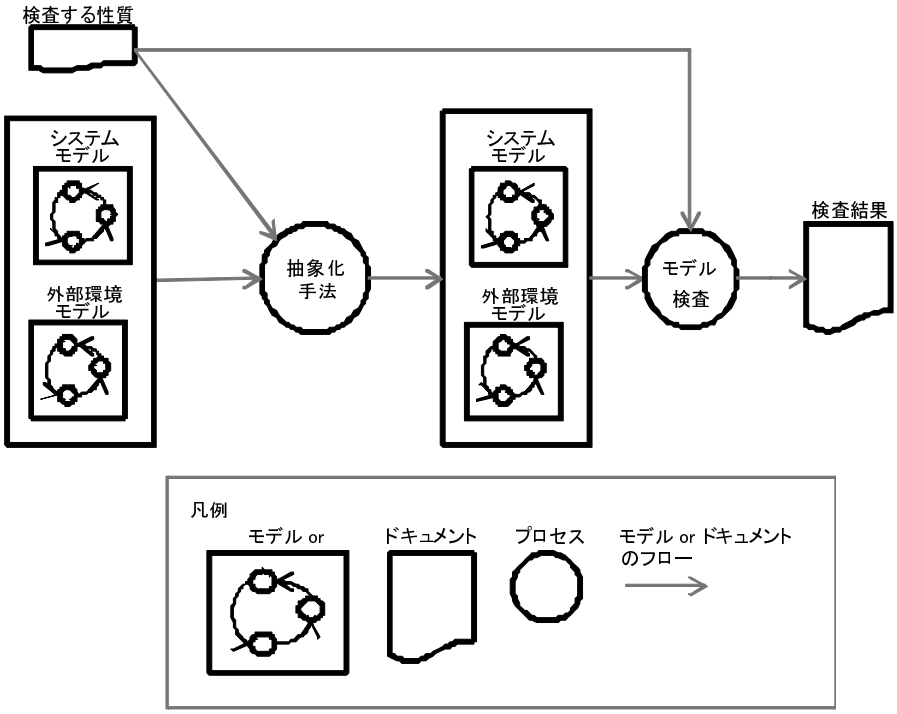

図 1.3: 抽象化法の概要

分析することでモデル検査に不要な入力値を限定する手法である．一般的な入力値を限定する手法の概要を図 1.4 に示す．限定手法は外部環境モデルに対して，システムの実行環境など外部の要因や検査する性質を分析し，不要な（例えば物理的に発生しない，または検査に必要がない）外部環境の入力を決定することで状態削減を行う手法であり，ユースケースから考慮不要な入力値を削減する方法 [21][22][25] などがある．なお，抽象化手法と入力値を限定する手法は依存なく適用が可能である．

本論文では，入力値を限定する方法，特に今まで整理されていない，入力値の時系列の変化幅を限定させる方法に着目する．時系列の変化に着目した理由は，

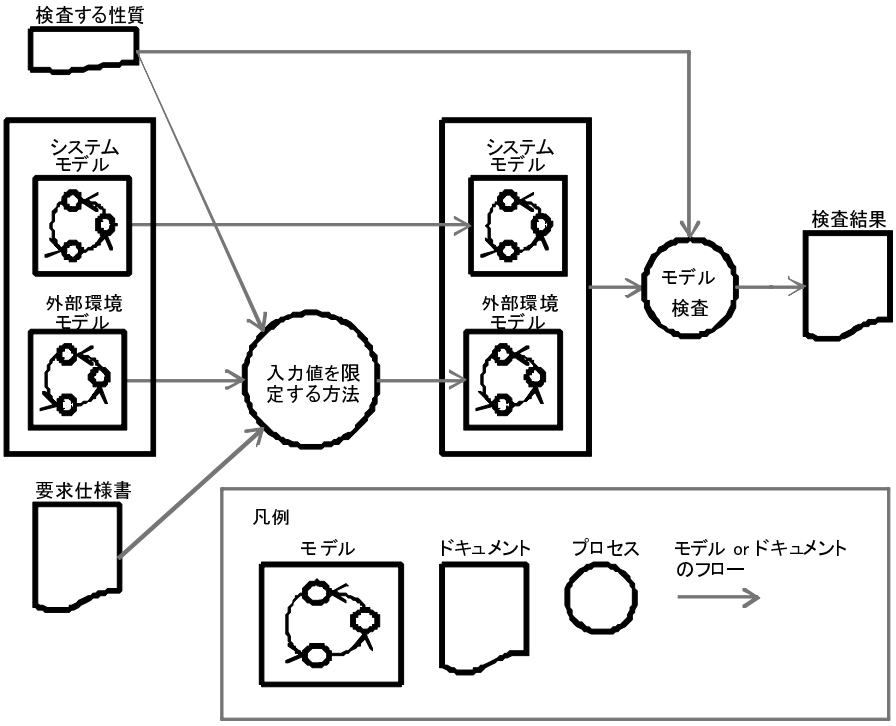

図 1.4: 入力値を限定する手法の概要

自然現象の値を連続的に取得する設計モデルを対象としているため，入力値の時系列の変化幅の限定により，探査空間の大幅な削除が期待されるためである．本手法の流れを図 1.5 に示す．まず，システムの動作シナリオの中で，入力値の変化幅が最大となるものを選択し，次にそのシナリオにおいてプロトタイププログラミングを用いてその入力値の変化幅を実測し，そしてモデル検査における変化幅を実測値の最大変化幅に制限（時系列変化の制約）することによって，探査空間を削減する．システムが動作する手順の中で，入力値（センサ値）の時点 n から時点 $n+1$ までの変化幅が最大となるを “ワーストケースシナリオ” と本論文では呼び，その分析には検査する性質，要求仕様書，設計モデル，そしてドメイン知識を用いる．本手法を適用するフェーズは，設計フェーズであり設計者自身がモデル検査を行うことを想定する．図 1.5 の要求仕様書は，機能要求，非機能要求，そしてセンサやアクチュエータの仕様が記載されているものとする．また，ドメイン知識は，要求仕様書に記載されていない，時系列の入力値が変化す

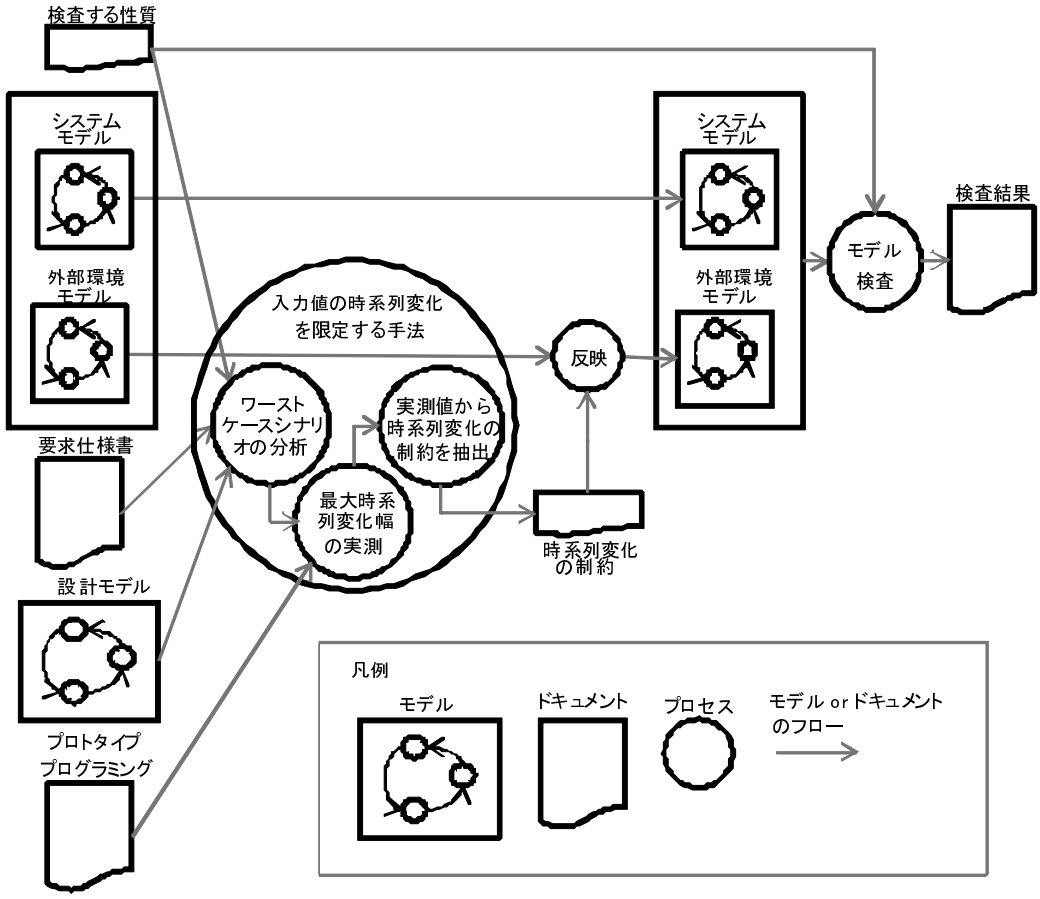

図 1.5: 提案手法の流れ

る要因の知識情報を指す．例えば，水を加熱する機能に関して，水は直接要因であるのに対して，水温が上昇する上で関係する沸点や気圧は間接要因である．次に，検査する性質は，連続的なセンサ値の入力値の時系列の変化を基に外部環境の状況を把握しアクチュエータで制御を行う，システムが振る舞う性質を検証するものである．最後に，設計モデルは，要求仕様書を基に設計された UML の状態遷移図やシーケンス図などの振舞いモデルである．

ところが，ワーストケースシナリオの分析には，検査する性質に関係する入力値の最大時系列の変化が発生する状況を検討する必要があるが，着目すべき入力変数（入力値が格納される変数）が検査する性質によって異なり抽出が困難である．その要因として，設計モデルや検査する性質には，この入力変数のすべてが現れていないためである．そのため，ワーストケースシナリオの分析において，

もし着目すべき入力変数を全て抽出していなければ，抽出できている場合と比べて，妥当な分析が行われるとは言えない．また，ワーストケースシナリオによっては，考慮不要な外部環境の要因が含まれており排除する必要があるが，要求仕様書や設計書にその要因について一般的に明記されていないため排除が困難である．もし排除されるべき要因が残ると，入力値の最大変化幅が大きくなり，時系列の変化の制限幅が大きくなるため，探査空間を十分に削減できない．

そこで，これらの問題を解決するため本論文では，検査する性質に依存する入力変数を抽出する手法（ EIVP - Extraction method of Input Variable related to a verification Property ）と，考慮不要な外部環境の要因を排除したワーストケースシナリオを分析できるゴール指向分析手法（ GWEU - Goal-oriented analysis of Worst case scenario with Eliminating Unnecessary contexts ）を提案する．それらにより，対象の検査モデルにおいて，モデル検査時の探査空間が削減され，状態爆発の発生確率の軽減が可能となる．EIVP は，検査する性質に依存した入力変数を抽出するために，従来の基本的なゴール指向分析（AND の関係を用いたゴール分解）を用いた手法である．また GWEU は，ワーストケースシナリオに含まれる具体的な外部環境の要因を分析するため，入力値の変化要因がシステムの振舞いか外部環境の要因かの観点でグルーピングを行いながらゴール分解を行い，グループ化された具体的な外部環境の要因から考慮不要なものを分析する記法と手順を提案したゴール指向分析手法である．

本論文で提案する手法の有効性を２つの事例に適用し，モデル検査時の探査空間における一定の削減効果を確認する．この成果により，複数の変数の組合せ問題になる場合でも更なる状態数の削減が可能となり，状態爆発の発生確率を従来より軽減できる．本論文の事例では SPIN[31] を用いるが，その他のモデル検査器（SMV[38]，LTSA[36] など）への適用も可能である．その理由として，本提案手法では，外部環境記述へ反映する時系列変化の制約を抽出し，対象の検査モデルに反映してモデル検査を行うが，検査モデルに反映する手法については言及していないため，モデル検査器に依存した手法ではないためである．

本論文の構成は，本研究で扱う基盤技術について 2 章で述べ，次に対象システムと特徴および状態爆発が発生する事例を 3 章で説明する．その後，入力値の時系列の変化を限定する手法について 4 章で述べ，その手法に含まれるワースト

ケースシナリオを分析する手順での２つの問題点を5章にてあげる．そして，その対策とするゴール指向分析を用いたワーストケースシナリオの分析方法を6章で提案し，２つの事例による本提案手法の評価を7章で行う．最後に，本研究の関連研究を8章にて述べ，9章にてまとめと今後の課題を述べる．

第 2章

基盤技術

本章では，本論文に関係する基盤技術として，モデル検査の技術を 2.1 節にて，ゴール指向分析の技術を 2.2 節にて述べる．

2.1 モデル検査

モデル検査（Model Checking）とは，形式手法の一つであり，ソフトウェアの設計モデルが仕様を満たしているかについて，モデルに存在する状態を，そのモデルと時相論理式をコンピュータに入力することで自動的に網羅探索し検査する技術である．モデル検査を行うためには，モデル検査を実行するツール（モデル検査器）が必要である．そのモデル検査器には，SMV[38] を拡張した NuSMV[3]，FDR[30]，LTSA[36]，そして SPIN[31] などがある．NuSMV は，CTL と LTL という 2 種類の時相論理式を用いた有界モデル検査法が提供されている．また，FDR と LTSA は共に，Hoare の CSP 系のプロセス代数 [39] を基にした仕様言語を入力としている．特に FDR は，CSP の意味論に従った検証方法を採用しており，一方，LTSA は有限のプロセス表現を特殊化した FSP を入力とするモデル検査器で，ラベルを付与することで状態遷移モデルベースの形式を実現したものである．また SPIN は，オートマトンベースのモデル検査器で，検査する対象のシステムもしくは検査する性質は記述言語 Promela (Process Meta Language) で記述され，FDR と同様にラベルを付与することで状態遷移モデルベースの形式を実現したものである．Promela は，非同期分散アルゴリズムを非決定性オートマト

ンとしてモデル化し，検査する性質は時相論理式で表される．この論理式の否定形を Buchi オートマトンに変換して Promela のモデルと合成し，網羅探索を行うオートマトンの一部とするモデル検査器である．

2.1.1 抽象化

抽象化とは，プログラム中の各モジュールの状態の中で，検査したい性質に関係のない状態を排除することによって，構造の状態数を小さくすることで実現される．ある一つのプログラムに対して，検査したい性質は複数存在しており，各モジュールのほとんどの状態はこれらの性質のいずれかに関係している．しかし，そのプログラムに対して性質の検査を試みれば，検査に不要な状態が多く含まれるため，探査空間が多くなり状態爆発が発生しやすくなる．そのため，検証したい性質ごとに抽象化を行う必要がある．一般的な状態削減のための抽象化手法には，データマッピング法による抽象化 [13][19]，プログラムスライシング [57] の技術に類似した抽象化 [13]，そして述語抽象化 [33] などがある．

例えば，7.2 節で述べるライントレーサの事例において，検査モデル 2.1 [1] のような黒（ $now \geq th$ （36 行目））の時は左旋回，白（ $now < th$ （38 行目））の時は右旋回とした On/Off 制御であれば，検査モデル 2.2 のような白と黒の 2 値（9，11 行目）としたデータマッピングの抽象化が可能である．つまり，検査モデル 2.2 の 1 行目にて，その 2 値を *mtype* で表現し（定義を増やし），ランダム生成の値に対するデータマッピングで抽象化した値として用いている．

検査モデル 2.1: etrb_on_off.pml

```
1 mtype = {get}
2 chan light = [0] of { int }
3
4 /* 外部環境記述 */
5 active proctype external()
```

[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…繰り返し 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．

```
{
  int color0 = 500;
  /* コース上の光センサ値 */
  /* 500(白)〜700(黒)を */
  /* ランダムに生成 */
  do
  ::do
    ::(500<color0)->color0--;
    ::(700>color0)->color0++;
    ::break;
    od;
    /* システム記述へ送信 */
    light!color0;
  od;
}

/* システム記述 */
active proctype system()
{
  /* 今回取得した光センサ値変数 */
  int color0 = 0;
  /* 設計した白黒の閾値 */
  int th = 575;
  /* 旋回値変数 */
  int turn0 = 0;
  /* 外部環境記述から受信 */
  do
  ::light?color0;
    if
    /* color0>=th時 50の力にて右旋回 */
    ::(color0>=th) -> turn0= 50;
    /* color0< th時 50の力にて左旋回 */
    ::else -> turn0=-50;
    fi;
  od;
}
```

検査モデル 2.2: etrb_on_off_g.pml

```
mtype = {get,over_th,under_th}
```

```
2  chan light = [0] of { mtype }
3
4  /* 外部環境記述 */
5  active proctype external()
6  {
7    do
8    /* 575以上の場合 */
9    ::light!over_th;
10   /* 575未満の場合 */
11   ::light!under_th;
12   od;
13 }
14
15 /* システム記述 */
16 active proctype system()
17 {
18   /* 今回取得した光センサ値変数 */
19   mtype color0;
20   /* 旋回値変数 */
21   int turn0 = 0;
22   /* 外部環境モデルから受信 */
23   do
24   ::light?color0;
25     if
26     /* 575以上時 50の力にて右旋回 */
27     ::(color0==over_th) -> turn0=50
28     /* 575未満時 50の力にて右旋回 */
29     ::else -> turn0=-50;
30     fi;
31   od;
32 }
```

2.1.2 過大近似と過小近似

抽象化には，過大近似と過小近似の考え方がある．プログラムの振舞いの過大近似（over-approximation）とは，変換後のプログラムが変換前のプログラムの振舞いを全て含むような近似である．逆に，変換後のプログラムは変換前のプロ

グラムには存在しない実行パスの経路を含む．例えば，プログラムの安全性などの性質（常に成立が望ましい性質）に対して，弱保存を満たす抽象化を行う際に利用される．ここで弱保存とは，抽象化後のプログラムにおける性質が真である場合，抽象化前のプログラムにおける性質も真であれば弱保存の関係と呼ばれる．つまり，抽象化前のプログラムより実行パスが増えた抽象化後のシステム問題がなければ，抽象化前のプログラムの性質も成立することになる．

一方，プログラムの振舞いの過小近似（under-approximation）とは，変換前のプログラムが変換後のプログラムの振舞いを全て含むような近似である．逆に，変化後のプログラムは変換後のプログラムの実行パスの一部の経路を含まない．そのため，不変性質に対しては弱保存は明らかに成り立たないが，不変性質をモデル検査する際に利用されることがある．例えば，同じ機能を実現する上で，バッファのサイズを削減する方法，またはシステムのプロセス数を削減する方法は過小近似で行われる代表的な手法である．

2.1.3　一般的な抽象化の手順

一般的な抽象化の適用手順について図 2.1 を用いて説明する．凡例中の「モデル or ドキュメントのフロー」は，モデルとドキュメントがプロセスへ入力もしくは出力される手順であり，その矢印の下の括弧は流れる条件を示している．抽象化は，設計の振舞いの性質を保ちながら，振舞いを追加（過大近似）する場合や検査する性質に関係しない振舞いを削除（過小近似）する場合がある．

まずプロセス１は，要求仕様書を基に設計を行い設計モデルを構築する一般的な手順である．この設計モデルは，モデル検査を行う対象となる設計の振る舞いモデルである．次にプロセス２では，設計モデルと検査する性質から検査モデル（検査する性質に依存する部分を設計モデルから抽出したモデル）の構築を行う．この検査モデルには，図 1.2 に示すシステムモデルと外部環境モデルが含まれ，作成時に抽象化が用いられることもある．次にプロセス３では，検査する性質と検査モデルをモデル検査器へ入力し検査結果を得る．ここで，状態爆発が発生した事により十分な検査ができなかった場合，プロセス４へと移行する．プロセス４では，検査する性質と検査モデルから抽象化を行い，抽象化により状態を削減

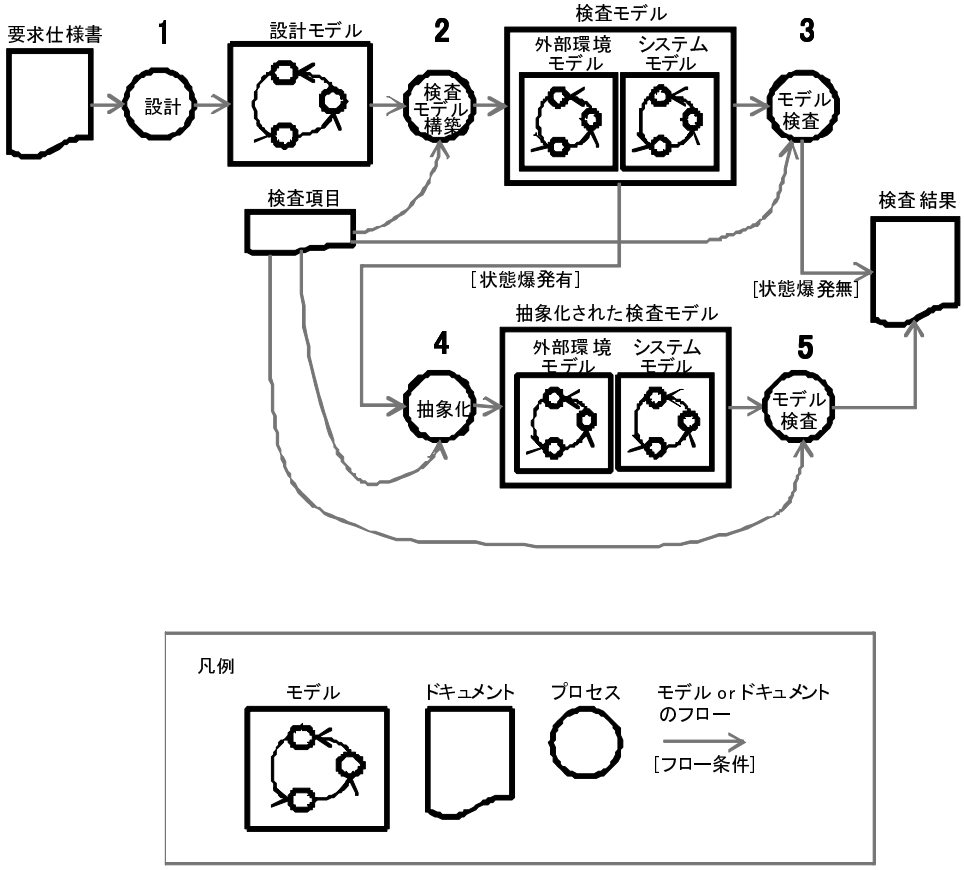

図 2.1: 一般的なモデル検査の手順

された検査モデルを構築する．最後のプロセス５では，プロセス３と同様，検査する性質と抽象化された検査モデルを入力としたモデル検査を行う．このプロセス５で状態爆発が発生した場合，更に抽象化がなされる場合もある．

2.2 ゴール指向分析

ゴール指向分析とは，何らかの Goal（目標）を満たすために，どのような Sub goal（具体的な目標）が必要であるか，またはそれらの関係はどのようなものであるかを，通常はトップダウンに分析する技法である．この分析手法は，古くから多くの分野で利用されており，特に生産管理などの現場ではゴール分解として用いられている．与えられた問題に対し，その原因から解決法を見出そうとする

発想は，われわれ人間が持っている基本的な思考法である．しかし，問題の設定の仕方やゴールをどこまで分解すればよいかの判断，ゴール分解の中で発生する Sub goal 同士の衝突への対応などは，解の品質に大きく影響を与える上，判断基準が適用領域に依存する．そのため，判断基準を適用領域ごとに検討する必要がある．

2.2.1 一般的な記法

図 2.2 をゴール木と呼び，一番トップの Goal を Top goal とし，分解後の Goal を Sub goal とする．そのゴール分解を繰り返すことで，具体的な目標の抽出およびそれらの関係を分析を行う．ゴール分解は，基本的に AND 分解と OR 分解が存在する．AND 分解は，分解されたゴール全てが達成されることで，分解前のゴールが達成される関係の分解である．表記例として，図 2.2 の Top goal は Sub goal 1 と Sub goal 2 の AND 分解である．OR 分解は，分解されたゴールのいずれかまたは全てが達成されることで，分解前のゴールが達成される関係の分解である．表記例として，図 2.2 の Sub goal 1 は Sub goal 3 と Sub goal 4 の OR 分解である．

2.2.2 ゴール指向要求分析

要求工学 [7][55] におけるゴール指向要求分析は，ゴール指向分析のサブセットであり，ドメイン分析やシナリオ分析と共に，代表的な要求分析法の１つであり，個々の要求にはその要求の元となる上位目標があるという前提のもとに，ゴール分解を通して要求の導出を行おうとする手法であり，いくつも手法が提唱されている [61]．代表的なものに KAOS[8][27]（Knowledge Acquisition in autOmated Specification），i*[58][59][60]，NFR フレームワーク [42][44] などの手法がある．KAOS はゴール指向要求分析のサブセットであり，米 Oregon 大学とベルギー Louvain 大学の共同研究（1900 年～）により提案され，ゴール指向要求分析のゴール分解と形式仕様を組み合わせた手法で，分解された Sub goal の中に要求を発見するためのもので，次の四つのモデルから構成されている．

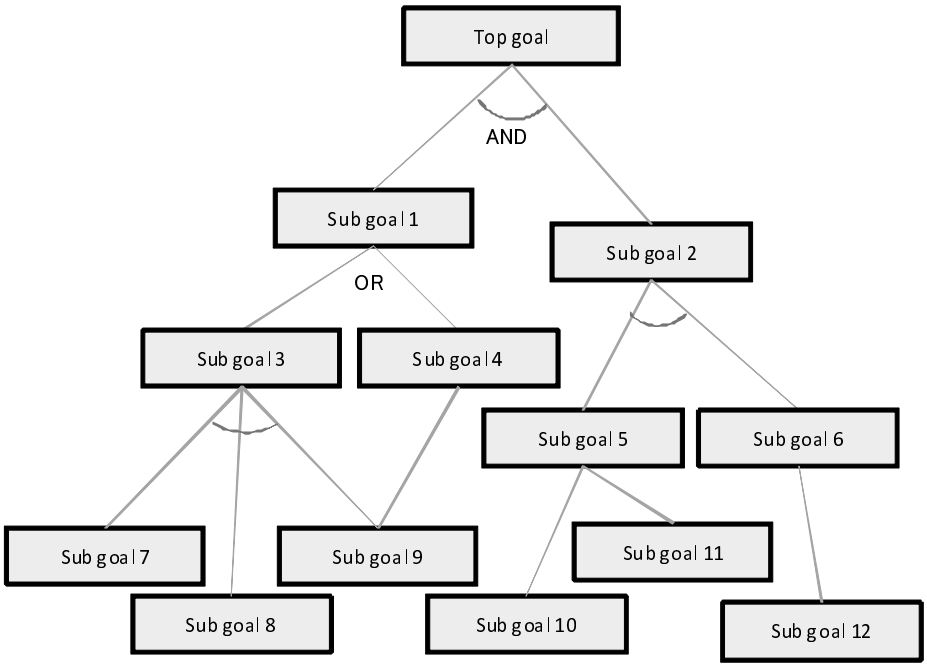

図 2.2: ゴール木

ゴールモデル

ユーザのニーズや意図から個別の要求までを，目標とする状態によって系統的に表現したモデルである．Agent の責務となる Sub goal が現れるまでゴール分解が行われることで，ゴールの洗練化が行われる．また，非機能要求をソフトゴールとして表現し，各ゴールがその非機能要求に対する阻害要因となれば「衝突」の表記を用いる．

責務モデル

どの Agent が，どのゴールの達成に責務を持っているのかを表現したモデルである．KAOS 法では，対象 Agent が特定できた時点でゴール分解を停止することになっている．

操作モデル

ゴール達成のための操作と，物理的なオブジェクトに対する入出力を表現したモデルである．KAOS 法では，要求（末端ゴール）の形式的な記述から，操作を導出する．これを Operationalization という [6] ．

オブジェクトモデル

ゴールを実現するための操作で使用するオブジェクトを表現したモデルであり，UML のオブジェクト図に相当する．

第 3章

対象システム

対象とするシステムの特徴を 3.1 節で説明し，検査対象となる性質の特徴を 3.2 節で述べ，状態爆発が発生する事例を 3.3 節にて述べる．

3.1　対象システムの特徴

本論文で対象とするシステムは，図 1.2 に示すとおり，外部環境の情報をセンサを用いて取得し，外部環境の状態を把握し，そして外部環境をアクチュエータを用いてシステムを操作するリアクティブシステムである．ここで，外部環境の情報とは，水や音や色などの物理量をセンサで取得した情報である．また，このセンサ・アクチュエータシステムは，あるシステムに外部環境から入力を与えると，それに対する応答をある一定時間内に外部環境に出力することを繰り返し実行するリアクティブシステムに分類される．本論文では複数のセンサとアクチュエータを持つシステムを対象とし，特に自然の物理量を周期的にセンサで取得した入力値（物理量の離散値）を扱う検査モデルを対象とする．複数のセンサとアクチュエータを持つシステムを対象とする場合は，単一のシステムを対象とした場合の提案手法を利用することで対応が可能であるため，まず 6 章の手法を提案する箇所にて単一を対象とした場合について言及し，次に 7.5 節の評価の章にて対象が複数である場合の適用について述べる．

まず単一の入出力を持つ対象の検査モデルは，出力変数（アクチュエータの振舞いに直接関係する値を格納する変数）を Y，入力変数（センサによる入力値を

格納する変数）を X とすると，ある定数 a,b において 3.1 式となる

$$Y^0 = f(Y^1, Y^2, \cdots, Y^a, X^0, X^1, \cdots, X^b) \quad (1 \leq a, 0 \leq b) \tag{3.1}$$

ここで，入力変数 X について X^0 を現在値, $X^k(0 \leq k \leq b)$ を k 時点前の値を格納する変数として定義する．また，出力変数 Y について Y^0 を現在値, $Y^k(1 \leq k \leq a)$ を k 時点前の値を格納する変数として定義する．例えば出力変数 Y^0 は，出力値の履歴の変数 Y^1, Y^2 と，入力値の履歴の変数 X^0, X^1, X^2, X^3 が関係する場合，その Y^0 を算出する関数 f は 3.2 式となる．

$$Y^0 = f(Y^1, Y^2, X^0, X^1, X^2, X^3) \tag{3.2}$$

次に複数の入出力を持つ対象の検査モデルについて，センサが m 個，アクチュエータが n 個ある場合，ある定数 a,b において，
入力変数のベクトルを $\mathbf{X} = [X_1, X_2, \cdots, X_m] \quad (1 \leq m)$
出力変数のベクトルを $\mathbf{Y} = [Y_1, Y_2, \cdots, Y_n] \quad (1 \leq n)$
とすると，求める出力 $\mathbf{Y}^0$ は 3.3 式であらわされる．

$$\mathbf{Y}^0 = \mathbf{F}(\mathbf{Y}^1, \cdots, \mathbf{Y}^a, \mathbf{X}^0, \cdots, \mathbf{X}^b) \quad (1 \leq a, 0 \leq b) \tag{3.3}$$

例えば，2 種類のアクチュエータと 2 種類のセンサの場合，$\mathbf{F} = [F_1, F_2]$ とすると，3.4 式と 3.5 式が対象の検査モデルとなる．

$$Y_1^0 = F_1(X_1^0, X_1^1, X_1^2, X_2^2, X_2^3) \tag{3.4}$$

$$Y_2^0 = F_2(X_1^2, X_1^3, X_2^0, X_2^1) \tag{3.5}$$

対象の検査モデルは，外部環境の状態を把握する必要があるため，出力変数と比べて入力変数の数が多くなる傾向がある．結果として，入力である外部環境の情報の組合せ数が膨大となり，状態爆発を起こす要因となりやすい．対象の検査モデルが単一のセンサ・アクチュエータである場合の組合せ数が膨大となる理由は次の通りである．入力変数に格納される値はセンサで取得した入力値（物理量の離散値）を扱っており，3.1 式の入力変数 $X^0, X^1, \cdots, X^b$ それぞれに格納される入力値により，外部環境モデルの入力値の組合せが構成される．そのため，全

ての変数とその履歴を組み合わせると組合せ数が膨大となりやすい．モデル検査を行う上では，システムモデルに含まれる状態数とその入力値の組合せ数の論理積が検査中の状態数となるため，入力値の組合せ数の要因により状態爆発が発生しやすい．なお，この入力変数に格納される値の上下限の範囲はセンサの仕様に依存する．これらのことから，対象の検査モデルは外部環境の情報の組合せが膨大となり，組合せ数の削減が必要である．

例えば，対象の検査モデルに含まれる入力の履歴数が 4 である場合，サンプルする値の範囲が 1〜10 の時に組合せ数は 10 の 4 乗となり，1〜1000 の時は 1000 の 4 乗となる．さらに，システムモデルの状態数との論理積による状態数は膨大となり，状態爆発が発生しやすい．7.2 節のライントレーサの事例では，取りうる範囲の光センサ値は 500〜700，検査モデルで扱う入力の履歴数が 3 つあることから，入力値の組合せは約 200 の 3 乗（800 万）となる．さらに，その組合わせ数と論理積を行うシステムモデルの状態数に依存するところもあるが，その入力値の組合せ数が状態爆発の要因となる（その状態爆発が発生する例を 3.3 節にて示す）．一方，7.1 節の話題沸騰ポットの事例においては，サーミスタは-10 °C〜150 °C の範囲で小数第一位までの値の測定が可能，検査モデルで扱う入力の履歴数が 3 つあることから，入力値の組合せは約 1600 の 3 乗（40 億）となり，ライントレーサの事例と比べても 500 倍である．そのため，この組合わせ数は，システムモデルの状態数との論理積が行われることを考慮すると状態爆発の要因となりやすい．この事から，システムモデルの状態数にも依存するが，外部環境モデルによる入力値の組合せ数が，状態爆発の発生要因となっていることがわかる．また，3.1 式の b の数は，事例に依存するがこれらの事例では 3 であった．本論文では，その組合せを絞り込む一つの手法を提案する．

3.2 検査対象となる性質の特徴

本論文で対象の検査を行う性質を V とすると，V は時相論理で表現可能な入力値または出力値を含む性質である．つまり，V は対象のモデル（3.1 式）に含まれる出力変数の時系列 $Y^0, \cdots, Y^a$ および入力変数の時系列 $X^0, \cdots, X^b$ を含む時相論理の式である．また，検査する性質には一般的に下記４つが存在するがい

ずれも対象である．

- 到達可能性：ある特定の状態に本当に到達するか
- 安全性　　：ある条件の下で何かが決して起こらないか
- 活性　　：ある条件の下で最終的には何かが起こるか
- 公平性　　：ある条件の下で何かが何度も起こるか

なお，検査を実施する上でその対象となる性質の表現方法は，時相論理でも assert 文などを用いてもよいため本論文では言及しない．ただし，assert 文の使用について，活性や公平性に関する性質は assert 文だけでは表現できない．例えば，到達可能性の「常に変数 var が A の状態である．」に対する時相論理であれば，「assert(ver==A)」と表現すれば，時相論理を assert 文で表現できる．しかし，「ある条件の下で最終的に何かが起こる」や「ある条件の下で何かが何度も起こる」は表現ができない．本研究での事例の検査する性質は，到達可能性のものであるため assert 文で表現する．

3.3　状態爆発が発生する事例

本論文で対象とするシステムの事例として，7.2 節で述べるライントレーサロボットの事例を用いる．このシステムは，ラインをスムーズにトレースするための一般的な技術である PID 制御を用いており，その設計モデルを抽象化したものを検査モデル 3.1 に示す．

検査モデル 3.1: etrb_pid.pml

```
1  chan light = [0] of { int }
2
3  /*********************************************/
4  /*****************外部環境記述*****************/
5  /*********************************************/
6  active proctype external()
7  {
8    /* ライントレーサが取得する光センサ値の初期値 */
```

```
9     int color0 = 500;
10
11    /* ライントレーサが取得する光センサ値 */
12    /* (500〜700)の全組合せパターンを */
13    /* 構築する */
14    do
15    ::do
16      ::(500<color0)->color0--;
17      ::(700>color0)->color0++;
18      ::break;
19      od;
20      /* システムモデルへ送信 */
21      light!color0;
22    od;
23  }
24
25
26  /***********************************************/
27  /*****************システム記述*****************/
28  /***********************************************/
29
30  /* 設計した白黒の閾値 */
31  #define TH 575
32
33  /* P成分計算(36はP成分係数) */
34  /* P成分変数:ラインとライントレーサの進行 */
35  /* 方向の角度 (光センサ値の今回と前回の取得値の差を角度) */
36  #define P_VALUE ((color0-color1)*36)
37
38  /* I成分計算(1はI成分係数) */
39  /* I成分変数:ラインとライントレーサが離れ */
40  /* た距離 (光センサ値の前々回・前回・今回の */
41  /* それぞれの値と,閾値との差の積分)*/
42  #define I_VALUE ((color0-TH)*1)
43
44  /* D成分計算(4はD成分係数) */
45  /* D成分変数:ラインとライントレーサとの */
46  /* 角速度 (光センサ値の,前々回・前回と前回・今回の角度の差) */
47  #define D_VALUE (((color0-color1)-(color1-color2))*4)
```

```
48
49 /* 旋回値決定 */
50 #define POWER (P_VALUE + I_VALUE + D_VALUE)
51
52 active proctype system()
53 {
54   /* 今回取得した光センサ値変数 (X^0) */
55   int color0 = 0;
56   /* 旋回値変数 (Y^0) */
57   int turn0 = 0;
58   /* 前回の旋回値変数 (Y^1) */
59   int turn1 = 0;
60   /* 前回取得した光センサ値変数 (X^1) */
61   int color1 = 0;
62   /* 前々回取得した光センサ値変数 (X^2) */
63   int color2 = 0;
64
65   /* 外部環境モデルから受信 */
66   do::light?color0;
67
68     /* 旋回値決定 */
69     turn0 = POWER;
70
71     /* 検査する性質 */
72     /* 今回のセンサ値と閾値の差が前回のセンサ値と */
73     /* 閾値の差より小さければ,モータへのパワー値は */
74     /* 前回より必ず小さい.」*/
75     if
76     ::!(((color0-TH)<(color1-TH))&&(turn0<turn1))->assert(false);
77     ::else;
78     fi;
79
80     /* 前々回取得した光センサ値を更新 */
81     color2 = color1;
82     /* 前回取得した光センサ値を更新 */
83     color1 = color0;
84     /* 前回旋回値を更新 */
85     turn1 = turn0;
86   od;
```

```
87 }
```

3 行目から 23 行目までの外部環境記述と，26 行目から 87 行目までのシステム記述がある．また，検査する性質は 76 行目までの「今回のセンサ値と閾値の差が前回のセンサ値と閾値の差より小さければ，モータへのパワー値は前回より必ず小さい」である．システム記述において，69 行目で算出している出力変数 $turn0$ に関連する入力値の履歴が格納される変数は，現在値の入力変数 $color0$（36 行目，42 行目，47 行目），前回値の入力変数 $color1$（36 行目，47 行目），そして前々回値の入力変数 $color2$（47 行目）である．一方，出力値の履歴が格納される変数がないことから，3.1 式に置き換えると次のようになる．出力変数 Y^0 に相当する $turn0$，その値を算出するための関数 f_1 は，現在値が格納される X^0 に相当する入力変数 $color0$，前回値が格納される X^1 に相当する入力変数 $color1$，そして前々回値が格納される X^2 に相当する入力変数 $color2$ によって算出され，$\mathbf{X}$ = [color] および $\mathbf{Y}$ = [turn] とすると 3.6 式となる．

$$turn0 = f_1(color0, color1, color2) \tag{3.6}$$

このシステムに対してモデル検査を行った結果，外部環境記述からシステム記述へ入力される 500〜700 の値のあらゆる組合せで，モデル検査時の探査空間が構築されることから状態爆発が発生した．この状態爆発を回避するため，抽象化手法と外部環境からの入力値を限定する方法が一般的に用いられるが，本論文では入力値を限定する方法，特に時系列の入力値の変化を限定する方法に着目する．その方法について 4 章で説明する．

第 4章

入力値の時系列の変化を限定する手法

本章では，入力値の時系列の変化を限定する手法について言及する．まず入力値の時系列の変化の限定について 4.1 節で説明し，次にその変化の限定方法について 4.2 節で述べ，そしてその変化幅の限定に使用する “ワーストケースシナリオ” について 4.3 節で述べる．

4.1　入力値の時系列の変化の限定

システムがセンシングする外部環境の情報の中で，設計検証において考慮不要な情報を排除するため，入力値の時系列の変化に制約を設ける．つまり，図 4.1 のセンサ値の組合せ集合から，考慮不要なセンサ値の時系列変化を排除する制約（本論文では時系列変化の制約）を設ける事で，探索する入力空間を削減させる方法である．時系列の変化の制約に着目した理由は次の通りである．対象の検査モデルは，自然現象の値を連続的に取得することで外部環境の状況を把握する．そのため，入力値はもともと物理量であるため短時間に大きく変化する可能性は低い．このことから，入力値の時系列の変化幅を限定できると，探査空間を削除でき状態爆発の回避が期待できる．

なお，限定の効果は対象の検査モデルにより異なる．センサ値の時系列の変化幅が大きい（例えば t 時と t+1 時のセンサ値の変化が，最大値から最小値まで極端に変化する）と効果が少ない．つまり，制約の時系列変化の幅が大きくなり，入力値の組合せはあまり減少されず探査空間の削減効果が少ない．ただし今回の

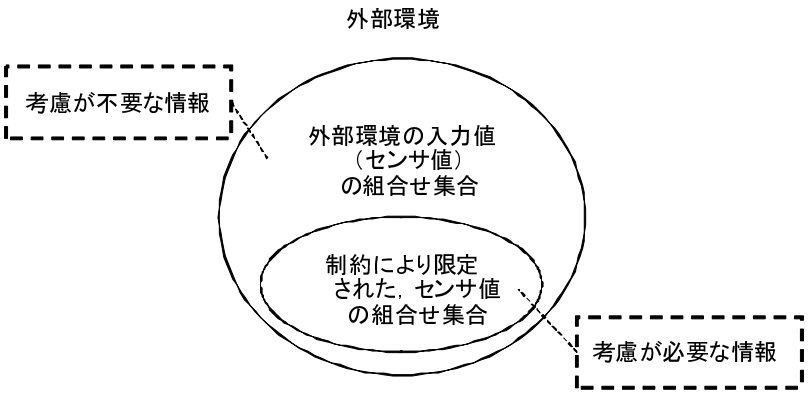

図 4.1: 抽出する時系列変化の制約の観点

対象は連続的に変化する物理量 (色や力) なので変化幅は小さく，削減効果も期待できる．

4.2 制限値による変化幅の限定手法

入力値の時系列の変化の限定は図 4.2 のように，その時系列の変化が最大となるシステムのシナリオ（“ワーストケースシナリオ”）をシステムが動作する環境から分析し，次にそのシナリオでの最大の変化幅を実測，そしてその最大の変化幅をモデル検査上の入力値の変化の制限値として用いることで行う．“ワーストケースシナリオ” は，センサ値の時系列変化の最大値を測定することを目的としており，システムの内部状態を決定する履歴と入力系列（センサ値）を発生させるシステムの動作順序である．その定義と分析方法を 4.3 節にて記載する．このシナリオは，図 4.3 に示すように，システムが動作する上で，X^t 時と X^{t+1} 時の入力値（センサ値）の変化幅が最大となる状況を含んだものである．また，このシナリオでの入力値の実測にはプロトタイププログラミングを用い，その実測値から最大の変化幅を抽出し，モデル検査上の入力値の変化の制限値として用いる．その結果，入力値の組合せの削減が行え，モデル検査時の探索空間の削減が可能となる．つまり，対象の検査モデル 3.1 式において，入力値の組合せ（$X^0, X^1, \cdots, X^b$ の変数の履歴の組合せ）は，b+1 次ベクトルで各値の候補が

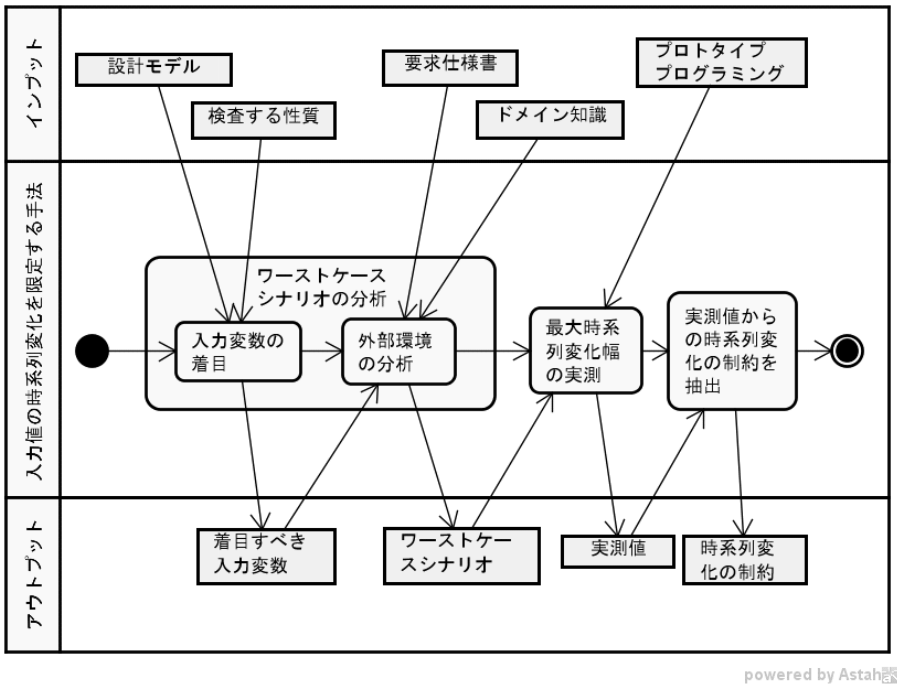

図 4.2: 入力値の最大変化幅の抽出手順

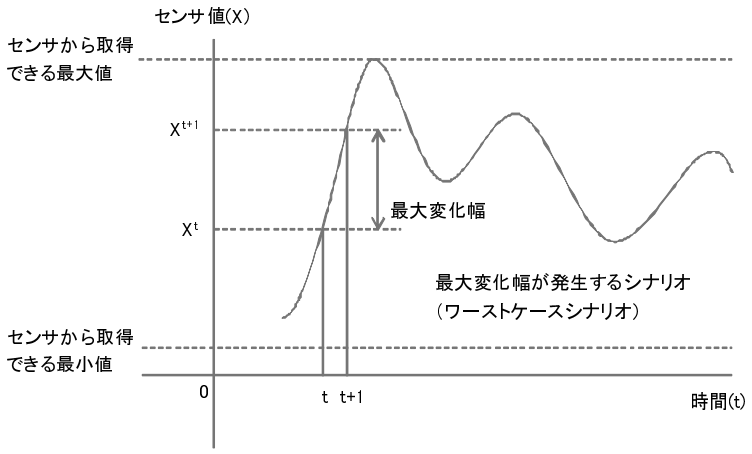

図 4.3: 入力値の最大変化幅

ΔX 個なら全体は $(\Delta X)^{b+1}$ となるが，制限値を入力値の変化の制約として用いる（$0 \leq \Delta X \leq$制限値 とする）ことでモデル検査時の探査空間の削除が可能となる．

例えば，マイクで音量（1〜10 までの範囲の入力値）を取得しある一定の音量

になるようにスピーカの出力を調整するシステム（入力変数を X，出力変数を Y とする 4.1 式とした検査モデル）の例を挙げる．

$$Y^0 = f(X^0, X^1) \tag{4.1}$$

そのシステムが実行される環境を考慮して，連続でサンプリングする音量の時系列の変化が最大となるシナリオを分析し，そのシナリオにて時系列変化の実測を行い，その結果が図 4.4 のように最大変化幅が 4 であったとする．この場合，セ

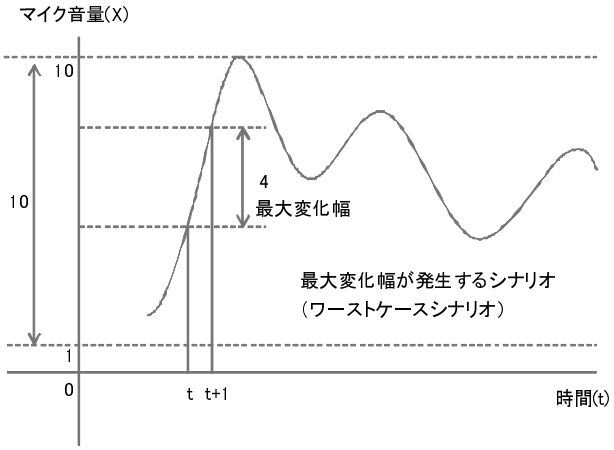

図 4.4: マイク音量の最大変化幅

ンサ値の時系列の変化幅が 5 未満であるとの制約が得られる．この制約により，前回に取得したセンサ値が 1 であれば，次は 1〜5 の 5 通りの組合せとなる．つまり，適用前は図 4.5 のように，t 時の送信値が 1 の場合 $t+1$ 時の送信値が 1〜10 の 10 通りであるが，適用後は図 4.6 のように，$t+1$ 時の送信値が 1〜5 の 5 通りとなり組合せの削減が行える． よって，検査モデル 4.1 式において，前提条件が $0 \leq \Delta X \leq 9$ から $0 \leq \Delta X \leq 4$ に変わり，モデル検査時の入力値の組合せが 10^2 から 5^2 に削減され，探査空間が 4 分の 1 に削減される．

4.3 ワーストケースシナリオ

ワーストケースシナリオとは，センサ値の時系列変化の最大値を測定することを目的としており，システムの内部状態を決定する履歴と入力系列（センサ値）

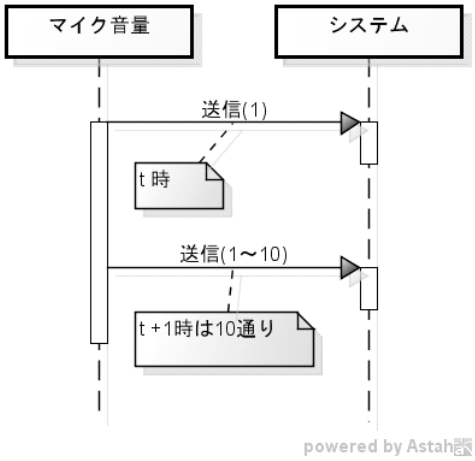

図 4.5: 外部環境モデルのマイク音量送信（適用前）

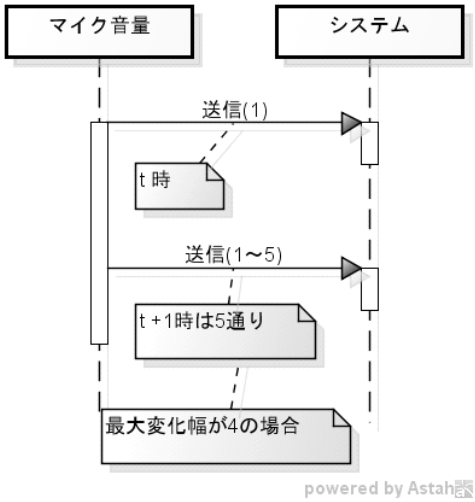

図 4.6: 外部環境モデルのマイク音量送信（適用後）

を発生させるシステムの動作順序である．また，本限定手法では 4.2 節で述べた通りこのシナリオ上で入力値を実測する必要があるため，このシナリオはシステムが動作する次の情報を含む.

- システムの内部状態
- 場所やタイミングを含む外部の状況

これらの情報は，入力値が変化する要因である，システムの振舞いによる要因と，外部環境による要因のいずれかに関係を持つ．その理由は，この入力値の実測に必要なシナリオの情報は，入力値が変化する要因をシナリオ化して表現したものだからである．つまり，ワーストケースシナリオに含まれる，システムの内部状態の情報は，センサ値が変化する要因（システムの振舞いによる要因）の情報である．また，場所やタイミングを含む外部の状況の情報も，センサ値が変化する要因（外部環境による要因）の情報である．

次に，図 4.2 のワーストケースシナリオの分析について説明する．ワーストケースシナリオの分析には，時系列変化する対象の入力値に焦点を当てる必要がある．また，モデル検査を行う上で必要な情報は検査する性質に依存するため，その入力値に焦点を当てるためには，その値が格納される検査する性質に依存する入力変数の着目が必要となる．なお，この検査する性質に依存する入力変数を本論文では “着目すべき入力変数” と呼ぶ．次にその入力値の時系列変化の要因を要求仕様書とドメイン知識を用いて分析し，ワーストケースシナリオを抽出する．

第 5章

“ ワーストケースシナリオ ”を分析する上での２つの問題点

4.2 節で述べたワーストケースシナリオの分析における，２つの問題について 5.1 節と 5.2 節で説明する．

5.1　問題１：着目すべき入力変数の抽出が困難

図 4.2 に示すようにワーストケースシナリオを分析するためには，まず検査する性質に依存する入力変数に着目（“着目すべき入力変数” を抽出）する必要がある．しかしながら，その変数は検査する性質や設計モデルにすべて表現されない．その理由は次のとおりである．対象の検査する性質（3.2 節を参照）には，出力値の履歴が含まれ，それぞれの出力値の算出には時系列を基準とした入力値の履歴が必要である．しかしながら，3.2 節で述べたように，それらの入力値の履歴が格納される変数（着目すべき入力変数）は設計モデルに全て現れないためである．具体的には，3.1 節で述べた対象システムの例である，3.2 式（$Y^0 = f(Y^1, Y^2, X^0, X^1, X^2, X^3)$）の設計モデルに対して，ある時間 U において，検査する性質に出力値 Y^U と Y^{U+3} が表現されている場合，算出する関数はそれぞれ 5.1 式と 5.2 式となる．

$$Y^U = f(Y^{U+1}, Y^{U+2}, X^U, X^{U+1}, X^{U+2}, X^{U+3}) \tag{5.1}$$

$$Y^{U+3} = f(Y^{U+4}, Y^{U+5}, X^{U+3}, X^{U+4}, X^{U+5}, X^{U+6}) \tag{5.2}$$

そのため，この着目すべき入力変数に格納される値は，5.1 式と 5.2 式から，少なくとも $X^U, \cdots, X^{U+6}$ となり，着目すべき入力変数は $X^0, \cdots, X^6$ を必要とする．しかしながら，$X^4, \cdots, X^6$ は，設計モデルと検査する性質に表現されていないため，抽出が困難である．

3.3 節で述べた対象システムの一つである，ライントレーサの事例を用いて，考慮すべきセンサ値の抽出が困難である例を示す．ライントレーサの設計モデルは，3.6 式の $Y^0 = f(X^0, X^1, X^2)$ であるのに対し，1 単位時間過去を T と表記すると，検査する性質は「今回のセンサ値と閾値の差が前回 (1T) のセンサ値と閾値の差より小さければ，モータへのパワー値は 1T より必ず小さい」である．この性質を V，V に含まれる入出力変数の集合を $R(V)$ とすると，$R(V) = \{Y^0, Y^1, X^0, X^1\}$ となる．ここである時間 U において，検査する性質には，今回のセンサ値 X^U，1T のセンサ値 X^{U+1}，今回のモータへのパワー値 Y^U，そして 1T のモータへのパワー値 Y^{U+1} が表現される．これらの Y^U と Y^{U+1} のセンサ値とのの関係式は 5.3 式と 5.4 式となる．

$$Y^U = f(X^U, X^{U+1}, X^{U+2}) \tag{5.3}$$

$$Y^{U+1} = f(X^{U+1}, X^{U+2}, X^{U+3}) \tag{5.4}$$

そのため,「着目すべき入力変数」に格納される値（センサ値）X は，検査する性質に含まれている X^U と X^{U+1} に加えて Y^U と Y^{U+1} の関係式の 5.3 式と 5.4 式から $X^U, \cdots, X^{U+3}$ である．よって，それらの値を格納する「着目すべき入力変数」は $X^0, \cdots, X^3$ となる．しかしながら，検査する性質に表現されている値に関してモデル検査を行う上で扱う変数の中で，X^3（3 つ前 (3T) の値を格納する変数）は，設計モデルの 3.6 式と検査する性質に現れていないため抽出が困難である．

5.2 問題２：考慮不要な外部環境要因の排除が困難

着目すべき入力変数を抽出した後，ワーストケースシナリオを分析する必要があるが，不要な外部環境の要因も含まれるため，それらを排除することが困難で

ある．ここで不要な外部環境の要因とは，実際には発生しない（考慮しなくてもよい）外部環境の要因の事である．不要な外部環境の要因には，ワーストケースシナリオでの「実測時の状況」で考慮が不要な外部環境の要因と,「初期状態から実測までの状況」を考慮すると不要な外部環境の要因が存在する．その要因が排除できなければ，分析したワーストケースシナリオでプロトタイププログラミングを用いて最大変化幅を実測した場合，その変化幅が大きくなり時系列変化の制約が大きくなり，十分な状態削減が行われない．その要因の排除が困難である理由として，ワーストケースシナリオは要求仕様およびドメイン知識から外部環境を考慮して分析されるが，そのワーストケースシナリオにトレースされた不要な外部環境の要因について，要求仕様書や設計書にも一般的に記載がないためである．

3.3 節で述べたライントレーサの事例を用いて，考慮不要な外部環境の要因の排除が困難である例を示す．5.1 節で述べた内容から，仮に着目すべき入力変数が，今回から 3T までのセンサ値を格納する変数 $X^0, \cdots, X^3$ とし，それぞれ格納される入力値の変化が最大となる状況を分析し，その結果「フルスピードで急カーブを曲がる」との状況を想定した．また，フルスピードになるためには，PID 制御で走行し，フルバッテリ残量の状態で，坂道を下った直後の状態が，ワーストケースシナリオと分析した．そして，そのワーストケースシナリオにてプロトタイププログラミングを用いて実測した結果，入力値の時系列の変化の最大は 5 であったため，その値を時系列変化の制約としてモデル検査（検査モデル A.1 を参照）を行ったが状態爆発が発生した．時系列変化の制約は，検査モデル A.1 の 16 行目に記載されており，また 32 行目と 33 行目にて比較を行っている．しかしながら，コースのスタート地点から坂道までには距離があり，バッテリ残量が最大であることは実運用上ありえなかった．そのため，フルバッテリ残量で坂道を下る条件ではなく，少しバッテリを消費した状態で実測すべきであった．また，この考慮が必要な状態については，要求仕様書には記載されていなかった．再度，プロトタイププログラミングを用いて実測し，その結果の時系列変化の制約を 3 として (外部環境記述のみを記載した検査モデル A.2 を参照のこと) モデル検査を行ったところ，状態爆発は発生しなかった．検査モデル A.1 との違いは，検査モデル A.2 の 16 行目の値を 3 に変更しているところである．このワースト

ケースシナリオにおいて，フルバッテリ残量の場合は考慮不要であるとの情報は，7.2 節の要求仕様書や設計書にも記載されておらず排除が困難であった．つまり，バッテリ残量について「実測時の状況」のみ着目しており,「初期状態から実測までの状況」を考慮できていなかったため排除が困難であった．

これら 5.1 節と 5.2 節の問題を解決できるゴール指向分析手法を 6 章にて提案する．

第 6章

ゴール指向分析を用いた“ ワーストケースシナリオ ”の分析手法

本章では，5 章で述べた 2 つの問題に対して，既存のゴール指向分析手法を用いて設計モデルから着目すべき入力変数を抽出する手法（ EIVP - Extraction method of Input Variable related to a verification Property ）と，考慮不要な外部環境の要因を排除したワーストケースシナリオを分析できるゴール指向分析手法（ GWEU - Goal-oriented analysis of Worst case scenario with Eliminating Unnecessary contexts ）を提案する．

EIVP のオリジナリティは，KAOS のサブセットとしてゴール分解に意味を与えたことである．下位ノードの論理積が上位ノードである KAOS のゴール分解の意味に加えて，下位ノード（対象の検査モデルの式の右辺の変数）の論理積が上位ノード（対象の検査モデルの式の左辺の変数）との意味を加える．また，GWEU のオリジナリティは，考慮不要な外部環境の要因の発見を容易にし，システムの振舞いと外部環境の要因の 2 つの観点でグループ化しつつゴール分解を行うゴール指向分析の記法と手順である．

まず，提案するワーストケースシナリオの分析手法の概要について 6.1 節で述べ，次に EIVP について 6.2 節で述べ，そして GWEU について 6.3 節で述べる．

6.1　ワーストケースシナリオの分析手法の提案

提案するワーストケースシナリオの分析手法の流れを図 6.1 に示す．5.1 節で述べたように図 4.2 の「入力変数の着目」で，検査する性質に依存する入力変数（“着目すべき入力変数”）を抽出することが困難である．これに対し，既存のゴール指向分析手法を用いて設計モデルから着目すべき入力変数を抽出する手法（ EIVP ）を提案する．また，図 4.2 の「外部環境の分析」で，ワーストケースに含まれる考慮不要な外部環境要因を排除することが困難である．これに対し，考慮不要な外部環境の要因を排除したワーストケースシナリオを分析できるゴール指向分析手法（ GWEU ）を提案する．図 6.1 のワーストケースシナリオの分析において，まず設計モデルと検査する性質に表現される値を格納する入出力変数の集合から，EIVP を用いて着目すべき入力変数の集合を抽出する．次に，

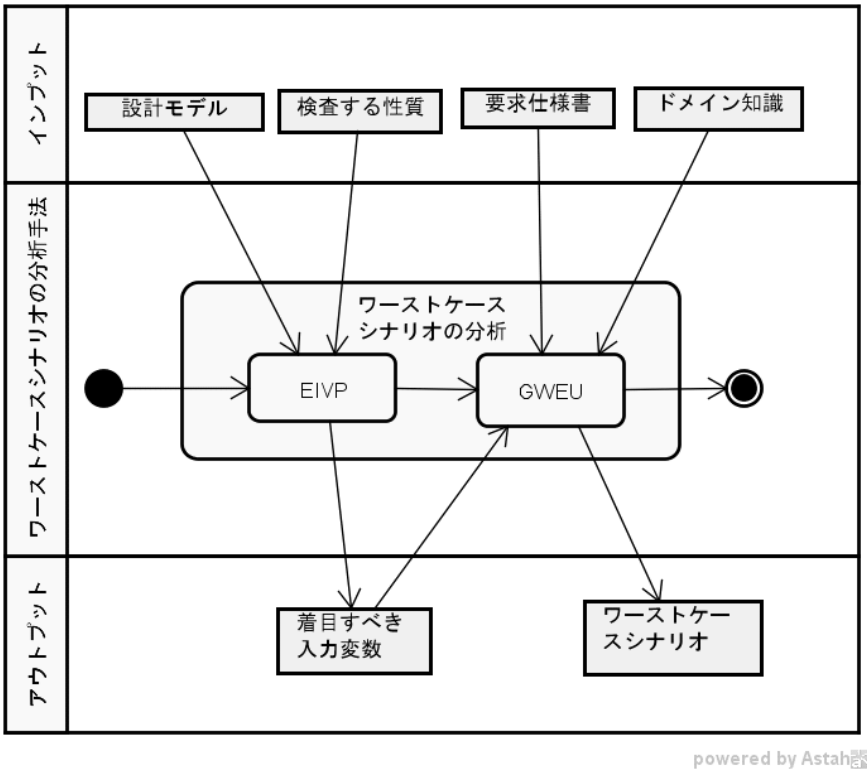

図 6.1: 提案するワーストケースシナリオの分析手法の流れ

EIVP で抽出した入力変数の集合に着目し，センサ値の時系列変化が最大となる

状況が発生するワーストケースシナリオ（考慮不要な外部環境要因を排除したもの）を，GWEU を用いて抽出する．

この GWEU の詳細な流れを図 6.2 で示す．これは３つの手順で行うゴール指向分析手法である．まず手順１では，システムが動作する全てのユースケース上でセンサ値の時系列変化が最大となる状況を，着目すべき入力変数を用いて分析する．次に手順２では，手順１で抽出したセンサ値の時系列変化が最大となる状況を Top goal としてゴール分解を行いワーストケースシナリオを分析する．このゴール分解を行う上で，センサ値が時系列に変化する要因がシステムの振舞いと外部環境の要因の２つが存在するため，それら２つの観点を用いてゴール分解を行い，ワーストケースシナリオのゴール木を構築する．最後手順３にて，構築したゴール木から考慮不要な外部環境要因を分析し，考慮不要な外部環境要因を排除したワーストケースシナリオを抽出する．

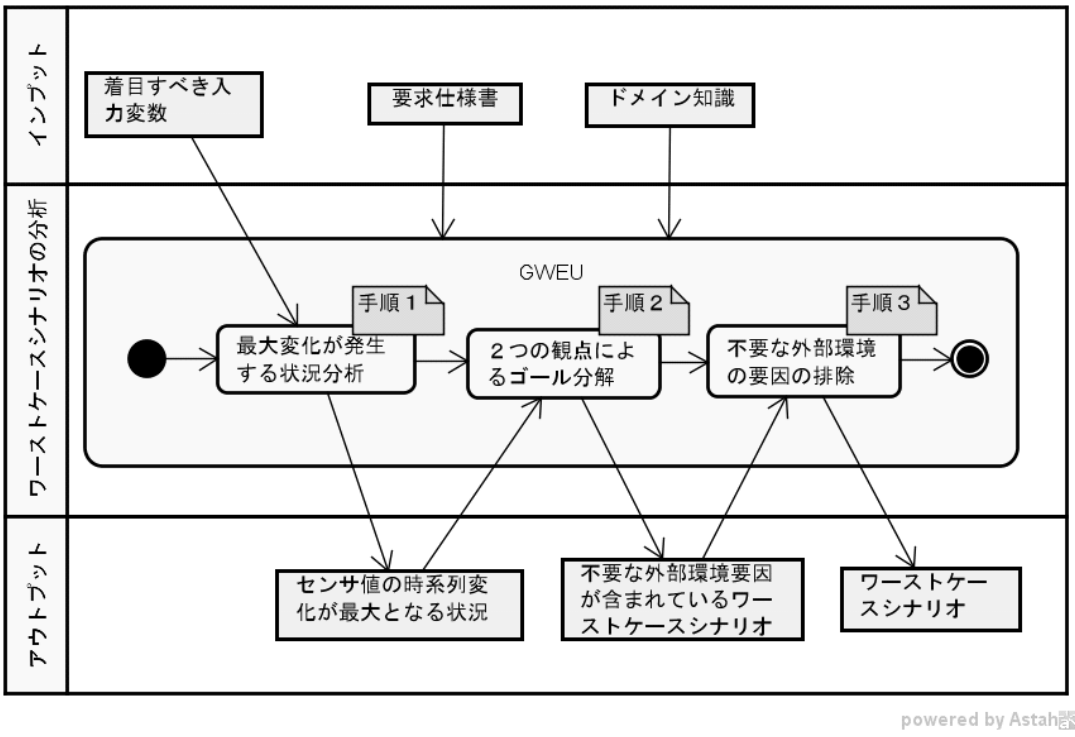

図 6.2: GWEU の３つの手順

6.2　着目すべき入力変数を抽出する手法（EIVP）

本節では 5.1 節の問題１に対応した，着目すべき（検査する性質に依存する）入力変数の集合を抽出する手法（EIVP）を提案する．EIVP の目的は，検査する性質に表現されている入出力値が格納される入出力変数の集合から，ゴール指向分析を用いて，検査する性質に依存する入力変数の集合を抽出することである．

まず，本手法で用いるゴール指向分析の基本的な記法を 6.2.1 節で述べ，次に着目すべき入力変数を抽出する手順を 6.2.2 節で述べる．

6.2.1　EIVP で使用するゴール指向分析の記法

本記法は，図 6.3 に示す通り，KAOS の Top goal，Sub goal，AND 分解，そして Agent の表記を用いる．

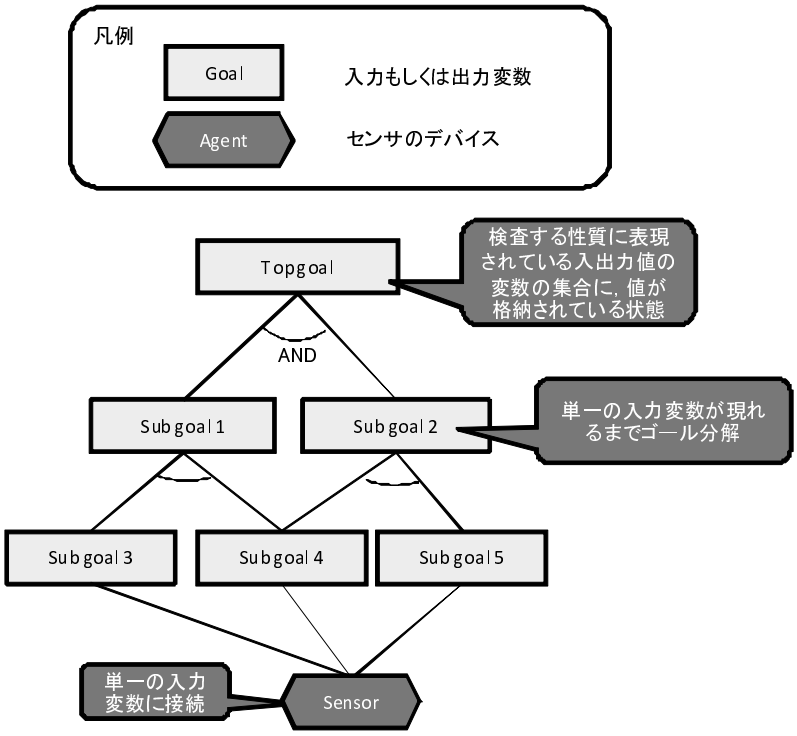

図 6.3: EIVP で使用するゴール指向分析の記法

まず Top goal は，検査する性質に表現されている入出力値が格納されている変

数の集合とする．次に，Sub Goal は Top goal を AND 分解して記載する．ここでゴール分解の意味は KAOS のサブセットであり，下位ノードの論理積が上位ノードである KAOS の意味に加えて，下位ノード（6.1 式の右辺の引数（変数））の論理積が上位ノード（6.1 式の左辺の変数）との意味を加える．

$$Y^t = f(Y^{t+1}, Y^{t+2}, \cdots, Y^{t+a}, X^t, X^{t+1}, \cdots, X^{t+b}) \quad (0 \leq a, 0 \leq b) \tag{6.1}$$

6.1 式は，対象の検査モデル 3.1 式の現在 Y^0 から t 回過去の出力履歴 Y^t の入出力変数の関係式である．また，st はシステムが起動して現在までの総履歴数である．

6.2.2 EIVP の詳細手順

Top goal の入力変数の集合を，単一の変数になるまで AND 分解する詳細な手順を説明する．まず，検査する性質を V とし，V に表現されている入出力値が格納される入出力変数の集合を $R(V)$ とした時，$R(V)$ と出力変数 Y および入力変数 X の関係式は，対象の検査モデルの 3.1 式を基に 6.2 式となる．

$$R(V) \supseteq \{Y^0, Y^1, \cdots, Y^a, X^0, X^1, \cdots, X^b\} \quad (0 \leq a, 0 \leq b) \tag{6.2}$$

この $R(V)$ を図 6.4 のように Top goal とする．
次に，6.2.1 節で述べた手法に従ってゴール分解を行う．なお，ゴール木のコメントにある変数はゴール分解を行う観点である．このゴール分解は，個々の Sub goal が，単一の入力もしくは出力変数を格納している状態の Sub goal になるまで，KAOS の下位ノードの論理積が上位ノードとなるゴール分解の意味を用いて分解する．さらに，Sub goal の単一の出力変数（6.1 の左辺）は，対象の検査モデルの特徴より，複数の入出力変数（6.1 の右辺の引数（変数））から導出されるため，本手法のオリジナリティである下位ノード（6.1 式の右辺の引数）の論理積が上位ノード（6.1 式の左辺）となるゴール分解の意味を用いて，単一の入力変数が格納される状態になるまで分解を繰り返す．最後に，単一の入力変数を含んだ Sub goal に対して，センサの Agent を接続させ，接続された Sub goal の全ての入力変数を抽出することで，Top goal の検査する性質に依存する入力変数（着目すべき入力変数）の抽出が可能となる．

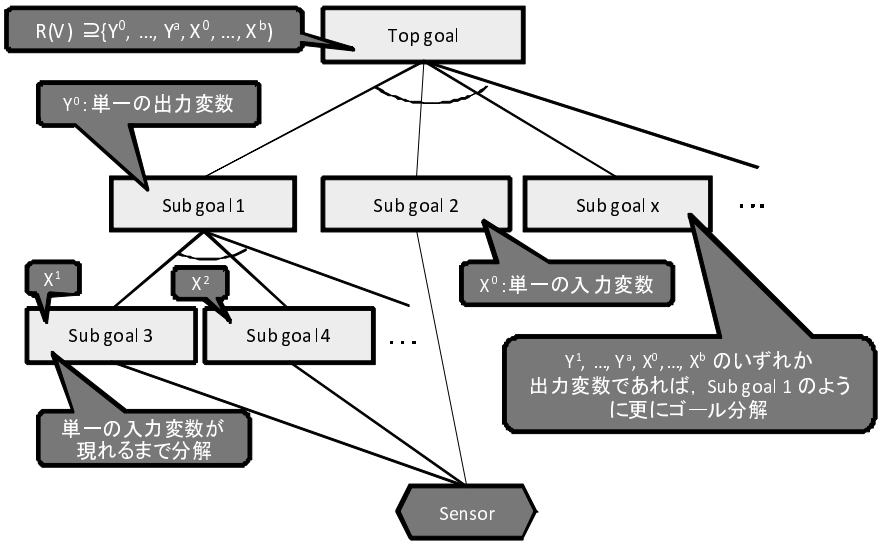

図 6.4: EIVP で使用するゴール指向分析の手順

具体的な例を示すため，7.2 節で述べるライントレーサの事例を記載する．まず Top goal を図 6.5 のように，検査する性質が「今回のセンサ値と閾値の差が 1T のセンサ値と閾値の差より小さければ，モータへのパワー値は 1T より必ず小さい」であることから，Top goal をその性質に表現されている入出力値が格納されている状態「今回と 1T のセンサ値および今回と 1T のモータへのパワー値をそれぞれ変数に格納できている.」とした．そして，センサ値またはモータへのパワー値を格納する変数を含んだ Sub goal に分解する．なお，4.2 節で述べ

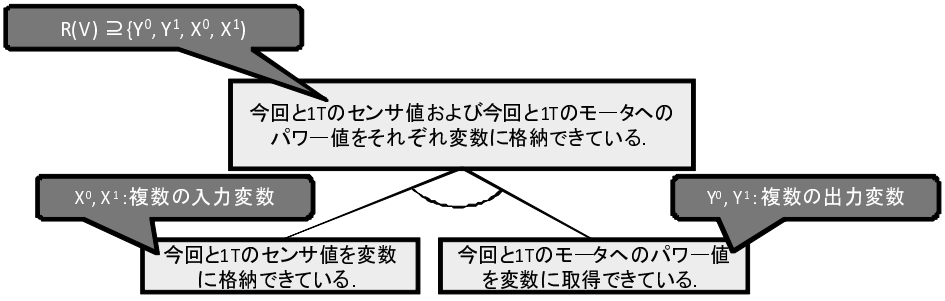

図 6.5: ライントレーサの事例の EIVP（その１）

たように，検査する性質の表現されている入力または出力値は，変数に格納され

ている状態として表現する．単一のセンサ値を格納する変数またはモータへのパワー値を格納する変数を含んだ Sub goal となるまで図 6.6 のようにゴール分解を繰り返す．ここで， Sub goal 「今回のセンサ値を変数に取得できている.」と「1T のセンサ値を変数に取得できている.」は，単一の入力変数のみを含んでいるため，ゴール分解を終了させる． 次に，単一の出力変数を含んだ Sub goal 「今

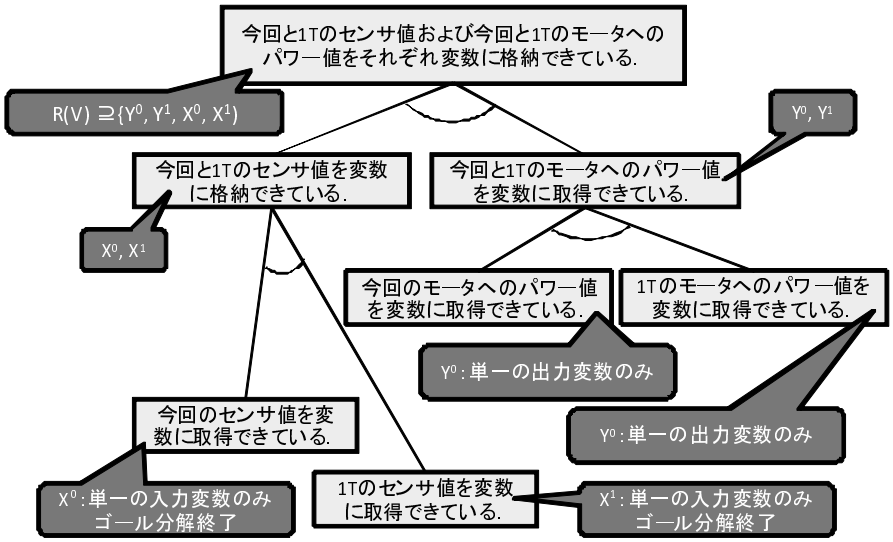

図 6.6: ライントレーサの事例の EIVP（その 2）

回のモータへのパワー値を変数に取得できている.」と「1T のモータへのパワー値を変数に取得できている.」は，下位ノード（6.1 式の右辺の引数）の論理積が上位ノード（6.1 式の左辺）とのゴール分解の意味を用いて，3.3 節の 3.6 式の検査モデル $Y^0 = f(X^0, X^1, X^2)$ であることから，それぞれ $Y^0 = f(X^0, X^1, X^2)$ と $Y^1 = f(X^1, X^2, X^3)$ となり，図 6.7 のような今回から 3T までの光センサ値を格納する変数が現れる（$X^0, \cdots, X^3$ に相当する） Sub goal への分解となる．最後に，図 6.8 のように単一の入力変数を含んだ Sub goal に対して，光センサの Agent を接続させ，接続されている Sub goal の入力変数を抽出することで， 着目すべき入力変数 $X^0, \cdots, X^3$ の抽出が行えた．

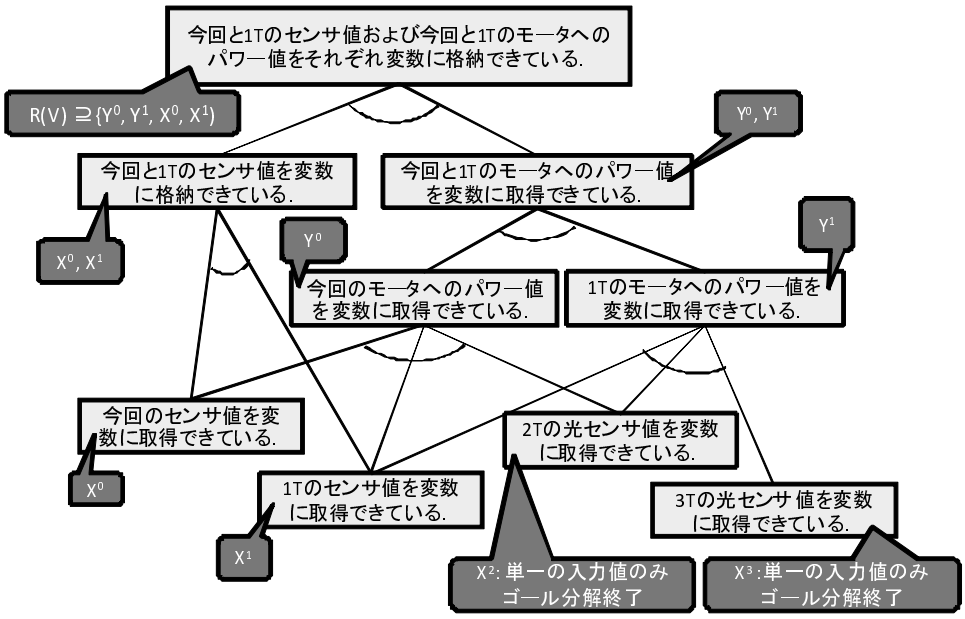

図 6.7: ライントレーサの事例の EIVP（その３）

6.3 考慮不要な外部環境の要因を排除するゴール指向分析手法（ GWEU ）

本節では問題２に対応した，考慮不要な外部環境の要因を排除したワーストケースシナリオを分析するゴール指向分析手法を提案する．この GWEU の成果物はワーストケースシナリオであり，プロトタイププログラムを動作させられる状況が明確になるまで，システムの振舞い要因と外部環境の要因の２つの観点，および「実測時の状況」と「初期状態から実測するまでの状況」に分けてゴール分解を行う手法である．まず，本手法で提案するゴール指向分析の記法を 6.3.1 節で述べ，次に本手法の３つの詳細な手順について 6.3.2 節〜6.3.4 節にて述べる．

6.3.1 GWEU の記法

本記法は KAOS のサブセットであり，図 6.9 に示す通り，KAOS の Top goal, Sub goal，AND 分解，OR 分解，そして障害の表記を用いる．また，本手法のオリジナリティである，ゴール分解時の外部環境の要因の散見を防ぐための，シス

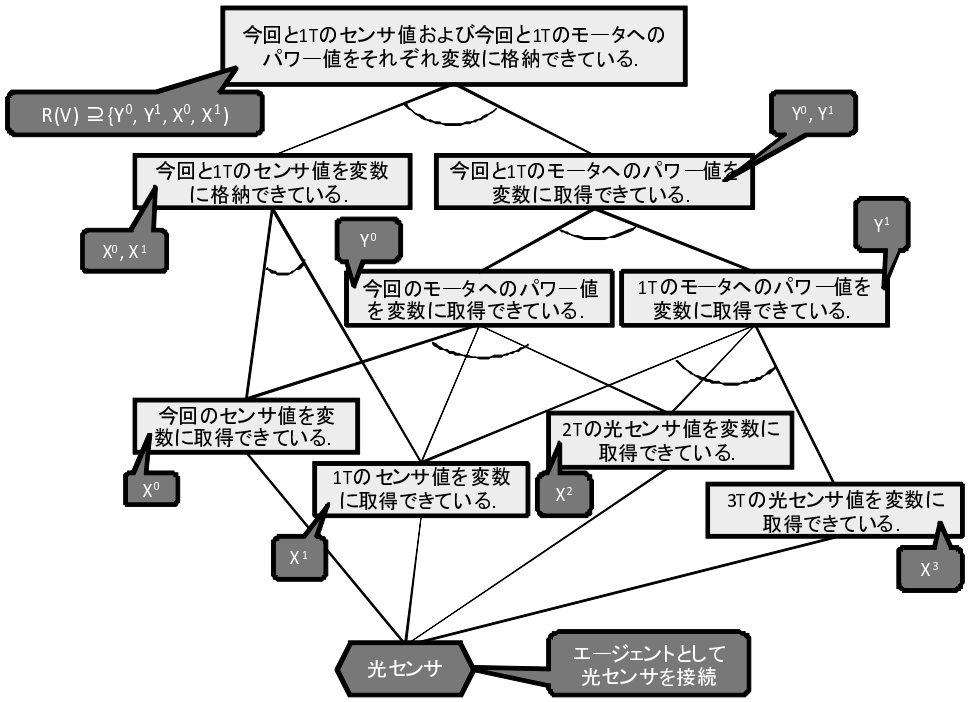

図 6.8: ライントレーサの事例の EIVP（その４）

テムの振舞いと外部環境の要因とを分類する点線の表記を用いる．まず，システムが動作する全てのユースケース上で，センサ値の時系列変化が最大となる状況を Top goal と表記する．次に，ワーストケースシナリオの一部分となる，センサ値が最大に変化する要因を Sub goal と表記する．また，同時に考慮する必要がないその要因間の関係を 障害 と表記する．最後に，点線による分解要因のグループ化を追加する．

6.3.2　手順１：センサ値の時系列変化が最大となる状況の分析

本節では図 6.2 に示す，センサ値の時系列変化が最大となる状況を分析する手順について述べる．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には，要求仕様書とドメイン知識を用いて，システムが動作するユースケース，特にその際の外部環境を分析する必要がある．その外部環境を分析するため，外部環境記述用 UML プロファイル [18] に対応させた表 6.1 を使用した８つの観点を

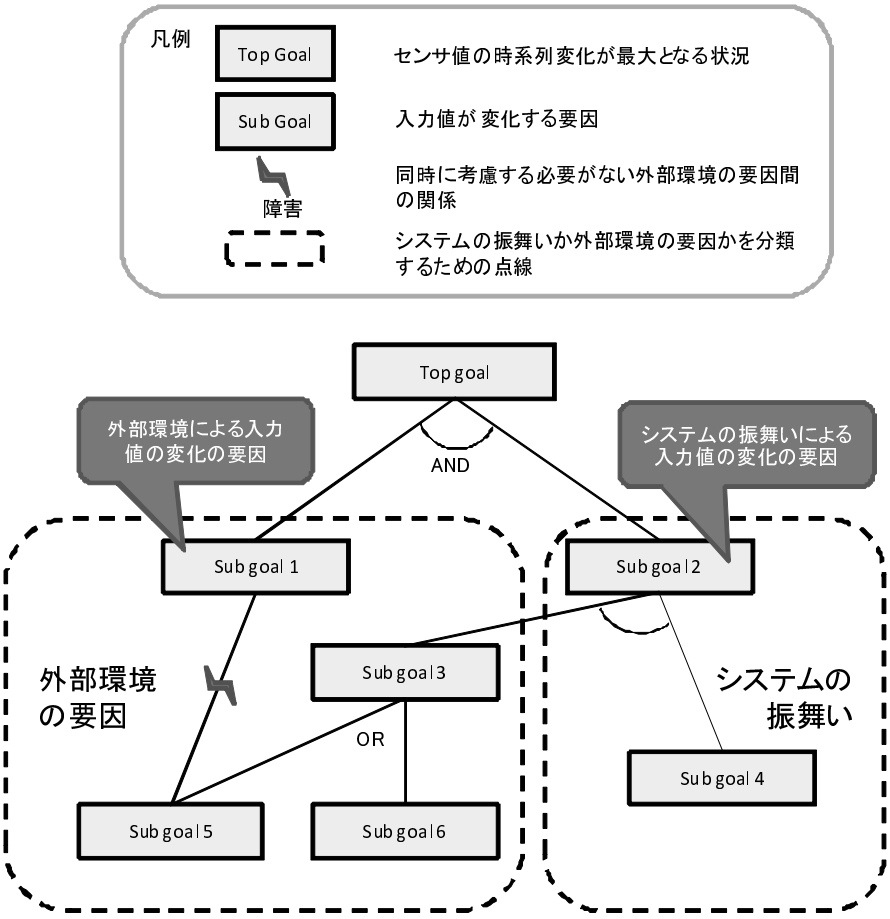

図 6.9: GWEU の記法

用いて，その状況を分析する．この UML プロファイルを分析する項目として用いることで，外部環境の分析を用いない場合と比べて容易になる．その表 6.1 の関係を図 6.10 に示す.

外部環境記述用 UML プロファイルは，３種類のクラスの拡張とする<<Context>>, <<Hardware>>，<<Software>>のステレオタイプと，５種類の関連の拡張とする <<Observe>>，<<Control>>，<<Noise>>，<<Transfer>>，<<Affect>>のステレオタイプで外部環境とシステムの関係をモデル化するためのものである．<<Observe>>は，ハードウェアが外部環境の要素を観測する関係を意味する．これは本研究では，EIVP で抽出された「着目すべき入力変数」に格納されるセンサ値の変化の関係に位置づけされる．<<Context>>は，組込みシステムの動作に影響する外部環境の要素を意味し，システムに対する外部からの入力やそれに影響を与える要

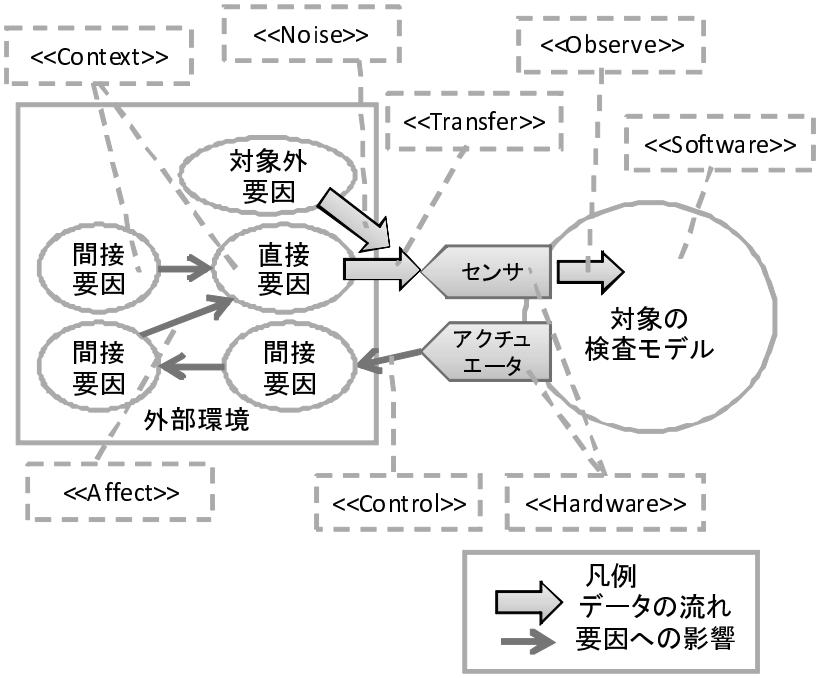

図 6.10: センサ値の時系列変化が最大となる状況と 8 つの分析観点について

素がこれに分類される．本研究では最も大きく変化するセンサ値に直接および間接的に影響する外部環境の要因に相当する．<<Transfer>>は，ハードウェアが外部環境の要素を観測する際に，直接的に観測することができず，別の要素に変換しなければならない場合があり，このときの要素同士の変換の関係を意味する．本研究では，センサ値に直接的に影響する外部環境の要因による，入力値の変化の関係に相当する．<<Noise>>は，ハードウェアが外部環境の要素を観測する際に，観測の対象外の要素が混ざった状態で入力されることがあリ，このときのハードウェアと観測の対象以外の要素との関係を意味する．本研究では，センサ値に間接的に影響する外部環境の要因による，入力値の変化の関係に相当する．<<Affect>>は，外部環境の要素が変換される際に，ハードウェアが観測したい要素とは別の要素が影響を与える場合があり，このときの変換後の要素と影響を与える要素の関係を意味する．本研究では，最も大きく変化するセンサ値に直接及び間接的にに影響する外部環境の要因間の関係に相当する．<<Control>>は，ハードウェアが外部環境の要素を制御する関係を意味する．本研究では，外部環境に

外部環境記述用UMLプロファイル			本研究
名前	適用する要素	定義	センサ値の時系列変化が最大となる状況を分析する観点
<<Observe>>	Association	ハードウェアが外部環境の要素を観測する関係を意味する.	旧VPで抽出された「着目すべき入力変数」に格納されるセンサ値の変化の関係
<<Context>>	Class	組込みシステムの動作に影響する外部環境の要素を意味する. システムに対する外部からの入力やそれに影響を与える要素がこれに分類される.	最も大きく変化するセンサ値に直接および間接的に影響する外部環境の要因
<<Transfer>>	Association	ハードウェアが外部環境の要素を観測する際に, 直接的に観測することができず, 別の要素に変換しなければならない場合がある. このときの要素同士の変換の関係を意味する.	センサ値に直接的に影響する外部環境の要因による, 入力変数の変化の関係
<<Noise>>	Association	ハードウェアが外部環境の要素を観測する際に, 観測の対象外の要素が混ざった状態で入力されることがある. このときのハードウェアと観測の対象以外の要素との関係を意味する.	センサ値に間接的に影響する外部環境の要因による, 入力変数の変化の関係
<<Affect>>	Association	外部環境の要素が変換される際に, ハードウェアが観測したい要素とは別の要素が影響を与える場合がある. このときの変換後の要素と影響を与える要素の関係を意味する.	最も大きく変化するセンサ値に直接及び間接的にに影響する外部環境の要因間の関係
<<Control>>	Association	ハードウェアが外部環境の要素を制御する関係を意味する.	外部環境に影響する出力変数の変化の関係
<<Hardware>>	Class	組込みシステムを構成するハードウェアを意味する.	センサとアクチュエータ
<<Software>>	Class	組込みシステムを構成するソフトウェアを意味する.	検査対象の設計モデル

表 6.1: 本分析項目の外部環境記述用 UML プロファイル対応表

影響する出力値の変化の関係に相当する．最後に，<<Hardware>>と<<Software>>は，それぞれ組込みシステムを構成するハードウェアとソフトウェアを意味する．本研究ではそれぞれ，センサおよびアクチュエータと対象の検査モデルに相当する．

ライントレーサの事例では，要求仕様書およびドメイン知識に加えて，図 6.11 に示す８つの分析観点で，センサ値の時系列変化が最大となる状況を「フルス

ピードで急カーブを曲がる」と想定した．その８つの分析観点は次の通りであ

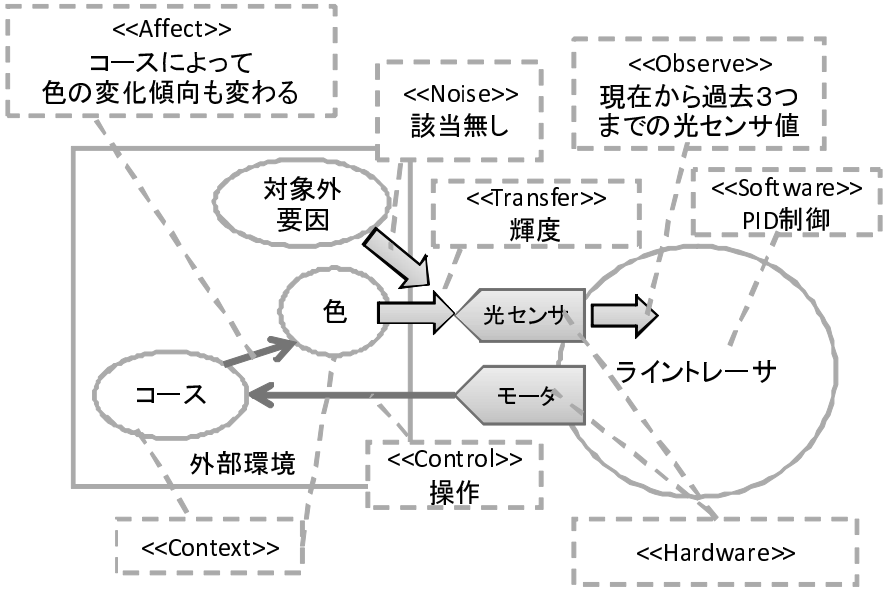

図 6.11: ライントレーサの事例での８つの分析観点

る．<<Observe>>は 6.2.2 節の例で述べたすべてのセンサ値 $X^u, \cdots, X^{u-3}$（現在から 1T,2T,3T の全ての値），<<Context>>は色とコース，<<Transfer>>は輝度，<<Noise>>は該当無し，<<Affect>>はコースによって色の変化傾向も変わる，<<Control>>はモータによる操作，そして最後に<<Hardware>>と<<Software>>は光センサおよびモータと PID 制御であるとした．

6.3.3　手順２：２つの観点によるゴール分解

本節では，手順１で分析した，センサ値の時系列変化が最大となる状況を用いて，ワーストケースシナリオを分析する手順について述べる．ワーストケースシナリオは，4.3 節で述べた通りセンサ値が変化する要因（システムの振舞いによる要因と外部環境による要因）をシナリオ化して表現したものであり，センサ値の時系列変化が最大となる状況に対して，システムの内部状態と，場所やタイミングを含む外部の状況を分析する必要がある．そこで，まず手順１で分析したセンサ値の時系列変化が最大となる状況を図 6.12 のように Top goal とし，Sub

goal1 および Sub goal2 のように，センサ値が変化する要因を，システムの振舞いによる要因と外部環境による要因の 2 つに分解する．

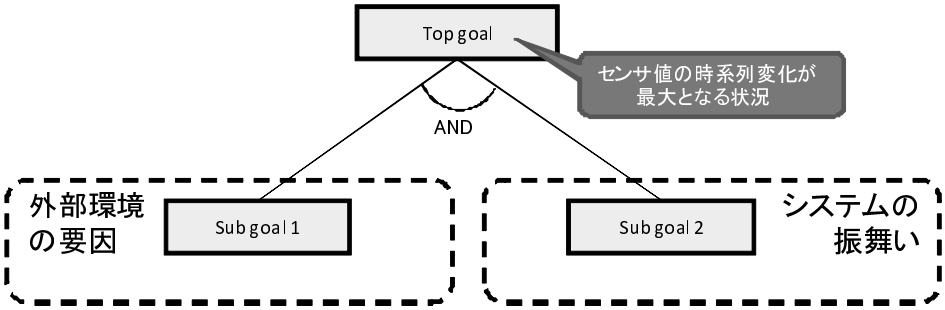

図 6.12: GWEU 手順２（その１）

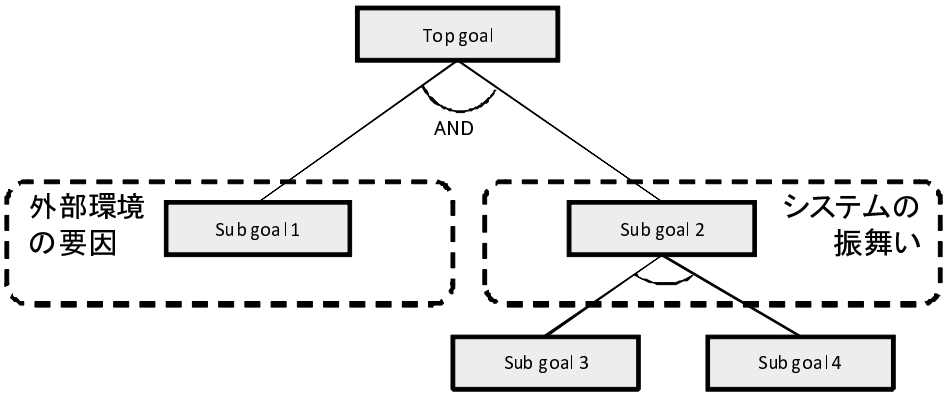

図 6.13: GWEU 手順２（その２）

このゴール分解は，プロトタイププログラムを動作させられる状況が明確になるまで，２つの観点でゴール分解と点線によるグルーピングを繰り返す．また，プロトタイププログラムでパラメータ値が必要であれば，そのパラメータが用いられて走行している状況までゴール分解を行う．２つの観点で分解を繰り返す理由として，外部環境の要因およびシステムの振舞いである Sub goal は，どちらの要因であっても，さらにゴール分解を行うと外部環境の要因またはシステムの振舞いの要因となる Sub goal が現れるためである．例えば，図 6.13 のように Sub

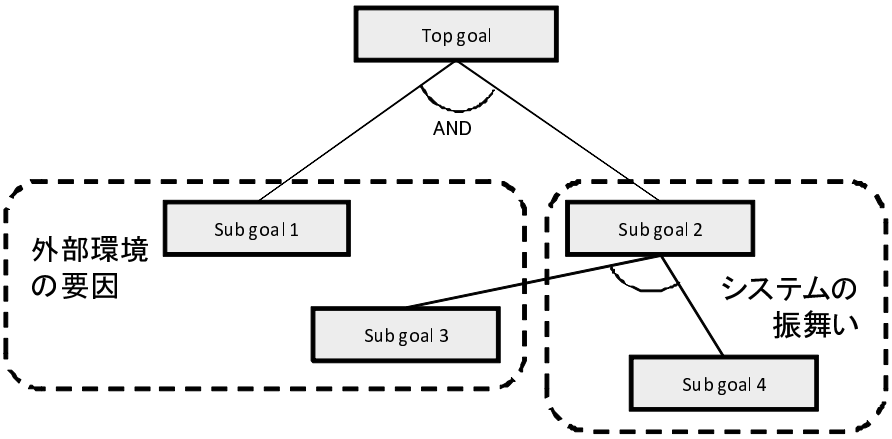

図 6.14: GWEU 手順２（その３）

goal 2 をさらに Sub goal 3 と Sub goal 4 にゴール分解を行ったとする．ここで図 6.14 のように，Sub goal 3 が外部環境の要因で，Sub goal 4 がシステムの振舞いであった場合，それぞれの観点のグルーピングを行う． さらに，Sub goal 3

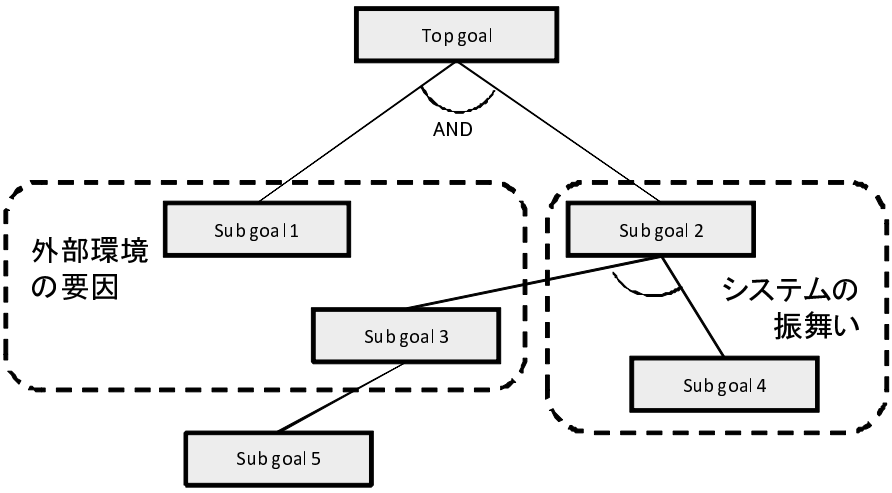

図 6.15: GWEU 手順２（その４）

にパラメータの抽出が必要である場合は，さらに図 6.15 のように Sub goal 5 へのゴール分解を行い，図 6.16 のように外部環境の要因としてグルーピングを行う．結果として，センサ値の時系列変化が最大となる状況から，外部環境の要因とシステムの振舞いの２つの観点にてゴール分解することで，プロトタイププロ

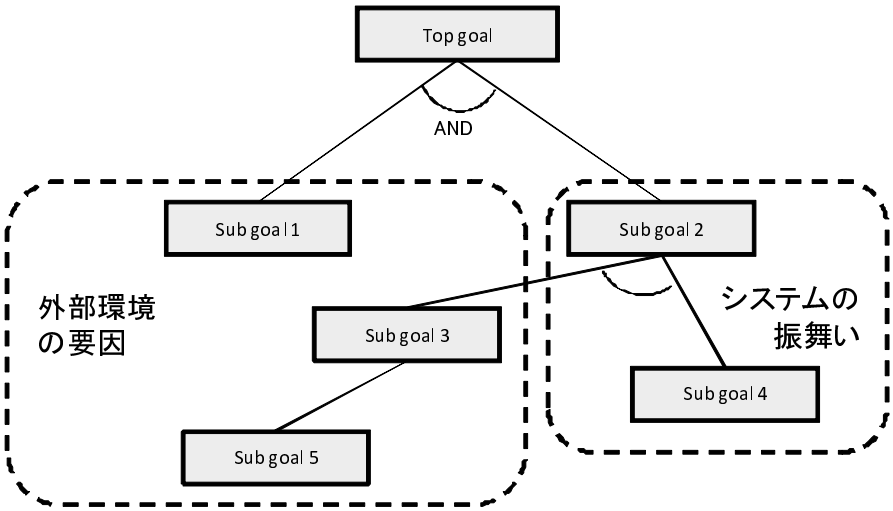

図 6.16: GWEU 手順２（その５）

グラミングで使用するパラメータを含めた具体的な要因の収集が行える．

ライントレーサの事例では，手順１により分析したセンサ値の時系列変化が最大となる状況「フルスピードで急カーブを曲がる」を Top goal とし，その状況に含まれる光センサ値が変化する要因を，図 6.12 のようにシステムの振舞いと外部環境の要因「急カーブを曲がる」および「フルスピードで走行する」に分解した．ここで，Sub goal「急カーブを曲がる」はコースに依存しており，実測に

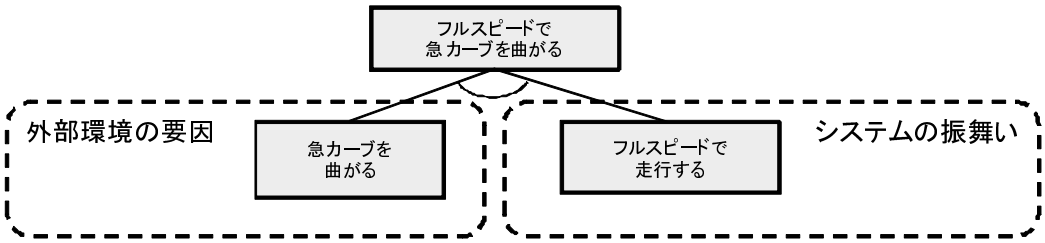

図 6.17: GWEU 手順２（ライントレーサその１）

必要な状況が現れたためゴール分解を終了した．次に Sub goal「フルスピードで走行する」をゴール分解すると，図 6.18 のように「坂道を下る」「PID 制御で走行する」「フルバッテリ残量で走行する」の状態がフルスピードになると分析した．このゴール分解を行う際にドメイン知識を用いた．フルスピードで走行するためには，坂道を下るときの加速と，ラインをトレースする際の左右の蛇行に

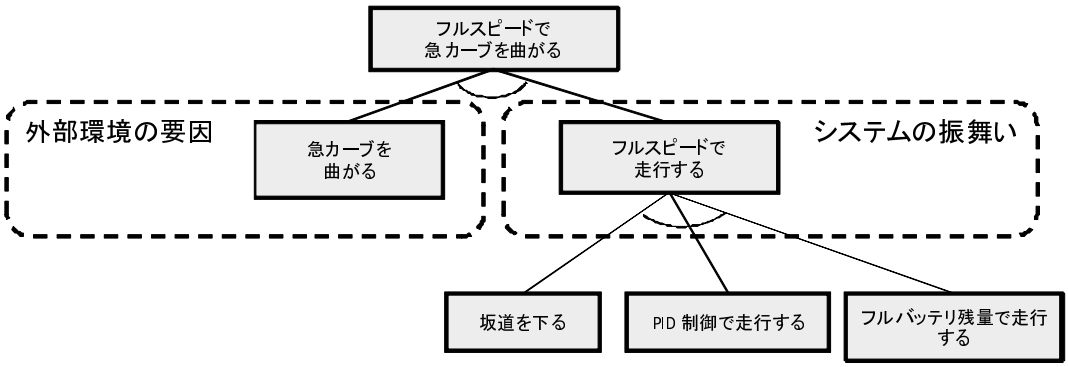

図 6.18: GWEU 手順２（ライントレーサその２）

よる速度低減に対した PID 制御による軽減と，バッテリ残量が最大によるモータのパワー最大による速度向上が必要であると検討した．

次に図 6.19 のように，Sub gaol「PID 制御で走行する」がシステムの振舞い，Sub gaol「坂道を下る」と「フルバッテリ残量で走行する」が外部環境の要因の観点であったため，それぞれの観点毎にグルーピングを行った． さらに図 6.20

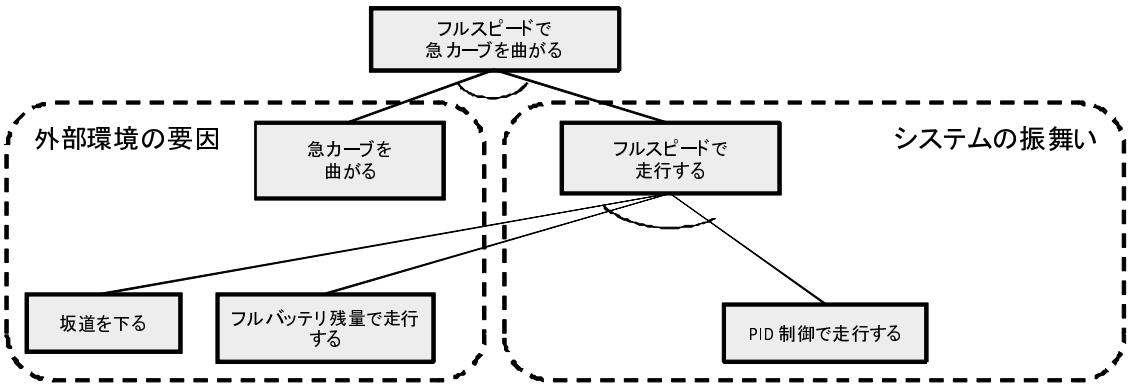

図 6.19: GWEU 手順２（ライントレーサその３）

のように，それぞれの Sub goal を実測に必要な状況が現れるまでゴール分解を行い，図 6.21 のように，それぞれの Sub goal のグループに追加を行った．なお，PID 制御のパラメータは PID の設計段階で決定されている値であり，本手法の分析によって左右される値ではない．結果として，センサ値の時系列変化が最大となる状況「フルスピードで急カーブを曲がる」から，外部環境の要因とシステ

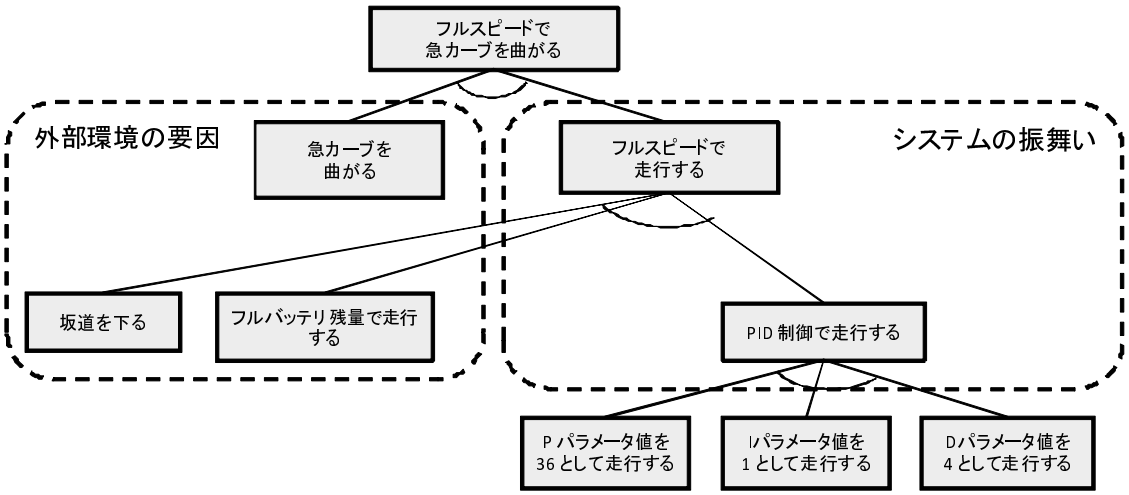

図 6.20: GWEU 手順２（ライントレーサその４）

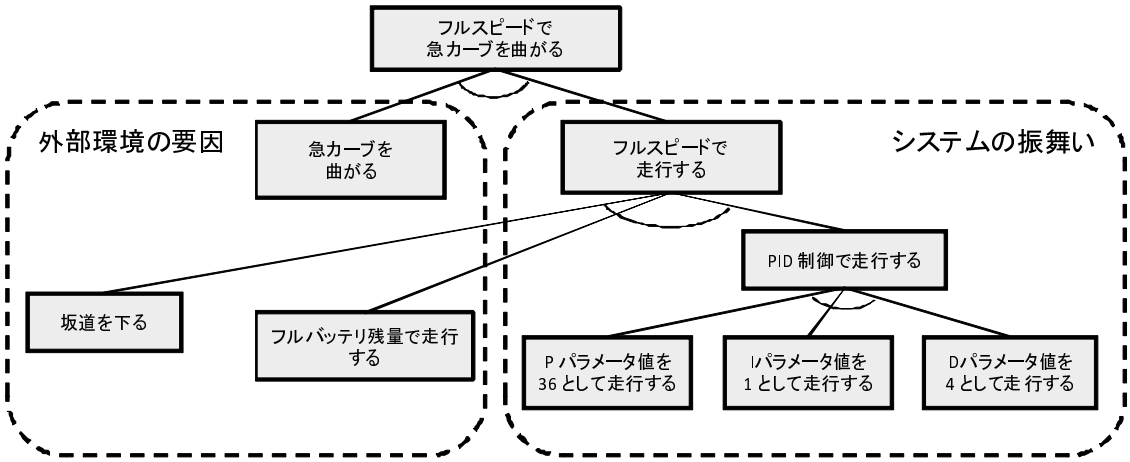

図 6.21: GWEU 手順２（ライントレーサその５）

ムの振舞いの２つの観点にてゴール分解することで，プロトタイププログラミングで使用する「PID 制御で走行する」「P パラメータ値を 36 として走行する」「I パラメータ値を 1 として走行する」「D パラメータ値を 4 として走行する」「坂道を下る」「フルバッテリ残量で走行する」「急カーブを曲がる」とのワーストケースシナリオの抽出が可能となった.

6.3.4　手順３：不要な外部環境の要因の排除

次に分解された外部環境の要因の中で，考慮不要な要因が存在するかを検討する．考慮不要かどうかは，手順２で分析されグルーピングされている詳細な外部環境の要因から，要求仕様書やドメイン知識を用いて，実際に発生しうるユースケースがあるかどうかで判断する．もし図 6.16 の Sub goal 1 と Sub goal 5 の関係が該当（実際のユースケース上では発生しないと）すれば，図 6.22 に示すように，その Sub goal 間に線を接続し「障害」の表記を付与する．この障害が現れる条件は，5.2 節で述べた内容から下記とする．

- 「実測時の状況」から考慮不要な要因が抽出できた場合.
- 「初期状態から実測までの状況」から考慮不要な要因が抽出できた場合.

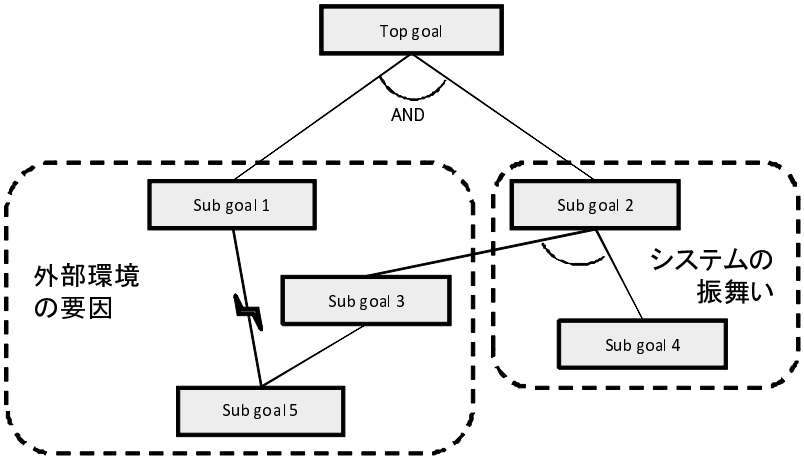

図 6.22: GWEU 手順２（その６）

また，図 6.23 のように，Sub goal 5 の代替えとなる Sub goal 6 を Sub goal 3 の OR 分解に追加する．この結果，Top goal のセンサ値の時系列変化が最大となる状況から，考慮不要な外部環境の要因 Sub goal 5 の抽出と，その要因を排除した Sub goal 1，Sub goal 4，そして Sub goal 6 の要因によるワーストケースシナリオの分析が可能となる．

ライントレーサの事例では，図 6.21 の外部環境の要因の中で，考慮不要な要因が存在するかを検討した．その結果,「実測時の状況」において「障害」はな

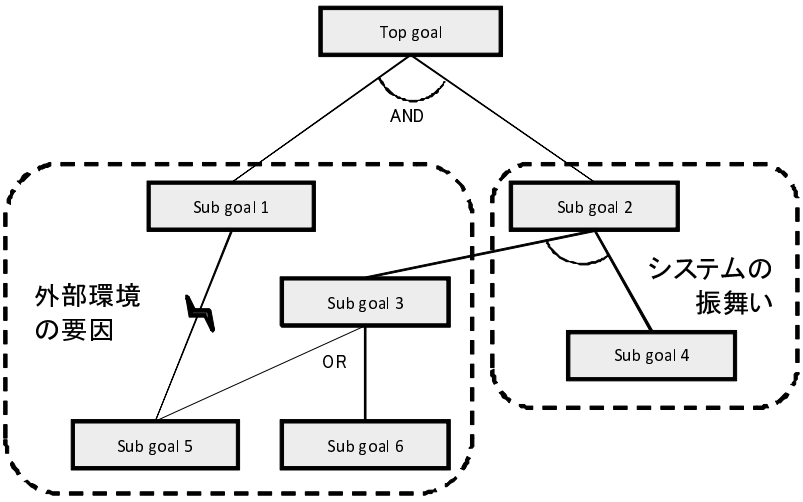

図 6.23: GWEU 手順２（その７）

かったが,「初期状態から実測までの状況」の考慮において，コースのスタート地点から坂道までには距離があり，実際は少しバッテリを消費した状態となる事から，フルバッテリ残量で坂道を下る条件は考慮に入れる必要がないと分析した．そこで，図 6.24 の通りに 「坂道を下る」ためには「坂道までバッテリを消費する」との Sub goal を追加し,「フルバッテリ残量で走行する」の間に「障害」の表記を付与した．また，図 6.25 のように,「フルバッテリ残量で走行する」の代替えで,「消耗したバッテリ残量（95%）で走行する」を OR 分解として追加した．この結果，Sub goal「フルバッテリ残量で走行する」の考慮不要な外部環境の要因の抽出が行え，その要因を排除した「フルスピードで急カーブを曲がる」の状況に対して,「PID 制御で走行する」「P パラメータ値を 36 として走行する」「I パラメータ値を 1 として走行する」「D パラメータ値を 4 として走行する」「坂道を下る」「坂道までバッテリを消費する」「消耗したバッテリ残量（95%）で走行する」「急カーブを曲がる」の要因によるワーストケースシナリオの分析が行えた．

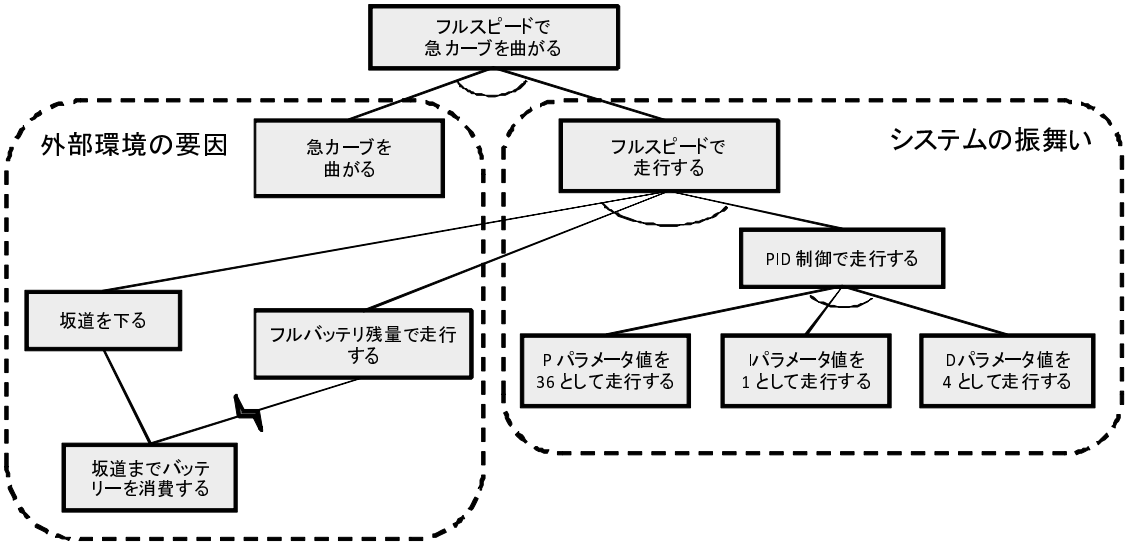

図 6.24: GWEU 手順２（ライントレーサその６）

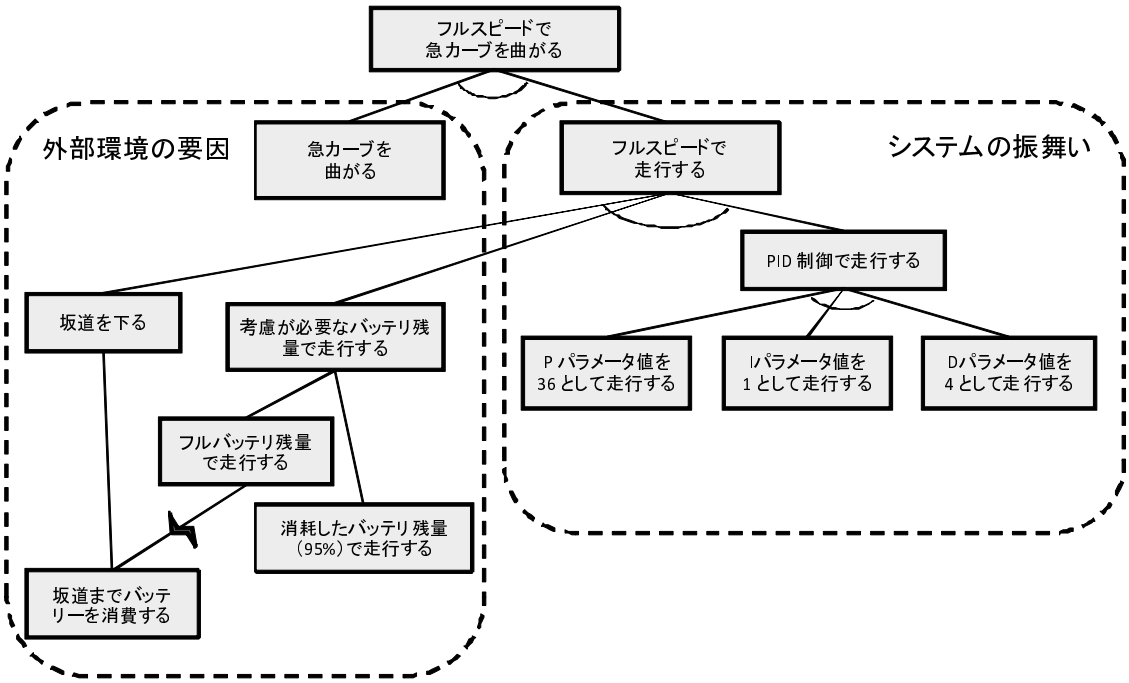

図 6.25: GWEU 手順２（ライントレーサその７）

第 7章

話題沸騰ポットとライントレーサの事例を用いた評価

本提案手法は，設計モデルや検査する性質を入力とし，EIVP を用いて入力変数の集合を抽出，GWEU を用いてその集合に関係する外部環境を考慮してワーストケースシナリオを分析，そして入力値の変化幅を限定する手法である．そのため，ドメイン（外部環境）が異なる設計モデルでの効果の確認と，同じ設計モデルで検査する性質が異なる場合の効果の確認，ワーストケースシナリオ分析の妥当性の確認，そして外部環境を限定してモデル検査を行う手法 [21] との比較について評価する．ここでの外部環境が異なる設計モデルの事例として，組込みソフトウェア管理者・技術者育成研究会（SES-SAME）[50] の「話題沸騰ポット」と，ET ロボコン [28] で用いられる「ライントレーサロボット」（ライントレーサ）の２つの設計モデルを例に挙げる．

まず，7.1 節にて話題沸騰ポットの事例を述べる．この事例は，7.1.1 節の要求仕様記載の通りサーミスタの精度が小数点第一位であることから，状態爆発が発生しやすい事例である．次に，7.2 節にてライントレーサの事例を述べる．この事例はコースの難所が複数あり，センサ値の時系列変化に影響を及ぼす様々な外部環境の要因が存在する事例である．次に 7.3 節にて，外部環境を限定してモデル検査を行う他研究と本手法の比較を両事例を用いて行う．次に 7.4 節にて，両事例への本提案手法の適用結果から，ワーストケースシナリオ分析の妥当性や状態削減の効果の確認を含めた本研究の評価を行う．そして最後に，複数のセン

サ・アクチュエータを持つ検査モデルを対象とした場合の，本手法の有効性について 7.5 節にて述べる．

7.1 話題沸騰ポットの事例

本節では，7.1.1 節にて話題沸騰ポットの仕様と設計モデルについて述べ，7.1.2 にて話題沸騰ポットの検査モデルについて述べ，7.1.3 節と 7.1.4 節にて提案手法の効果確認を行うため図 6.1 の流れに沿って本提案手法の適用を行う．なお，7.1.3 節での検査する性質は「今回の水温と 2T の水温が異なれば，ヒータの熱量は 2T より必ず大きい」とし，7.1.4 節では「今回と 1T と 2T の水温がそれぞれ異なれば，ヒータの熱量は 4T のものより必ず大きい」とした場合の効果の確認を行う．

7.1.1 話題沸騰ポットの仕様と設計モデル

「話題沸騰ポット」の仕様のうち，以下のみを考慮するものとする．

- 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 の範囲で小数第一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 制御を用いて，ヒータの出力値 (W 数) を操作し，水温の目標温度への制御を行う．
- 水位センサが off になると加熱を停止する．

次に設計モデルとして，システム構成図を図 7.1 に示し，状態遷移図を図 7.2 に示し，そしてシーケンス図を図 7.3 に示す．
図 7.1 のシステム構成図において，システムにはフィードバック制御を行うコントローラ部，蓋の開閉を管理するモジュール，保温設定ボタンを管理するモジュール，センサのサーミスタを制御するモジュール，そしてアクチュエータのヒータ

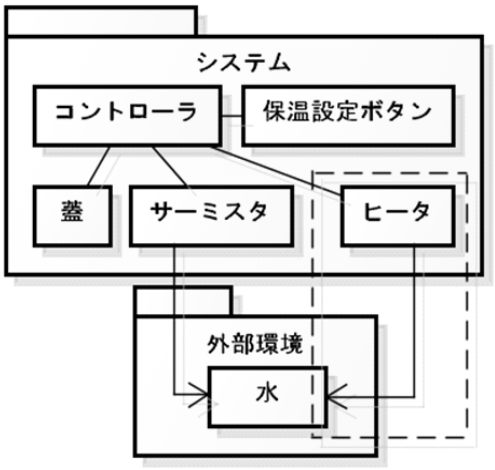

図 7.1: 話題沸騰ポットの設計のシステム構成図

を制御するモジュールが存在する．外部環境としてモデル化の対象となるのは水のみであり，ヒータで加熱された結果が水温に反映される．なお，サーミスタから値を取得する周期は 670ms とする．

7.1.2 話題沸騰ポットの検査モデル

この対象の検査モデルの状態遷移は，図 7.2 のように，ステートタスク，蓋監視タスク，ヒータ制御タスク，保温設定ボタン監視タスクが存在する．そのため，付録の検査モデル A.5 のように，外部環境記述として温度生成を行う処理 (5 行目から)，システム記述として，ステートタスク (58 行目から)，ヒータ制御タスク (121 行目から)，蓋監視タスク (177 行目から)，保温設定ボタン監視タスク (199 行目から)，そして温度操作量を算出する処理 (99 行目から) が存在する．しかしながら，検査する性質は「今回の水温と 2T の水温が異なれば，ヒータの熱量は 2T より必ず大きい」であるため，これに関連した部分の抽象化（2.1.1 節で述べたプログラムスライシングに類似した抽象化）を行った結果が，付録の検査モデル A.6 である．本事例は，この検査モデルを使用して評価を行う．

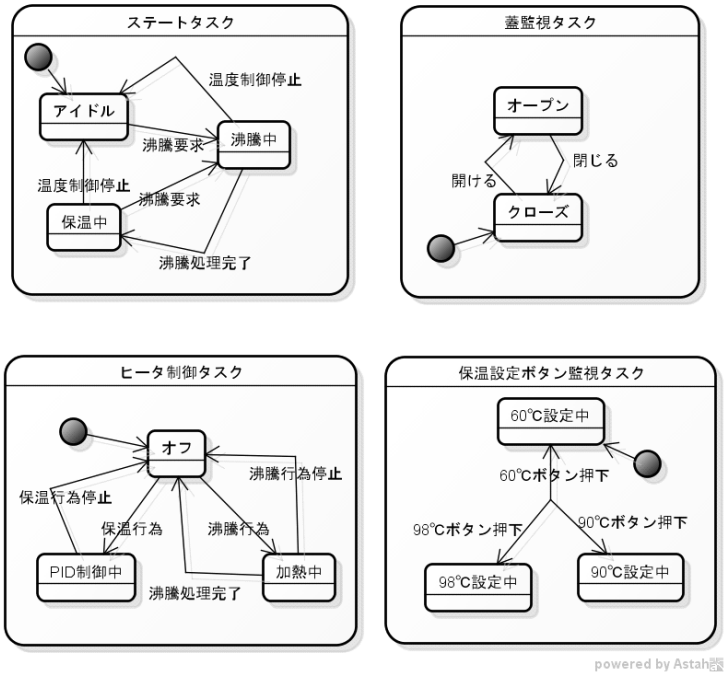

図 7.2: 話題沸騰ポットの状態遷移図

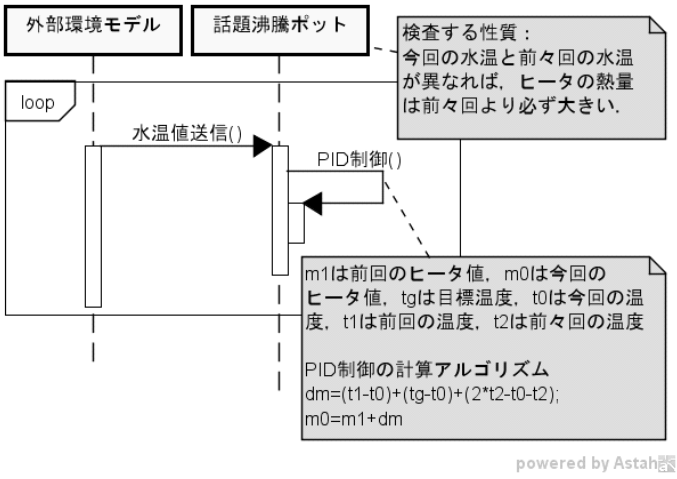

図 7.3: 話題沸騰ポットのシーケンス図

7.1.3 話題沸騰ポットへの適用

提案手法の効果確認を行うため，図 6.1 の流れに沿って，まず EIVP の適用を行い，次に GWEU の適用を行う．

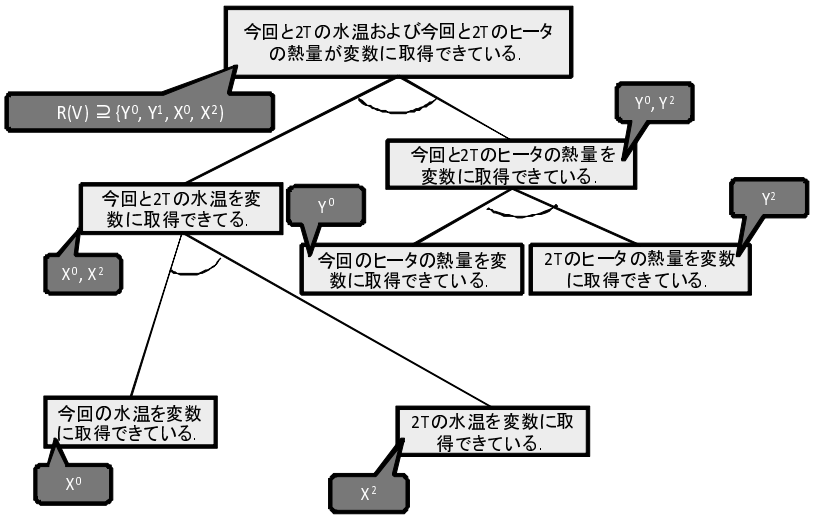

図 7.4: 話題沸騰ポットの EIVP 適用（その１）

検査する性質に依存する入力変数（“着目すべき入力変数”）を抽出する EIVP では，まず，検査する性質が「今回の水温と 2T の水温が異なれば，ヒータの熱量は 2T より必ず大きい」であることから，その性質に表現されている入出力値が格納されている状態「今回と 2T の水温および今回と 2T のヒータの熱量が変数に取得できている.」とした．次に，単一の入力もしくは出力変数を含んだ Sub goal へとゴール分解を行い，その結果が図 7.4 となった．次に，分解された単一の出力変数を含んだ Sub goal を，単一の入力変数を含んだ Sub goal へとゴール分解を行い，その結果が図 7.5 となった．これらの結果から，着目すべき入力変数は，現在から 1T,2T,3T,4T までの履歴のサーミスタ値を格納する入力変数 $X^0, \cdots, X^4$ であることを抽出した．

EIVP により抽出した入力変数，要求仕様書，そしてドメイン知識から，ワーストケースシナリオを分析する GWEU では，まず手順１において，話題沸騰ポットの８つの分析観点を検討した．その結果は図 7.6 である．ここから，現在から 1T,2T,3T,4T までの履歴のサーミスタ値の変化が最大となる状況を，要求仕様およびドメイン知識さらに図 7.6 の分析観点から想定した．その結果，サーミスタ値の変化幅が最大となる状況は，図 7.6 の水位と水温の関係から，水位が少ないと同じ熱量でも温度が上昇する速度があがるため,「少量の水での最大加

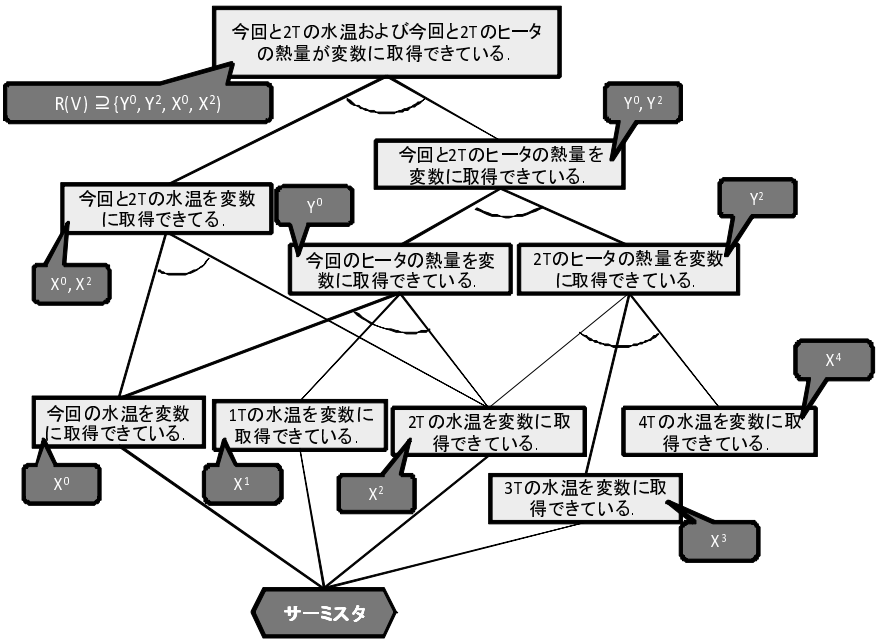

図 7.5: 話題沸騰ポットの EIVP 適用（その２）

熱を行う」と想定した.

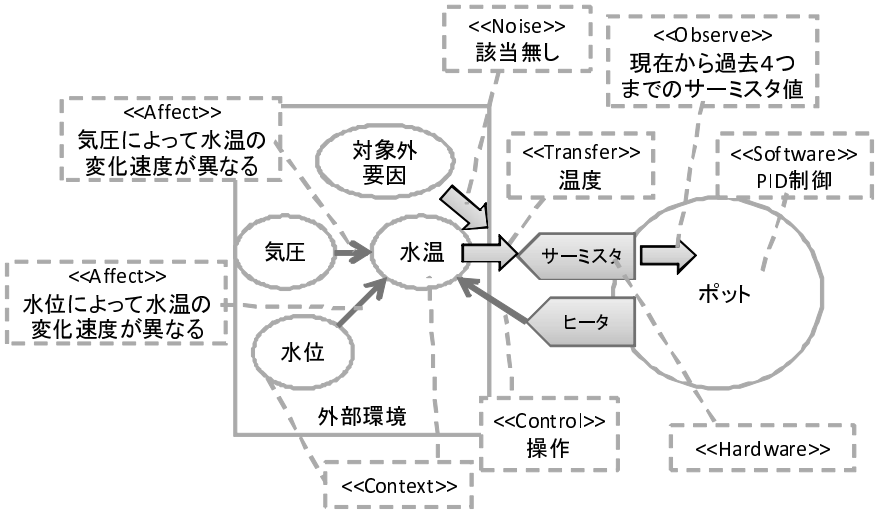

図 7.6: ８つの分析観点（話題沸騰ポットの事例）

次に手順２において，ワーストケースシナリオのゴール分解を行った．Top goal

の分解を図 7.7 に，外部環境の要因とシステムの振舞いの２つの観点にてゴール分解を行いそれぞれを点線で囲み，さらにゴール分解とグルーピングを行い，そして再度ゴール分解をプロトタイププログラミングが必要とするパラメータが抽出されるまで行った結果を，図 7.8〜図 7.10 までにそれぞれ示す.

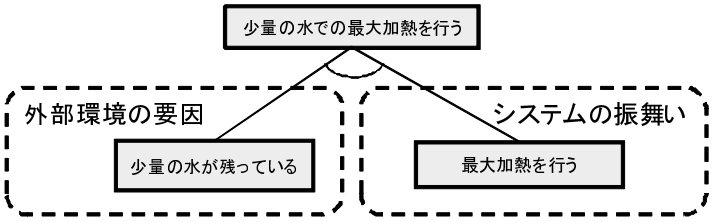

図 7.7: GWEU 手順２（ポットその１）

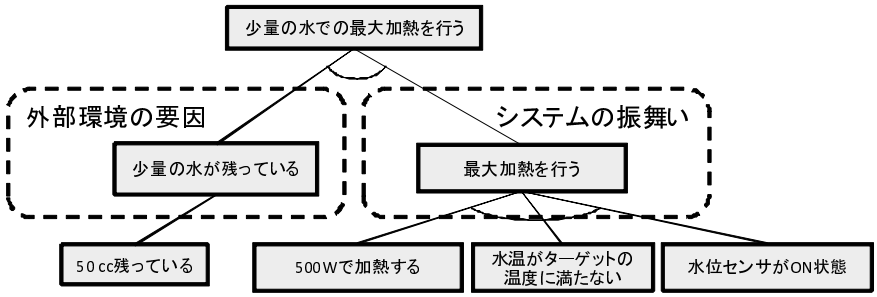

図 7.8: GWEU 手順２（ポットその２）

最後に手順３において，外部環境の要因の点線内にある Sub goal 間の関係で，考慮不要なものを分析した．「実測時の状況」において，実際の水位センサは水量「50cc 残っている」では水位センサが off 状態となり加熱できないため 障害 の表記で記載し，水位センサが on 状態となる水量「100cc 残っている」を代替えとして追加した結果を，図 7.11 と図 7.12 にそれぞれ示す．これらの結果，考慮不要な外部環境の要因「50cc 残っている」を抽出が行え，その要因を排除した「少量の水での最大加熱を行う」の状況に対して,「100cc 残っている」「水位センサが ON 状態」「ターゲットの温度に満たない（97°C)」「500W で加熱する」との要因によるワーストケースシナリオの分析が行えた.

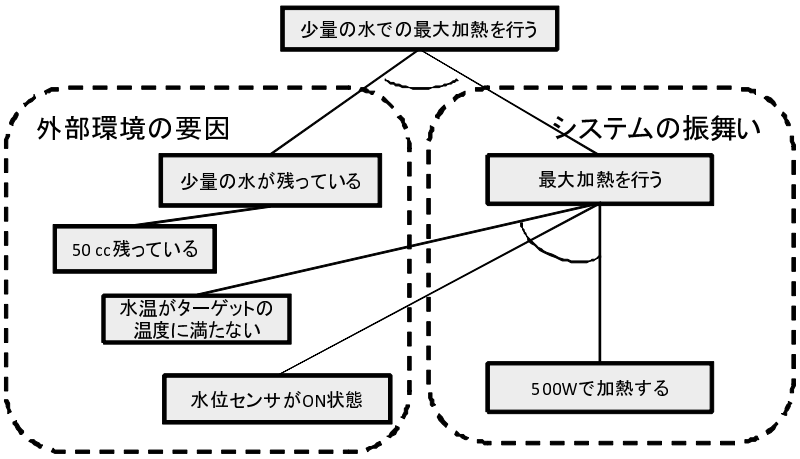

図 7.9: GWEU 手順２（ポットその３）

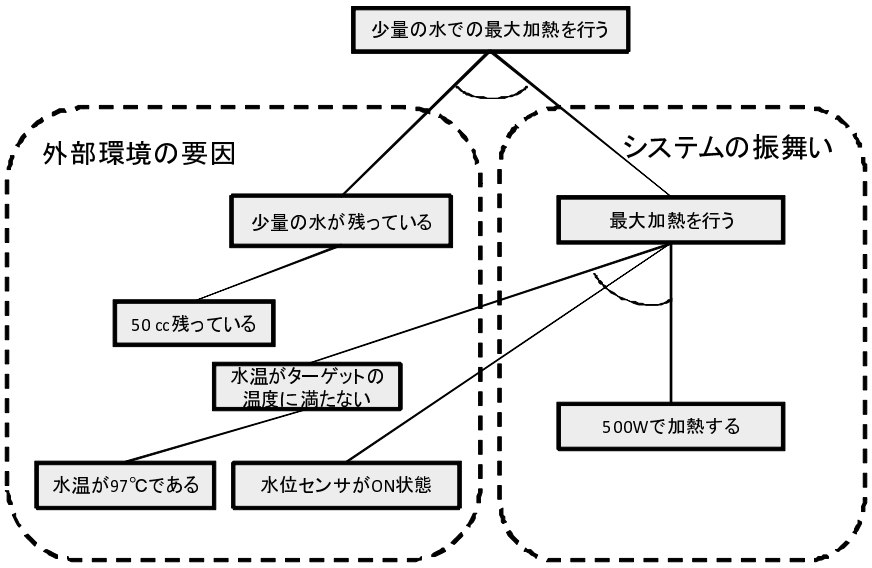

図 7.10: GWEU 手順２（ポットその４）

次に，その状況について 4.2 節の図 4.2 にある，プロトタイププログラミングを用いた時系列の最大変化幅の実測と，時系列変化の制約の抽出を行った結果，時系列変化の制約は約 1.0°C である．[1] また，排除しない（「50cc 残っている」を用いた）場合は，時系列変化の制約は約 2.0°C である．

[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（周期 s）／ 100（g）

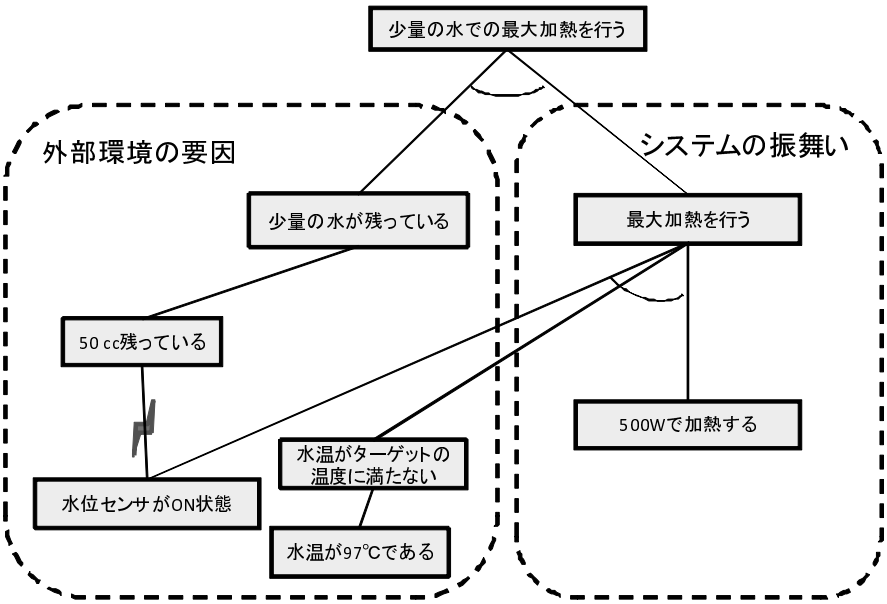

図 7.11: GWEU 手順２（ポットその５）

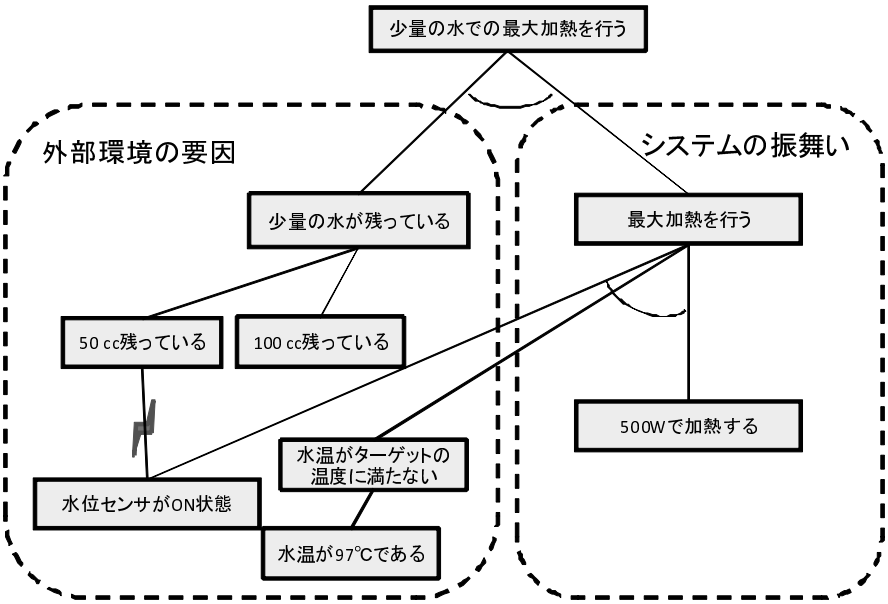

図 7.12: GWEU 手順２（ポットその６）

それらの時系列変化の制約を基に，付録の検査モデル A.6 に対して，排除した場合（「100cc 残っている」を用いた時系列変化の制約が 1.0°C の場合）の付録の検査モデル A.8 と，排除しなかった場合（「50cc 残っている」を用いた時系列変

化の制約が 2.0°C の場合）の付録の検査モデル A.7 とでモデル検査結果を比較した．その結果，状態数はそれぞれ 153921 と 1044977 であり，約 85% の削減が行えた．

7.1.4 異なる性質での話題沸騰ポットへの適用

一方，検査する性質を仮に「今回と 1T と 2T の水温がそれぞれ異なれば，ヒータの熱量は 4T より必ず大きい」とした場合，つまり，検査する性質に含まれる変数を 7.1 式から 7.2 式に変化させた場合の性質について EIVP を適用する．

$$R(V) \supseteq \{Y^0, Y^2, X^0, X^2\} \tag{7.1}$$

$$R(V) \supseteq \{Y^0, Y^4, X^0, X^1, X^2\} \tag{7.2}$$

ここで，V を検査する性質，$R(V)$ を V に関連する入出力変数の集合，出力変

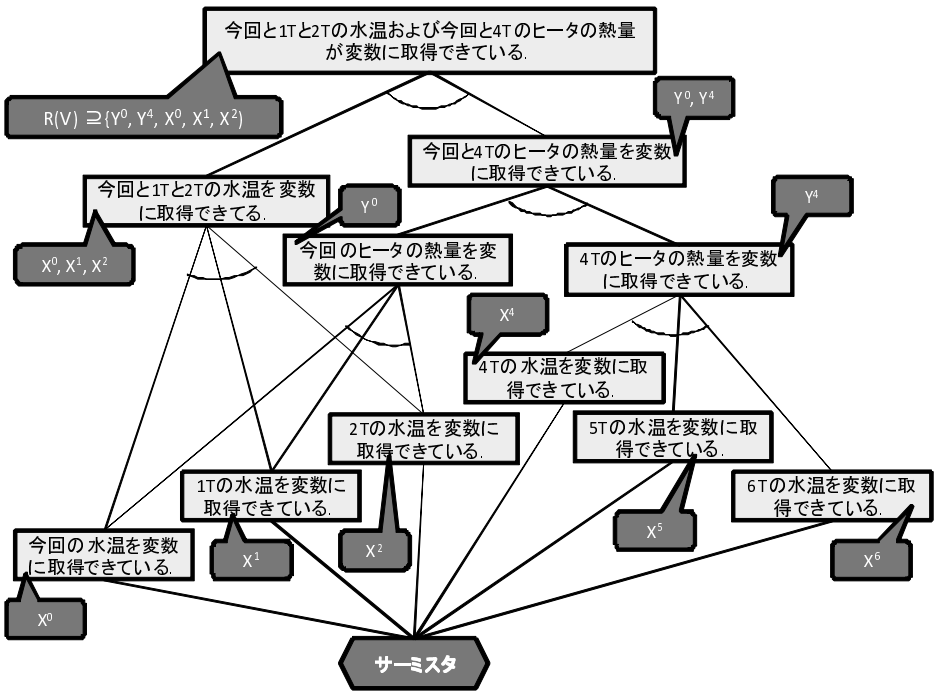

図 7.13: 話題沸騰ポットの EIVP 適用（異なる性質）

数をY，入力変数Xとする．その結果，図7.13であり，今回から2Tと4Tから6Tまでの履歴のサーミスタ値を格納する入力変数$X^0, \cdots, X^2$と$X^4, \cdots, X^6$であることを抽出した．また，GWEUの適用において，着目すべき入力変数は異なるがワーストケースシナリオの分析結果は同じであったため，7.1.4節と同じ結果が得られた．

7.2　ライントレーサの事例

本節では，7.2.1節にてライントレーサの仕様と設計モデルについて述べ，7.2.2節にてライントレーサの検査モデルについて述べ，7.1.3節と7.2.4節にて提案手法の効果確認を行うため図6.1の流れに沿って本提案手法の適用を行う．なお，7.2.3節での検査する性質は「今回のセンサ値と閾値の差が1Tのセンサ値と閾値の差より小さければ，モータへのパワー値は1Tより必ず小さい」とし，7.2.4節では「1Tと5Tのセンサ値が異なれば，1Tから3Tまでのモータへのパワー値はぞれぞれ異なる.」とした場合の効果の確認を行う．

7.2.1　ライントレーサの仕様と設計モデル

本提案手法を説明するための事例として，ETロボコン[28]で用いられる「ライントレーサロボット」（ライントレーサ）を例に挙げる．ライントレーサを図7.14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にてタイヤを制御する．
- 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．

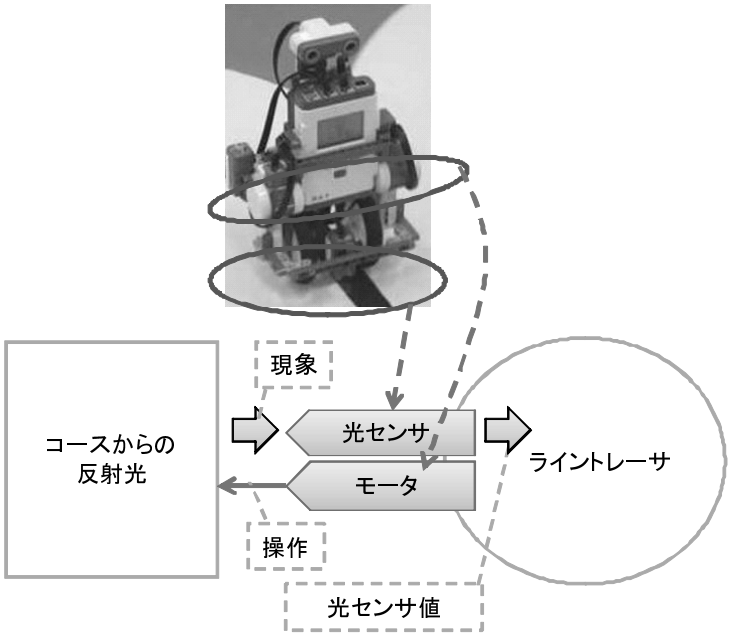

図 7.14: ライントレーサ

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 制御を用いる．この要求仕様は次の通りである．また，その設計のシステム構成図を図 7.15 に示す．このシステム構成図においてシステムには，フィードバック制御を行うコントローラ部があり，センサの光センサを制御するモジュール，そしてアクチュエータのモータを制御するモジュールが存在する．外部環境には，光センサ値の変化に直接影響を及ぼすラインの色をモデル化している．

- 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

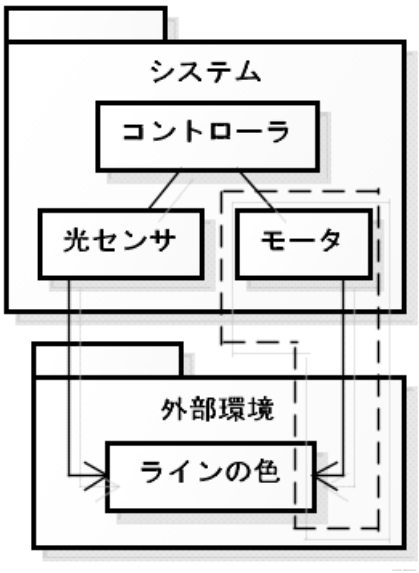

図 7.15: ライントレーサの設計のシステム構成図

よる旋回を行う.

- 光センサから値を取得する周期は 1ms 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－ 1T 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－ 1T の光センサ値）－（1T の光センサ値－ 2T 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．そのため検査する性質は,「直線をトレースする時，ライントレーサが旋回を行う力はいつか無くなる」や「デッドロックは常に発生しない」などが考えられるが，本事例では「今回のセンサ値と閾値の差が 1T のセンサ値と閾値の差より小さければ，モータへのパワー値は 1T より必ず小さい.」との検査する性質について検証する.

7.2.2　ライントレーサの検査モデル

この検査対象モデルは，次のような振る舞いを行う．図 7.16 に示すように，まず外部環境モデルから光センサ値を同期受信し，受信した値を今回の取得値として変数 *color*0 に格納する．次に，PID それぞれの成分の計算を行う．P 成分は今回の値と 1T の値との差分に係数 36 をかけたものである．I 成分は今回の値と閾値との差分に係数 1 をかけたものである．D 成分は今回・1T の差分と 1T・2T の差分との，差分値に係数 4 をかけたものである．この P 成分と I 成分と D 成分の総和が，モータへのパワーとなる．

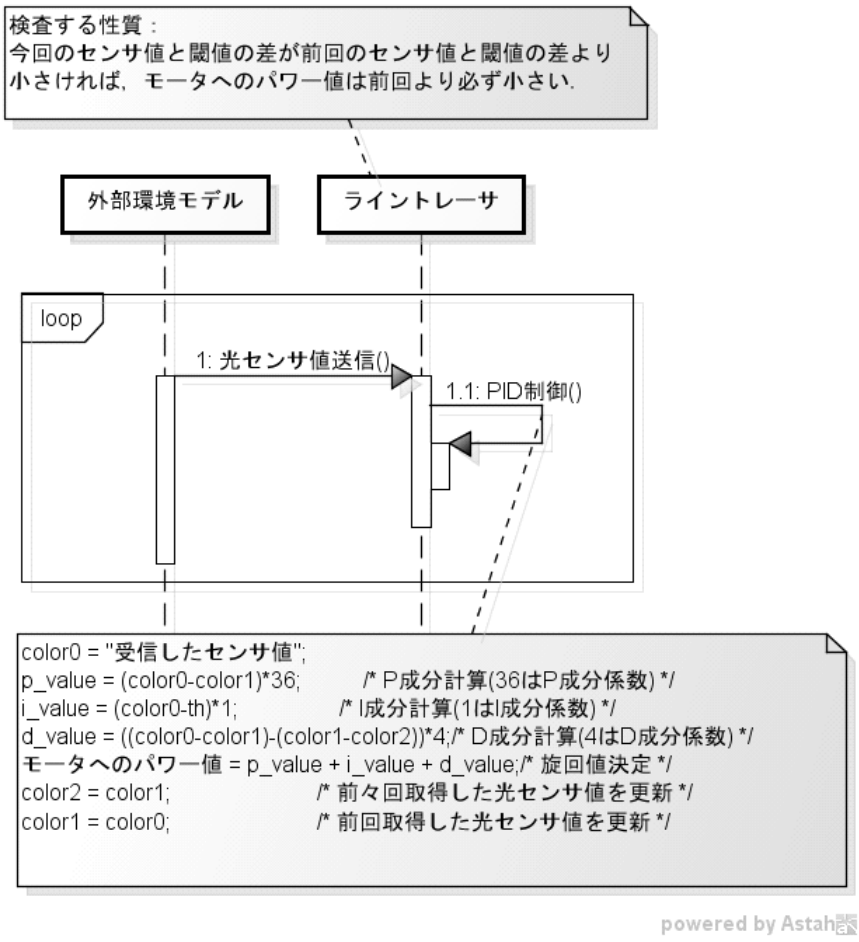

図 7.16: 対象の検査モデルの概略図

また，次にセンサ値取得の際の制御を行うため，変数 *color*1 に格納されている

1T の値を 2T の値を格納する変数 $color2$ へ，変数 $color0$ に格納されている今回の値を 1T の値を格納する変数 $color1$ へ格納する．

このライントレーサの検査モデル，付録の検査モデル 3.1 である．このシステムの検査モデルは，ライントレーサの振る舞いをシステム記述として記載し，ライントレーサがセンシングする光センサ値を生成する外部環境モデルの振る舞いを外部環境記述として記載する．この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，付録の検査モデル 3.1 のように，図 7.15 の点線部分の記述を行っていない．

7.2.3　ライントレーサへの提案手法の適用

提案手法の効果確認を行うため，図 6.1 の流れに沿って，まず EIVP の適用を行い，次に GWEU の適用を行う．着目すべき入力変数を抽出する EIVP では，まず検査する性質「今回のセンサ値と閾値の差が 1T のセンサ値と閾値の差より小さければ，モータへのパワー値は 1T より必ず小さい」を Top goal とし，単一の入力もしくは出力変数を含んだ Sub goal へとゴール分解を行う．その結果は，6.2.2 節で説明した通り，着目すべき入力変数である，現在から 1T,2T,3T までの履歴のセンサ値を格納する変数が必要であることを抽出できている．GWEU の適用結果についても，6.3 節で述べた通り,「フルバッテリ残量」の考慮不要な外部環境の要因の抽出が行え，その要因を排除した「坂道を下る」「少し消耗したバッテリ残量（95%）」「PID 制御（P パラメータ=36,I パラメータ=1,D パラメータ=4)」「急カーブを曲がる」の要因による「フルスピードで急カーブを曲がる」のワーストケースシナリオの分析が行えた．

次に，その状況について 4.2 節の図 4.2 にあるプロトタイププログラミングを用いた最大時系列変化幅の実測と時系列変化の制約の抽出を行った結果，5.2 節で述べた通り，時系列変化の制約は 3 であったが，排除しない（「フルバッテリ残量」を用いた）場合の時系列変化の制約は 5 であった．

それらの時系列変化の制約を基に付録の検査モデル 3.1 に対して，GWEU を用いて考慮不要な外部環境の要因を排除した場合（「少し消耗したバッテリ残量」

を用いた時系列変化の制約が 3 の場合）の付録の検査モデル A.2 と，GWEU を用いなかった場合（「フルバッテリ残量」を用いた時系列変化の制約が 5 の場合）の付録の検査モデル A.1 とでモデル検査結果を比較した．その結果，5.2 節で述べた通り時系列変化の制約が 5 の場合は状態爆発が発生したが，本手法を適用した時系列変化 3 では状態爆発が発生しなかった．

7.2.4 異なる性質でのライントレーサへの提案手法の適用

一方，検査する性質を仮に「1T と 5T のセンサ値が異なれば，1T から 3T までのモータへのパワー値はそれぞれ異なる.」とした場合，つまり，検査する性質に含まれる変数を 7.3 式から 7.4 式に変化させた場合の性質について EIVP を適用する．

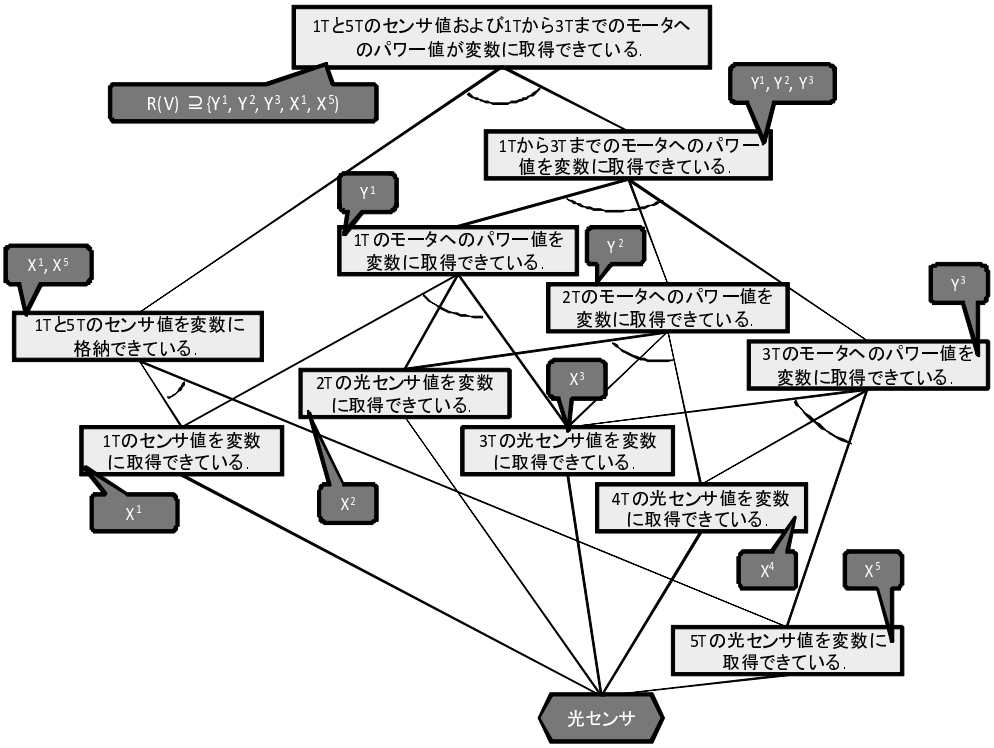

図 7.17: ライントレーサの事例の EIVP（異なる性質）

$$R(V) \supseteq \{Y^0, Y^1, X^0, X^1\} \tag{7.3}$$

$$R(V) \supseteq \{Y^1, Y^2, Y^3, X^1, X^5\} \tag{7.4}$$

ここで，V を検査する性質，$R(V)$ を V に関連する入出力変数の集合，出力変数を Y，入力変数 X とする．その結果，図 7.17 であり，今回から 5T までの履歴のモータへのパワー値を格納する入力変数であることを抽出した．また，GWEU の適用において，着目すべき入力変数は異なるがワーストケースシナリオの分析結果は同じであったため，7.2.4 節と同じ結果が得られた．

7.3 他研究との削減効果の比較

外部環境を限定してモデル検査を行う手法について，同様の研究である [21] との比較を行う．本手法は，システムが動作する環境上で外部入力の時系列変化が最大となる値を分析し，外部環境モデルへ反映させることで，考慮しなくてもよい外部環境の状態遷移を排除したモデル検査を行う手法である．一方，[21] は外部環境モデルに含まれる状態遷移に対して，シナリオ毎に分離させ個々の状態遷移についてモデル検査を行うことで，全体の状態遷移のモデル検査を行う手法である．

これらの手法の比較を行うために，まず 7.3.1 節と 7.3.2 節にて話題沸騰ポットとライントレーサの事例を用いた，適用前の検査モデル，本手法を適用した場合の検査モデル，そして [21] を適用した場合のモデルについて述べる．次に 7.3.3 節にてそれぞれの検査モデルを用いたモデル検査の結果から状態遷移の削減数の比較を行う．

7.3.1 話題沸騰ポットの検査モデル

まず，話題沸騰ポットでの検査モデルについて述べる．適用対象の検査モデルを検査モデル A.9 に示す．この検査モデルと検査モデル A.6 との相違点は次のとおりである．外部環境の情報の履歴を 2T まで（X^0, X^1, X^2 まで）扱っているため，外部環境から取得する回数は 3 回で十分であり，29 行目と 46 行目にてその取得回数を 3 に制限している点が異なる．検査モデル A.9 は検査モデル A.6 を抽

象化したものであるため，また検査する性質も 7.1 節と同様に「今回の水温と 2T の水温が異なれば，ヒータの熱量は 2T より必ず大きい」としたため，本提案手法の適用結果による変化幅の制限値は検査モデル A.8 と同じものとなった．そのため，本手法を適用した検査モデルは，検査モデル A.9 の外部環境記述が検査モデル A.8 の外部環境記述と置き換わったモデルとなる．次に [21] の適用において，外部環境モデルの取得は 3 回であり約 160（取り得る値の範囲-10〜150）の 3 乗の組合せとなるが，シーケンスを分離させる考えから初回は固定値とし残り 2 回を取得する際のモデル検査を実施した．つまり初回に取り得る個数分の約 160 通りのシナリオに分割し，モデル検査を行うと想定した場合のモデル化を行った．その結果が検査モデル A.10 である．30〜34 行目と 50 行目において，初回を固定値 0 とした場合の検査モデルとしている．もし，このモデル検査の結果において状態爆発が発生しなければ，この検査の初期値を変えた約 160 回通り実施することで，全体の状態遷移に対して状態爆発なしでモデル検査の実施が行える．

7.3.2 ライントレーサの検査モデル

次に，ライントレーサでの検査モデルについて述べる．適用対象の検査モデルを検査モデル A.3 に示す．この検査モデルと検査モデル 3.1 との相違点は，前述の話題沸騰ポットと同様に，外部入力の履歴値を格納する変数は X^0, X^1, X^2 を扱っているため，13 行目と 22 行目にて外部環境から取得する回数を 3 回としている点である．検査モデル A.3 は検査モデル 3.1 を抽象化したものであるため，また検査する性質も 7.1 節と同様に「今回のセンサ値と閾値の差が 1T のセンサ値と閾値の差より小さければ，モータへのパワー値は 1T より必ず小さい」としたため，本提案手法の適用結果による変化幅の制限値は検査モデル A.2 と同じであった．そのため，本手法を適用した検査モデルは，検査モデル A.4 の外部環境記述が検査モデル A.2 の外部環境記述と置き換わったモデルとなる．次に [21] の適用において，外部環境モデルの取得は 3 回であり約 200（取り得る値の範囲 500〜700）の 3 乗の組合せとなるが，シーケンスを分離させる考えから初回は固定値とし残り 2 回を取得する際のモデル検査を実施した．つまり初回に取り得る個数分の約 200 通りのシナリオに分割し，モデル検査を行うと想定した場合のモデ

ル化を行った．その結果が他研究での検査モデルは検査モデル A.4 である．10 行目と 15〜17 行目と 25 行目において，初回を固定値 700 とした場合の検査モデルとしている．もし，このモデル検査の結果において状態爆発が発生しなければ，この検査の初期値を変えた約 200 回通り実施することで，全体の状態遷移に対して状態爆発なしでモデル検査の実施が行える．

7.3.3 他研究との状態遷移数の削減比較

7.3.1 節と 7.3.2 節で述べた，適用前と本手法そして他研究の検査モデルでの，話題沸騰ポットとライントレーサの事例における状態遷移数を表 7.1 に示す．この結果より，本手法と他研究を比べた場合，両事例とも 90% 以上の状態遷移数の削減が行えている．また，両手法とも 2.1.2 節で述べた過小近似と捉えることもできる．

表 7.1: 他研究との比較表

事例	検査モデル	状態遷移数	備考
話題沸騰ポット	適用前	状態爆発	-
	本手法	367	-
	他研究	267466	160 回実施要
ライントレーサ	適用前	状態爆発	-
	本手法	1763	-
	他研究	385521	200 回実施要

7.4 事例に基づく評価

本節では，まず，5 章で述べた 2 つの問題に対する 6 章で述べた提案手法，EIVP および GWEU の評価を 7.4.1 節と 7.4.2 節で述べる．そして本手法を用いた “入力値の時系列の変化を限定する手法” に対する評価を 7.4.3 節にて述べる．

7.4.1 「着目すべき入力変数の抽出が困難」に対する EIVP の評価

ワーストケースシナリオを分析するためには，まず着目すべき入力変数に着目する必要がある．しかしながら，着目すべき入力変数は，検査する性質や設計モデルにすべては表現されない．

これに対して EIVP は，まず 7.1.3 節の話題沸騰ポットの事例において，

- Top goal を「今回の水温と 2T の水温が異なれば，ヒータの熱量は 2T より必ず大きい」とし，サーミスタ値およびヒータへのパワー値に分解し，さらにそのパワー値を導出するためのサーミスタ値を抽出することで，検査する性質や設計モデルに表現されない，現在から 1T,2T,3T,4T までの履歴のサーミスタ値を格納する入力変数（着目すべき入力変数）が必要であることを抽出できている．

- 検査する性質を変えた場合の適用の 7.1.4 節においても，Top goal を「今回と 1T と 2T の水温がそれぞれ異なれば，ヒータの熱量は 4T のものより必ず大きい」として，ゴール分解を行った結果，今回から 2T と 4T から 6T までの履歴のサーミスタ値を格納する入力変数が必要であることを抽出できている．

次に 7.2 節のライントレーサの事例において，

- Top goal を「今回のセンサ値と閾値の差が 1T のセンサ値と閾値の差より小さければ，モータへのパワー値は 1T より必ず小さい」とし，センサ値およびモータへのパワー値に分解し，さらにそのパワー値を導出するためのセンサ値を抽出することで，検査する性質や設計モデルに表現されない，現在から 1T,2T,3T までの履歴のセンサ値を格納する入力変数（着目すべき入力変数）が必要であることを抽出できている．

- 検査する性質を変えた場合の適用の 7.1.4 節においても，Top goal を「1T と 5T のセンサ値が異なれば，1T から 3T までのモータへのパワー値はぞれぞれ異なる.」として，ゴール分解を行った結果，今回から 5T までの履

歴のモータへのパワー値を格納する入力変数が必要であることを抽出できている．

これらの結果から EIVP は，3.1 節で述べた対象のセンサ・アクチュエータシステムにおいて，ドメイン（外部環境）が異なる設計モデルであっても，異なった検査する性質であっても，着目すべき入力変数の抽出を可能とする有効な手法である．

なお，EIVP のゴール木が大きくなる要因は，対象の検査モデルが 7.5 式のように右辺に出力変数が含まれている場合，$R(V)$ が 7.6 式であれば，ゴール木は 6.1 式の $t+a$ もしくは $t+b$ 個分の X および Y が出現する．

$$Y^0 = f(Y^1, X^0, X^1) \tag{7.5}$$

$$R(V) \supseteq \{Y^0, Y^1, X^0, X^1\} \tag{7.6}$$

つまり，図 7.18 のようにシステムの初期状態の履歴値まで遡るため，ゴール木

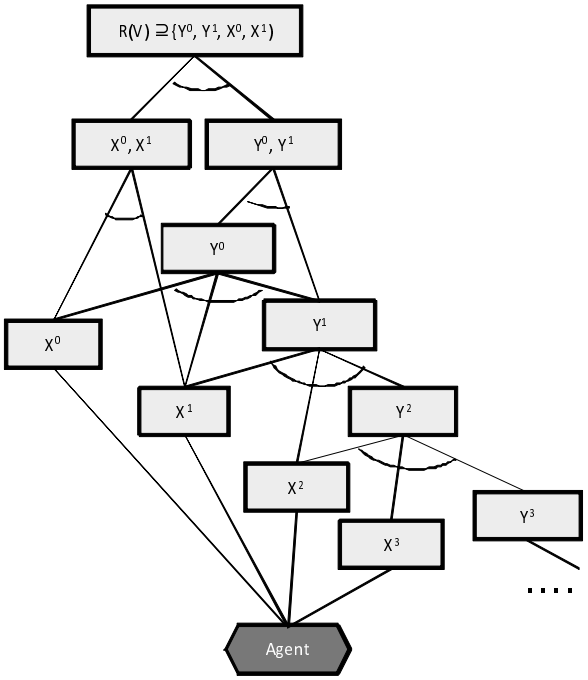

図 7.18: EIVP に不向きなモデル

が大きくなりEIVPとしては適用し難い．この事から，数式の右辺に出力変数が含まれる対象は，EIVPのゴール木の作成が困難であるが，初期状態の履歴値まで全て影響する事がわかるためゴール木の作成自体が不要となる．

7.4.2 「考慮不要な外部環境要因の排除が困難」に対するGWEUの評価

従来の手法では，着目すべき入力変数を抽出した後，ワーストケースシナリオを分析する必要があるが，不要な外部環境の要因も含まれる場合はそれらを排除することが困難である．これに対してGWEUは，まず7.1.3節の話題沸騰ポットの事例において，

- 手順１：８つの分析観点を検討しており，その結果である図7.6の水位と水温の関係から，センサ値の時系列変化が最大となる状況「少量の水での最大加熱を行う」を想定できている．
- 手順２：「少量の水での最大加熱を行う」をTop goalとし，サーミスタ値が変化する要因をシステムの振舞いと外部環境の要因の２つの観点でゴール分解し，本手法のオリジナリティである点線でグルーピングすることで，センサ値の時系列変化が最大となる状況の表現から，500Wの加熱やその時の水温など，より具体的な条件を，２つの観点でグループ化した抽出が行えている．
- 手順３：その点線で外部環境の要因がグループ化されていることから，そのグループ内での考慮不要な外部環境の要因の分析をより容易化し，障害が現れる条件の「実測時の状況」の考慮において，考慮不要な外部環境の要因「50cc残っている」の排除が行えている．また，その要因を排除した「100cc残っている」「水位センサがON状態」「ターゲットの温度に満たない（97°C）」「500Wで加熱する」との要因による「少量の水で最大加熱を行う」の状況に対応したワーストケースシナリオの分析が行えている．

と手順１～３が行えている．次に7.2節のライントレーサの事例において，

- 手順１：８つの分析観点を検討しており，その結果である図 7.6 のコースと色の関係から，センサ値の時系列変化が最大となる状況「フルスピードで急カーブを曲がる」を想定できている．

- 手順２：「フルスピードで急カーブを曲がる」を Top goal とし，センサ値が変化する要因をシステムの振舞いと外部環境の要因の２つの観点でゴール分解し，本手法のオリジナリティである点線でグルーピングすることで，センサ値の時系列変化が最大となる状況の表現から，バッテリ残量や PID 制御のパラメータ値など，より具体的な条件を２つの観点でグループ化した抽出が行えている．

- 手順３：その点線で外部環境の要因がグループ化されていることから，そのグループ内での考慮不要な外部環境の要因の分析をより容易化し，障害が現れる条件の「初期状態から実測までの状況」を考慮する事で，考慮不要な外部環境の要因「フルバッテリ残量」の排除が行えている．また，その要因を排除した「PID 制御で走行する」「P パラメータ値を 36 として走行する」「I パラメータ値を 1 として走行する」「D パラメータ値を 4 として走行する」「坂道を下る」「坂道までバッテリを消費する」「消耗したバッテリ残量（95%）で走行する」「急カーブを曲がる」の要因による「フルスピードで急カーブを曲がる」の状況に対応したワーストケースシナリオの分析が行えている．

と手順１～３が行えている．これらの結果から GWEU は，3.1 節で述べた対象のセンサ・アクチュエータシステムにおいて，ドメイン（外部環境）が異なる設計モデルであっても，異なった検査する性質であっても，オリジナリティである点線とそのグルーピングによるゴール分解を用いることで，その他同様なセンサ値の時系列変化が最大となる状況から，具体的な条件の抽出および考慮不要な外部環境の要因の排除したワーストケースシナリオの分析を可能とする有効な手法である．

なお,「障害」が現れる条件は次の通りである．システムの動作条件が複雑で，実運用での，システムの振舞いと外部環境の状況の整理を行わなければ，考慮が不要な要因の洗い出しが困難なシステムである場合である．実運用上で考慮が不

要な要因を「障害」としているため，システムの動作条件が多いほど，実運用上において考慮不要な要因が多くなり「障害」の発生確率が上がる．そのため，システムの動作条件が複雑なシステムであればあるほど「障害」が発生し，考慮不要な外部環境の要因の排除が可能となる．結果として，動作条件が複雑なシステムは本提案手法の効果が大きくなる．また，GWEU のゴール木が大きくなる要因は，機能と外部環境が含まれているシステムの構成要素が大きれば，GWEU のゴール木が大きくなる．ゴール木が大きくなった場合，提案手法でのグルーピングによる外部環境要因の散見を防止したとしても，分析を行う外部環境の要因が多いため考慮不要な要因の分析が困難になる．そのため，構成要素が多いシステムは分析が困難となり適用しづらい．

7.4.3 事例研究の全体評価

本研究の対象である入力値の時系列変化を限定する手法の 4.2 節の図 4.2 を基に，7.1 節の話題沸騰ポットの事例と 7.2 節のライントレーサの事例において，本提案手法 EIVP と GWEU 用いた，状態削減の効果の比較を行う．また，本手法のでワーストケースシナリオの分析の妥当性について述べる．

話題沸騰ポットの事例では，EIVP で抽出した着目すべき入力の「現在から 1T,2T,3T,4T までの履歴のサーミスタ値を格納する入力変数」を基に，ワーストケースシナリオを分析し，GWEU で考慮不要な外部環境の要因を排除することで，モデル検査の比較を行った結果 85% の状態数削減が確認された．また，ライントレーサの事例では，同様に EIVP で抽出した着目すべき入力の「現在から 1T,2T,3T までの履歴のセンサ値を格納する入力変数」を基に，ワーストケースシナリオを分析し，GWEU で考慮不要な外部環境の要因を排除することで，状態爆発を起こしモデル検査が行えなかったことに対し，モデル検査の網羅探索の実施が可能となった．また，7.3 節で示した通り，他の研究 [21] と比べても本手法は約 90% 以上の状態遷移数の削減効果があった．よって，事例や検査する性質に依存するが，同様のセンサ・アクチュエータシステムにおいて，ワーストケースシナリオの分析によって一定の状態の削減効果がある．

そのワーストケースシナリオの分析の妥当性について評価する．モデル検査を

行う上では，検査する性質に依存した情報に着目する必要がある．本手法でも，EIVP で検査する性質に依存する入力変数（着目すべき入力変数）の抽出を行い，GWEU にてその変数を基にワーストケースシナリオの分析がなされている．ここから，着目すべき入力変数を抽出せずワーストケースシナリオを分析した場合と比べ，本手法の EIVP で抽出した結果を基に分析した方が，より妥当な分析が行えていると言える．

7.4.4 本手法が有効でない問題領域とその特性

まず，出力値（アクチュエータを制御するための値）が入力値の履歴（周期的に取得したセンサ値の履歴）のみを用いて算出されるモデルの場合は有効であるが，出力値の履歴を用いて算出される場合，EIVP のゴール分解において再帰的にゴール分解を行うことになるため提案手法として有効でない．その場合，モデルにも依存するが履歴値を格納する全入力変数が着目すべき入力変数となり，本提案手法の改良よる適用においても効果は望めない．次に，入力値に関してデータマッピングによる抽象化を行った場合の検査モデルは，センサ値の履歴を扱わない（離散化された値となりセンサ値が取得する履歴値ではなくなる）ため，センサ値の時系列変化に着目できず提案手法を適用できない．入力値が離散化されることで，本提案手法の特徴である時系列変化への着目ができないため，本手法を改良して適用を行う対象とならない．最後に，ワーストケースシナリオにおいて，センサ値の時系列変化が極端に大きい場合（例えば，n 時から n+1 時のセンサ値の時系列変化が，センサ値が取得できる最大値から最小値に変化する場合），時系列変化の幅を限定できないため提案手法は有効でない．しかしながら，時系列変化の幅が変化する要因について何らかの特徴を見つけることが出きれば，本手法を改良することで，この時系列変化が極端に大きい対象についても，制約による探査空間の削減は望める可能性はある．

7.5 複数のセンサ・アクチュエータを持つ対象の検査モデル

対象の検査モデルが複数のセンサ・アクチュエータである場合，各アクチュエータの出力変数ごとに検討することになる．例えば，3.4 式（$Y_1^0 = f(X_1^0, X_1^1, X_1^2, X_2^2, X_2^3)$）と 3.5 式（$Y_2^0 = f(X_1^2, X_1^3, X_2^0, X_2^1)$）が対象の検査モデルであり，検査する性質が 7.7 式のように，それぞれの値を格納した変数の集合 $R(V)$ である場合の EIVP と GWEU の適用について次に述べる．

$$R(V) \supseteq \{Y_1^0, Y_2^1, X_1^1, X_2^0\} \tag{7.7}$$

3.4 式と 3.5 式のように，対象の検査モデルに X_1 と X_2 の入力変数が含まれていても，まず X_1 に対する検査モデルについて本手法を適用し，続いての X_2 について適用が可能である．複数のセンサを扱う 3.4 式と 3.5 式に対する EIVP と GWEU の適用を次に述べる．

まず EIVP において，7.7 式に含まれる Y_2^1 に対する入出力変数の関係式は，対象の検査モデル 3.5 式の現在 Y_2^0 から，6.1 式（$Y^t = f(Y^{t+1}, Y^{t+2}, \cdots, Y^{t+a}, X^t, X^{t+1}, \cdots, X^{t+b})(1 \leq a, 0 \leq b)$）で述べた関係を用いて 7.8 式となる．

$$Y_2^1 = f(X_1^3, X_1^4, X_2^1, X_2^2) \tag{7.8}$$

そのため，EIVP のゴール分解の意味は，下位ノードの論理積が上位ノードである KAOS の意味に加えて，下位ノード（3.4 式の右辺の引数（変数））の論理積が上位ノード（3.4 式の左辺の変数）と，下位ノード（7.8 式の右辺の引数（変数））の論理積が上位ノード（7.8 式の左辺の変数）となる．これらの意味を用いて，Top goal を 7.7 式として，EIVP のゴール分解を適用した結果は図 7.19 となり，着目すべき入力変数は Sensor1 の入力変数 $X_1^0, \cdots, X_1^4$ と Sensor2 の入力変数 $X_2^0, \cdots, X_2^3$ であると抽出が行えた．このように EIVP に関しては，複数のセンサ・アクチュエータであっても適用結果の入力変数の種類が増えるだけで手順は変わらないため，センサ数が 2 つ以上であっても適用が可能である．

次に GWEU において，本提案手法はシステムの動作環境上での対象のセンサ値の最大変化を分析することで，対象のセンサ値の組合せを制限するものであ

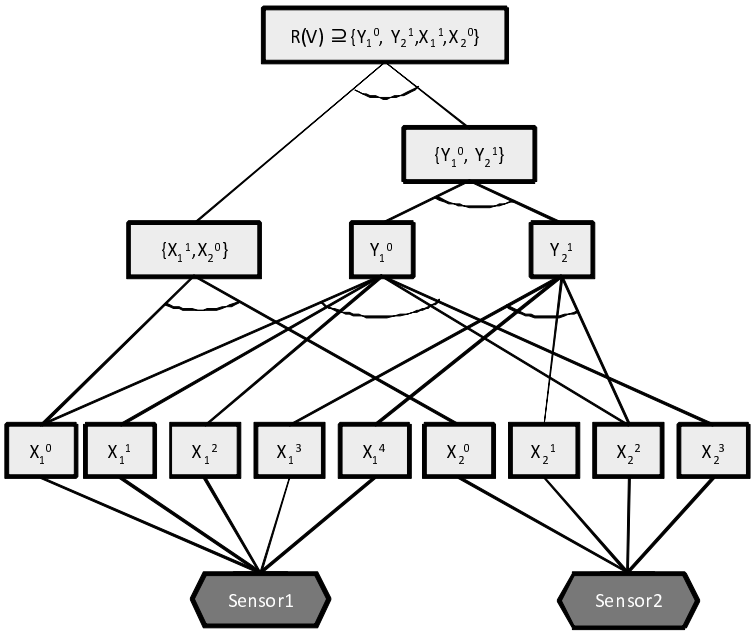

図 7.19: 複数のセンサ・アクチュエータのモデルに対する EIVP

る．そのため，複数のセンサを持つ場合は個々の最大変化について分析すればよい．よって，GWEU は単一のセンサ値に対したワーストケースシナリオ（センサ値の時系列変化の最大値を測定ことを目的としたシステムの動作順序）を抽出する手法であるため，対象のセンサが複数になれば個々のセンサ値に対して GWEU を適用すればよい．例えば，3.4 式と 3.5 式と 7.7 式の例では，着目すべき入力変数は Sensor1 の入力変数 $X_1^0, \cdots, X_1^4$ と Sensor2 の入力変数 $X_2^0, \cdots, X_2^3$ であるため，まず Sensor1 のセンサ値に対するワーストケースシナリオの分析を行い，その後 Sensor2 のセンサ値に対するワーストケースシナリオの分析，個別で GWEU の適用が可能である．

これらより，EIVP および GWEU の適用が可能であるため，複数のセンサ・アクチュエータを持つ検査モデルについても本提案手法を適用することが可能であると言える．

第 8章

関連研究

この章では関連研究として，まず抽象化手法による状態削減についての研究を述べ，外部環境の入力を限定する手法について述べる．次に，テストースの自動生成，外部環境のモデル化，そしてゴール指向要求分析について述べる．

8.1 抽象化手法による状態削減

8.1.1 一般的な抽象化手法

本節では，一般的な状態削減のための抽象化法である，データマッピング法による抽象化 [13][19] とプログラムスライシング [57] の技術に類似した抽象化 [13]，そして述語抽象化 [33] の説明を行う．

まずデータマッピング法は，データ抽象とも呼ばれ，取り扱う変数の値をデータとそれを操作する手続きの一体化によって，扱う個数を削減させる抽象化法である．たとえば，整数を扱う場合，抽象化前では整数の個数分，つまり無限個扱わなければならない．しかし，必要な情報が正・負・ゼロが分かれば良い場合，データマッピング法によって，抽象化後は３つの個数を扱うだけでよくなる．

次に述語抽象化は，システムの状態空間を述語で表現することで，状態数を削減する抽象化である．述語とは，変数を持つ等式もしくは不等式であり，その変数に実数値を与えて真偽値を返す条件文である．故に，真偽値で抽象化された状態の表現が可能である．たとえば，述語の数を κ とすると，連続空間 $\chi \subset \mathrm{R}^n$ は

κ 個の述語を用いて分割される．連続空間 χ は述語の条件文の真偽の 2 通りでによって分割されるため，抽象状態空間は述語の数が κ 個であれば 2^{κ} 個の抽象状態で表現できる．

そしてプログラムスライシング法の技術に類似した抽象化は，プログラムから目的とするソースコードを抽出する抽象化である．複数の機能を持つあるプログラムから，検証したい機能だけが動作するプログラムを抽出する方法である．この時，注目した部分の動作に関係する部分を抽出できるだけでなく，その動作も完全に保存するという特徴がある．

一般的な抽象化手法は，検査するモデルの振舞いに依存しないシステムの状態を削除するが，本手法はシステムが動作する環境から関係しない外部入力値を削除する手法である．特に本手法では，外部環境の入力値の時系列変化を制限する視点に着目してるため，適用するシステム（3.1 節を参照）と検査する性質（3.2 節を参照）に依存して，状態削減の効果が大きくなる．

8.1.2 CEGAR(Counter-example Guided Abstraction Refinement)

モデル検査の検査結果の偽反例に対して仮の条件を与えて検査を行う研究 [4][15][2] がある．これらの内 [4][15] 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，[2] 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にて検査を行う．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．

条件を付与してモデル検査を実施することは本手法と同じであるが，前術の研究はモデル検査器に入力された情報から機械的に導出された条件を扱う．これに対して，本手法の分析を自動的に行うためには，判断するために必要なシステムの振舞いや外部の環境の情報を全て入力して計算する必要があり膨大な計算量となるため現実でない．しかし本手法は，設計モデルや検査する性質にはないシステムの動作環境から，入力値の変化が最大となるシナリオを分析し，そのシナ

リオでの実測値を基に入力値の変化幅を限定してモデル検査を行う手法である．よって，CEGAR は入力値の時系列変化が最大となる条件に着目して抽出できないが，本手法は人為的判断にて（ワーストケースシナリオ分析の上），その条件の抽出が可能となる．

8.2 Assume-Guarantee

Assume-Guarantee 検証 [9][14][17][37] は，環境の動作を仮定して構成的に検証するものであり状態爆発を抑制できる．例えば，検査する性質 f と g が両方成り立っていることをモジュール M に対して検証する際，一般的に f と g を別々に証明するが，Assume-Guarantee 検証では，動作時の仮定となる論理条件 f が与えられることが前提として．各モジュール M が満たすべき性質 g を検証する方法のことである．また，性質 g を満たすべき最小の M モジュールの環境を制限する weakest assumption [23] などがある．

本研究はこの weakest assumption に類似しており，従来のものとの差分は，時系列変化の幅に着目した点と，時系列変化の制約値を外部環境のモデル化から分析し，そのある一定の根拠による制約値を用いたモデル検査の実行が行える点である．

8.3 有界モデル検査（Bounded Model Checking）

有界モデル検査は，モデル検査時の網羅探索を一定の深さまで行うことで状態爆発を緩和し，効率的に不具合を発見するモデル検査の手法である [1]．また，状態遷移グラフや二分決定グラフ (BDD) の全状態を記憶せず探索中の経路情報のみを記憶して探索を行い，充足可能性判定器（SAT solver）を用いて自動検証が可能で，網羅的に探索が可能であることから，産業界への導入にも着目が置かれている．しかしながら，探索空間外に不具合が存在するかどうかは判らない．一方，実際の多くの不具合は，仕様を逸脱する小さな実行シーケンスを持つ経験的事実（Small Scope Hypothesis）[35] との考えがある．この考えを踏まえた有界モデル検査の手法として Alloy[20][24] などもある．これらの有界モデル検査の考

え方を用いれば，入力値の時系列変化の幅を反例が発見されるまで小さい値から順に大きくする方法が考えられるが，どれくらいまで値を大きくすれば十分に検証できているかが不明である．また，入力変数が 2 つ以上になると，入力値の時系列変化の幅の組み合わせ爆発が起こる可能性がある．

本研究では，入力値の時系列変化の幅をアドホックに組合せるのではなく，EIVP および GWEU を用いることで，検査する性質に関係した外部環境を考慮した適切な組合わせを求めることができる．そのため，システムの実運用上で十分と判断できる入力値の時系列変化の幅を抽出することができ，また変化幅の組み合わせ爆発を回避できるため優位である．

8.4 外部環境の入力を限定する手法

外部環境を分析して条件を付与する研究の中で，システムが使用されるユースケースの前提による条件を付与し，モデル検査を行う研究 [21][22][25] がある．この研究は，システムのユースケース図やそれに伴うシーケンス図を，CDL(Context Description Language) という外部環境の記述言語に出力し，モデル検査器の記述言語に変換して外部環境記述の振舞いの範囲を限定し，モデル検査を行う研究である．

前述の研究に対して本研究は，ユースケースにより条件を付与するところは同じであるが，特に検査する性質に着目した入力値の時系列の変化が最大となるユースケースに焦点を当てているところが異なる．その結果，本手法では入力値の時系列変化の制約を抽出し，その制約を付与したモデル検査が行える点が異なる．

8.5 テストケースの自動生成

さらに，外部環境を分析して入力値を限定する視点は，テストケースの自動生成に関する研究も関連する．モデル検査を用いた自動テスト生成の研究 [5][48] は，要求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ある．そのテストシーケンスは，範囲が限定された整数型，論理型，そして列挙型 など，さま

ざまなデータタイプを基に，試験における開発費の削減を目的とした，効率的な試験ケースを抽出する手法である．

本提案手法は，要求仕様書から検査に不要な外部環境からの入力値を排除する，つまり検査に必要な外部入力を抽出する研究である．しかし，テストケースの自動生成のように複数の条件（前述のデータタイプ）について外部環境を分析して抽出する必要がある．本手法は，時系列変化の制限を行う一つの条件を抽出する手法であるため，前述の外部環境の分析を行う一つの手法として扱うことが可能である．

8.6 外部環境のモデル化

次に，外部環境のモデル化法の関連研究について述べる．まず，外部環境を分析しモデル化を行っている研究として [18][51][52][53][54][56] 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，要求仕様の観点からシステム全体が関わる外部環境の分析とモデル化を行っている．次に，ドメイン領域，その接続関係，システム，そしてそれらと問題領域との接続を表す，問題領域の外部環境のダイアグラム（”a problem diagram”）に関する研究 [40][16][34] がなされている．これらの研究は，対象のソフトウェア制御に関係する外部環境を分析する研究である．また，システムの外部環境に着目している視点は，[45][46][43] で提唱されている．しかし，仕様を信頼または保障するその独自なアプローチは，温度など連続的に変化する物理現象の対象へは十分に対応できていないため，様々なパラメータを用いることで，それらに対応した手法が [10][11] にて提案されている．そして，[12] を含めて，それらのアイデアを用いた様々な研究が行われている．

本論文では [18] を基に，モデル検査を行うため検査する性質から扱うセンサ値に着目することで，GWEU の手順 1 においてセンサ値の時系列変化が最大となる状況の分析を可能としている．

8.7 ゴール指向要求分析

8.7.1 KAOS

KAOS[8][27] と本提案手法で用いたゴール指向分析との違いについて述べる．KAOS の要求分析は，ゴールモデル，責任モデル，操作モデル，オブジェクトモデルの４つのモデルから構成され，サブシステムの役割までゴール分解を行い，Sub goal 間の要求に対する障害関係を分析する．しかし，ワーストケースシナリオから考慮不要な外部環境の要因を排除するゴール指向分析手法（ GWEU - Goal-oriented analysis of Worst case scenario with Eliminating Unnecessary contexts ）は，KAOS のゴールモデルのみ使用しており，システムの振舞いと外部環境の要因の観点にてグルーピングを行う点線がオリジナリティとなる．この点線により，ゴール分解時に外部環境の要因が散見されるのを防ぐことを可能としている．

8.7.2 i*

KAOS と同様にゴール指向要求分析の手法である i*について述べる．i*[58][59][60] は，As-Is 分析（early requirements hase）をサポートしている．i*は，システムへの要件定義（To-Be システム）を行う前に，現状を詳細に分析することが特徴である．また，i*は As-Is，To-Be のそれぞれに対し，さらにモデルを２種類に分類している．一つは，戦略的依存関係モデル（Strategic Dependency (SD) model）は，アクタ間の関係を表すことにより，組織的側面に注目するモデルである．もう一つは，戦略的根拠モデル（Strategic Rationale (SR) model）は，各アクタの関心事を詳細に記述するモデルである．特に SR model の関心事を種別化してゴール分解を行う視点は，GWEU の２つ観点で分解する内容と同じである．しかしながら，GWEU は外部環境からの入力値が変化する要因を分類する明確な目的があり，それに伴った記法およびより具体的なゴール分解を可能としている．

第 9章

おわりに

9.1 まとめ

本論文では，実行環境情報を考慮して外部環境モデルに時系列変化の制約を設けることで，状態数もしくは状態遷移数を削減させる手法，いわゆる入力値の時系列の変化幅を限定させる方法についてまとめた．また，その時系列変化の制約を抽出する上で 2 つの問題が存在したが，その対策として，検査する性質に依存する入力変数を既存のゴール指向分析手法を用いて抽出する手法（ EIVP - Extraction method of Input Variable related to a verification Property ）と，ワーストケースシナリオから考慮不要な外部環境の要因を排除するゴール指向分析手法（ GWEU - Goal-oriented analysis of Worst case scenario with Eliminating Unnecessary contexts ）を用いたワーストケースシナリオ分析の手順を提案した．さらに，話題沸騰ポットとライントレーサの事例を用いて，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．更に，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．

なお本提案手法は，6.1 節の図 6.1 のワーストケースシナリオを分析した範囲で，プロトタイププログラミングを用いて，入力値が最大と時系列の変化幅の実測を行う．しかし，その値の導出方法については言及していない．例えば，その値が要求仕様書に記載されていれば，その記載されている値を採用すればよい．

9.2 今後の展望

本論文では複数のセンサおよびアクチュエータを持つシステムを対象とし，センサ毎にワーストケースシナリオを分析する手法を提案しているが，センサ間の依存関係によって考慮不要な時系列変化の制約の抽出では困難である．例えば，２つのセンサが存在する対象の検査モデルに対して，片方のセンサ値が特定の範囲であれば，もう片方のセンサの取りうる範囲は限定される場合，本提案手法ではこの時の制約値を抽出することは困難である．そのため，これらのセンサ間の依存関係による不要なセンサ値領域を排除した，時系列変化の制約を抽出する手法の提案が今後の課題である．また，GWEU のワーストケースシナリオの分析にはドメイン知識を必要としているため，知識の詳細な分析および知識への依存を最小にする分析手法の提案も今後の課題である．

謝辞

本研究を行なうに当たり, 終始御指導を賜わった北陸先端科学技術大学院大学 情報科学研究科 落水浩一郎 教授 鈴木正人 准教授 に深謝いたします.

本研究において，初期のテーマ設定を含め，研究進展のポイントにおいて適切なアドバイスを頂きました国立情報学研究所 吉岡信和 准教授 に心より感謝申し上げます．オンライン電話での打ち合わせも含めて終始御指導いただきました.

本研究を開始する動機となったのは，国立情報学研究所 先端ソフトウェア工学 国際研究センターが主催する Top SE プロジェクトへの参加でした．本位田真一先生 はじめ関係各位の先生方に深謝いたします.

私が関西在住であり平日は会社勤務であることから，落水先生，鈴木先生，吉岡先生には休日をさいて頂いた上，東京や石川だけでなく名古屋や静岡にて研究の議論およびご指導をいただきました．おかげでさまで，本論文をまとめることができました．心より謝意を表します.

また，東京サテライトのスタッフの皆さまからも多大なるご支援をいただきました．心から御礼を申し上げます.

最後に，私の研究のために大きな協力と支援，そして励みを与えてくれた家族に最大の感謝を表します.

参考文献

[1] Armin, B., Alessandro C., Edmund, M. C., Ofer, S., and Yunshan, Z. :Bounded model checking. Advances in Computers, Vol. 58, pp. 118-149, 2003.

[2] Clarke, E. M. , Fehnker A. , Han Z. , Krogh, B. H. , Ouaknine J. , Stursberg O. , Theobald, M. :Abstraction and Counterexample-Guided Refinement in Model Checking of Hybrid Systems. Int. J. Found. Comput. Sci. (IJFCS) 14(4):583-604, 2003

[3] Alessandro, C., Edmund, M. C., Fausto, G., and Marco. R., "NuSMV: A new symbolic model checker," Int. J. Software Tools Technol. Transf. (STTT), vol. 2, no. 4, 2000

[4] Giannakopoulou D. , Pasareanu. C. S. , Barringer H. :Assumption Generation for Software Component Verification. ASE 2002:3-12

[5] Angelo, G. and Constance, H. Using model checking to generate tests from requirements specifications. In Proc. 7th European Engineering Conference held jointly with the 7th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pages 146-162. Springer-Verlag, 1999.

[6] A. van Lamsweerde, Requirements Engineering, Wiley, 2009.

[7] A. van Lamsweerde, Requirements Engineering in the Year 00: A Research Perspective. Invited Keynote Paper, Proc. ICSE '2000: 22nd Internrrtiorrul Conference on Sofhvtrre Enginerring, ACM Press, 2000, pp. 5-19.

[8] A van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", Invited minitutorial, Proc. of Int. Joint Conf. on Requirements Engineering, Toronto, 2001, pp.249-263

[9] A. Pnueli: "In transition from global to modular temporal reasoning about programs". In Logics and Models of Concurrent Systems, K. R.Apt, Ed. Nato Asi Series F: Computer And Systems Sciences, Springer-Verlag New York, vol. 13, pp. 123-144 (1985).

[10] Brendan. P. M., and Ian. J. H.. Using continuous real functions to model timed histories. In P. A. Bailes, editor, Proc. 6th Australian Software Engineering Conf. (ASWEC91), pages 257-270. Australian Comp. Soc., 1991.

[11] Brendan, P. M, and Ian, J. H.. A case study in timed refinement: A central heater. In Proc. BCS/FACS Fourth Refinement Workshop, Workshops in Computing, pages 138-149. Springer, January 1991.

[12] Brendan P. Mahony, Ian J. H.: A Case-Study in Timed Refinement: A Mine Pump. IEEE Trans. Software Eng. 18(9): 817-826 (1992)

[13] Bharadwaj, R. and Heitmeyer, C. Model checking complete requirements specifications using abstraction. Automated Software Eng. J., 6(1), January 1999.

[14] C. B. Jones: "Tentative steps towards a development method for interfering programs", ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 5, no. 4, pp. 596-619 (Oct. 1983).

[15] Alur R. , Dang T. , Ivancic F. :Counter-Example Guided Predicate Abstraction of Hybrid Systems. TACAS 2003:208-223

[16] Coleman, J. W. and Jones, C. B.: Examples of how to Determine the Specifications of Control Systems, In Proceedings of Workshop on Rigorous Engineering of Fault-Tolerant Systems (REFT 2005), pp.65-73, 2005.

[17] C. P. Pasareanu, M. B. Dwyer, and M. Huth. : “Assume-guarantee model checking of software : A comparative case study.”, In Theoretical and Practical Aspects of SPIN Model Checking. Springer-Verlag, (Sept. 1999). LNCS 1680.

[18] 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, N0. 8, pp. 2492–2507

[19] Choi, Y. and Heimdahl, M. “ Model Checking Software Requirement Specifications using Domain Reduction Abstraction,” in 18th IEEE International Conference on Automated Software Engineering (ASE 2003), 6.-10.X.2003. Montreal, Canada: IEEE.

[20] Daniel, J., Alloy: A Lightweight Object Modeling Notation. ACM Transaction on Software Engineering and Methodology (TOSEM) , 11(2):256-290, 2002.

[21] Dhaussy, P., Roger, J., and Boniol, F.: Reducing State Explosion with Context Modeling for Model-Checking. In 13th IEEE International High Assurance Systems Engineering Symposium (Hase’11), Boca Raton, USA, 2011.

[22] Dhaussy, P., Roger, J.,Leroux, L., and Boniol, F.: Context Aware Model Exploration with OBP tool to Improve Model-Checking. In Embedded Real Time Software and Systems (ERTSS’12), Toulouse, FRANCE, 2012.

[23] D. Giannakopoulou, C. S. Pasareanu, and H. Barringer: “Assumption Generation for Software Component Verification”. Proc. of 17th IEEE Int. Conf. on Automated Software Engineering (ASE), Edinburgh, UK, pp. 3-12 (Sept. 2002).

[24] Daniel, J., Ian, S. and Ilya S., Alcoa: the Alloy Constraint Analyzer Proc. International Conference on Software Engineering, Limerick, Ireland, 2000.

[25] Dhaussy, P., Boniol, F., Roger, J.-C., Leroux, L.: Improving model checking with context modelling. In: Advances in Software Engineering, ID 547157, 13 pages (2012)

[26] E. Clarke, . Grumberg, and D.Peled : Model Checking, MIT 1999

[27] E. Letier, " Reasoning about Agents in Goal-Oriented Requirements Engineering, " Universite Catholique de Louvain, 2001.

[28] http://www.etrobo.jp/.

[29] E. Yu. " Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering ", Proc. of RE '97, 1997

[30] Formal Systems(Europe), "FDR2 User Manual," http://fsel.com/documentation/fdr2/html/fdr2manual.html,2005

[31] Holzmann, G :" The SPIN Model Checker ", 2004, Addison-Wesley.

[32] Dardenne, A and Lamsweerde, A. V. and Fickas S. :Goal-Directed Requirements Acquisition, Science of Cmputer Programming. 20(1-2):3-50, 1993.

[33] Graf, S and Saidi, H. : Construction of abstract state graphs with PVS. LNCS, 1254:pp72-83, 1997.

[34] Hayes, I., Jackson, M., and Jones, C.: Determining the specification of a control system from that of its environment, In International Symposium for Formal Methods Europe (FME 2003), pp.154-169, 2003.

[35] Jackson, D., and Damon, C. 1996. Elements of style: Analyzing a software design feature with a counter example detector. IEEE Transactions on Software Engineering 22(7):484-495.

[36] Magee, J. and Kramer, J. :" Concurrency: State Models and Java Programming ", 2006, John Wiley and Sons.

[37] J.M. Cobleigh, D. Giannakopoulou, and C.S. Pasareanu: “Learning assumptions for compositional verification”. Proc. of 9th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), pp. 331-346, (Apr. 2003).

[38] McMillan, K. :“ Symbolic model Checking ”, 1993, Springer.

[39] 結縁祥治, : 通信プロセスモデルと形式意味論に基づくソフトウェアのモデル化, コンピュータ・ソフトウェア, Vol. 22, No.2, pages 22–43, 2005.

[40] M. A. Jackson. Problem Frames: Analyzing and structuring software development problems. Addison-Wesley, 2000.

[41] 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 .

[42] Mylopoulos, J., Chung, L. and Nixon, B., “Representing and Using Non-Functional Requirements: A Process-Oriented Approach”, IEEE Transactions on Software Engineering, Vol.18, No.6, pp.483?497, 1992.

[43] Michael A. J.: Software requirements and specifications - a lexicon of practice, principles and prejudices. Addison-Wesley 1995, ISBN 978-0-201-87712-0, pp. I-XVI, 1-228.

[44] Mylopoulos, J., Chung, L. and Yu, E., “From Object-Oriented to Goal-Oriented Requirements Analysis”, Communications of the ACM, Vol. 42, No. 1, pp. 31-37, 1999.

[45] Michael. A. J.. Problem analysis and structure. In Tony Hoare, Manfred Broy, and Ralf Steinbruggen, editors, Engineering Theories of Software Construction (Proceedings of the NATO Summer School, Marktoberdorf, August 2000). IOS Press, 2000.

[46] Michael A. J.: Problem Frames - Analysing and Structuring Software Development Problems. Pearson Education 2000, ISBN 978-0-2015-9627-4, pp. I-XIX, 1-390.

[47] 中島震 : ソフトウェア工学の道具としての形式手法. ソフトウェアエンジニアリング最前線 2007, 近代科学社, 2007, pp. 27–48, (改訂版，NII-2007-007J, July 2007).

[48] Paul, A. Paul, B, and William, M. Using model checking to generate tests from specifications. In Proceedings of the 2nd IEEE International Conference on Formal Engineering Methods, 1998.

[49] 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.

[50] 組込みソフトウェア管理者・技術者育成研究会 :http://www.sessame.jp/.

[51] 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).

[52] 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).

[53] 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).

[54] 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).

[55] Thayer, R.H. and Dorfman, M. :Software Requirements Engineering 2nd Edition, IEEE Computer Society (1997).

[56] Ubayashi, N., Kamei, Y., Hirayama, M. and Tamai, T.: A Context Analysis Method for Embedded Systems Exploring a Requirement Boundary between

a System and Its Context. In Proceedings of the 19th International IEEE Requirements Engineering Conference ,RE'11. (2011)

[57] Weiser, M. 1984. Program slicing. IEEE Transactions on Software Engineering, SE-10(4):352-357.

[58] Yu, E., "Models for Supporting the Redesign of Organizational Work", Proc. 1st IEEE International Symposium on Requirements Engineering (RE '93), pp.34?41, Jan. 1993.

[59] Yu, E., "Modelling Organizations for Information Systems Requirements Engineering", Proc. Conference on Organizational Computing Systems (COCS '95), pp.225?236, Aug. 1995.

[60] Yu, E., "Towards Modelling and Reasoning Support for Early-phase Requirements Engineering", Proc. 3rd IEEE International Symposium on Requirements Engineering (RE '97), pp.226?235, Jan. 1997.

[61] 山本修一郎, ゴール指向によるシステム要求管理技法, ソフトリサーチセンター, 2007.

付録

A.1 ライントレーサの検査モデル

検査モデル A.1: etrb_pid_limit5.pml

```
1  chan light = [0] of { int }
2
3  /*********************************************/
4  /*****************外部環境記述*****************/
5  /*********************************************/
6  /* 今回と前回の差分 */
7  #define DIFF (color0-color1)
8
9  active proctype external()
10 {
11   /* ライントレーサが取得する光センサ値の初期値 */
12   int color0 = 500;
13   /* 前回の送信値 */
14   int color1 = 500;
15   /* 時系列変化の制約 */
16   int limit = 5;
17
18   /* ライントレーサが取得する光センサ値 */
19   /* (500〜700)の全組合せパターンを */
20   /* 構築する. */
21   /* 今回の送信値が前回の送信値より */
22   /* 差分がlimitを超えると変更させない */
23   do
24   ::do
25     ::if
26       ::(500<color0)->color0--;
27       ::(700>color0)->color0++;
28       ::break;
```

```
29          fi;
30          /* 時系列変化の制約による判定 */
31          if
32          ::(DIFF>=0)&&(DIFF>=limit)->break;
33          ::(DIFF<0)&&(DIFF<=(limit*-1))->break;
34          ::else;
35          fi;
36        od;
37        /* システムモデルへ送信 */
38        light!color0;
39        /* 今回の送信値を前回の送信値として更新 */
40        color1 = color0;
41      od;
42  }
43
44
45  /***********************************************/
46  /******************システム記述*****************/
47  /***********************************************/
48
49  /* 設計した白黒の閾値 */
50  #define TH 575
51
52  /* P成分計算(36はP成分係数) */
53  /* P成分変数:ラインとライントレーサの進行 */
54  /* 方向の角度 (光センサ値の今回と前回の取得値の差を角度) */
55  #define P_VALUE ((color0-color1)*36)
56
57  /* I成分計算(1はI成分係数) */
58  /* I成分変数:ラインとライントレーサが離れ */
59  /* た距離 (光センサ値の前々回・前回・今回の */
60  /* それぞれの値と,閾値との差の積分)*/
61  #define I_VALUE ((color0-TH)*1)
62
63  /* D成分計算(4はD成分係数) */
64  /* D成分変数:ラインとライントレーサとの */
65  /* 角速度 (光センサ値の,前々回・前回と前回・今回の角度の差) */
66  #define D_VALUE (((color0-color1)-(color1-color2))*4)
67
```

```
68  /* 旋回値決定 */
69  #define POWER (P_VALUE + I_VALUE + D_VALUE)
70
71  active proctype system()
72  {
73    /* 今回取得した光センサ値変数 (X^0) */
74    int color0 = 0;
75    /* 旋回値変数 (Y^0) */
76    int turn0 = 0;
77    /* 前回の旋回値変数 (Y^1) */
78    int turn1 = 0;
79    /* 前回取得した光センサ値変数 (X^1) */
80    int color1 = 0;
81    /* 前々回取得した光センサ値変数 (X^2) */
82    int color2 = 0;
83
84    /* 外部環境モデルから受信 */
85    do::light?color0;
86
87      /* 旋回値決定 */
88      turn0 = POWER;
89
90      /* 検査する性質 */
91      /* 今回のセンサ値と閾値の差が前回のセンサ値と */
92      /* 閾値の差より小さければ,モータへのパワー値は */
93      /* 前回より必ず小さい.」*/
94      if
95      ::!(((color0-TH)<(color1-TH))&&(turn0<turn1))->assert(false);
96      ::else;
97      fi;
98
99      d_step{
100     /* 前々回取得した光センサ値を更新 */
101     color2 = color1;
102     /* 前回取得した光センサ値を更新 */
103     color1 = color0;
104     /* 前回旋回値を更新 */
105     turn1 = turn0;
106     }
```

```
107   od;
108 }
```

検査モデル A.2: etrb_pid_limit3.pml

```
1  chan light = [0] of { int }
2
3  /*********************************************/
4  /*****************外部環境記述*****************/
5  /*********************************************/
6  /* 今回と前回の差分 */
7  #define DIFF (color0-color1)
8
9  active proctype external()
10 {
11   /* ライントレーサが取得する光センサ値の初期値 */
12   int color0 = 500;
13   /* 前回の送信値 */
14   int color1 = 500;
15   /* 時系列変化の制約 */
16   int limit = 3;
17
18   /* ライントレーサが取得する光センサ値 */
19   /* (500〜700)の全組合せパターンを */
20   /* 構築する. */
21   /* 今回の送信値が前回の送信値より */
22   /* 差分がlimitを超えると変更させない */
23   do
24   ::do
25     ::if
26       ::(500<color0)->color0--;
27       ::(700>color0)->color0++;
28       ::break;
29       fi;
30       if
31       ::(DIFF>=0)&&(DIFF>=limit)->break;
32       ::(DIFF<0)&&(DIFF<=(limit*-1))->break;
33       ::else;
34       fi;
35     od;
```

```
36      /* システムモデルへ送信 */
37      light!color0;
38      /* 今回の送信値を前回の送信値として更新 */
39      color1 = color0;
40    od;
41  }
```

検査モデル A.2 のシステム記述は，検査モデル A.1 と同じ．

検査モデル A.3: etrb_pid_get_3times.pml

```
1  chan light = [0] of { int }
2
3  /*************************************************/
4  /*****************外部環境記述*****************/
5  /*************************************************/
6  active proctype external()
7  {
8    /* ライントレーサが取得する光センサ値の初期値 */
9    int color0 = 500;
10
11   /* ライントレーサが取得する光センサ値 */
12   /* (500〜700)の全組合せパターンを構築する */
13   int loop = 3; /* color0〜color2 の 3つを対象 */
14   do
15   ::do
16     ::(500<color0)->color0--;
17     ::(700>color0)->color0++;
18     ::break;
19     od;
20     /* システムモデルへ送信 */
21     light!color0;
22     loop--; if ::loop==0->break; ::else->skip; fi;
23   od;
24 }
```

検査モデル A.3 のシステム記述は，検査モデル A.1 と同じ．

検査モデル A.4: etrb_pid_get_3times_other_research.pml

```
1  chan light = [0] of { int }
2
```

```
3  /*********************************************/
4  /*****************外部環境記述*****************/
5  /*********************************************/
6  active proctype external()
7  {
8    /* ライントレーサが取得する光センサ値の初期値 */
9    int color0 = 500;
10   bool bInit = true;
11   /* ライントレーサが取得する光センサ値 */
12   /* (500〜700)の全組合せパターンを構築する */
13   int loop = 3; /* color0〜color2 の 3つを対象 */
14   do
15   ::if
16     ::bInit->color0=700;bInit=false;
17     ::else->
18       do
19       ::(500<color0)->color0--;
20       ::(700>color0)->color0++;
21       ::break;
22       od;
23       /* システムモデルへ送信 */
24       light!color0;
25     fi;
26     loop--; if ::loop==0->break; ::else->skip; fi;
27   od;
28 }
29
30 /*********************************************/
31 /*****************システム記述*****************/
32 /*********************************************/
33
34 /* 設計した白黒の閾値 */
35 #define TH 575
36
37 /* P成分計算(36はP成分係数) */
38 /* P成分変数:ラインとライントレーサの進行 */
39 /* 方向の角度 (光センサ値の今回と前回の取得値の差を角度) */
40 #define P_VALUE ((color0-color1)*36)
41
```

```
42 /* I成分計算(1はI成分係数) */
43 /* I成分変数:ラインとラインとレーサが離れ */
44 /* た距離 (光センサ値の前々回・前回・今回の */
45 /* それぞれの値と,閾値との差の積分)*/
46 #define I_VALUE ((color0-TH)*1)
47
48 /* D成分計算(4はD成分係数) */
49 /* D成分変数:ラインとラインとレーサとの */
50 /* 角速度 (光センサ値の,前々回・前回と前回・今回の角度の差) */
51 #define D_VALUE (((color0-color1)-(color1-color2))*4)
52
53 /* 旋回値決定 */
54 #define POWER (P_VALUE + I_VALUE + D_VALUE)
55
56 active proctype system()
57 {
58   /* 今回取得した光センサ値変数 (X^0) */
59   int color0 = 0;
60   /* 旋回値変数 (Y^0) */
61   int turn0 = 0;
62   /* 前回の旋回値変数 (Y^1) */
63   int turn1 = 0;
64   /* 前回取得した光センサ値変数 (X^1) */
65   int color1 = 0;
66   /* 前々回取得した光センサ値変数 (X^2) */
67   int color2 = 0;
68
69   /* 外部環境モデルから受信 */
70   do::light?color0;
71
72     /* 旋回値決定 */
73     turn0 = POWER;
74
75     /* 検査する性質 */
76     /* 今回のセンサ値と閾値の差が前回のセンサ値と */
77     /* 閾値の差より小さければ,モータへのパワー値は */
78     /* 前回より必ず小さい.」*/
79     if
80     ::!(((color0-TH)<(color1-TH))&&(turn0<turn1))->assert(false);
```

```
81      ::else;
82      fi;
83
84      /* 前々回取得した光センサ値を更新 */
85      color2 = color1;
86      /* 前回取得した光センサ値を更新 */
87      color1 = color0;
88      /* 前回旋回値を更新 */
89      turn1 = turn0;
90    od;
91  }
```

A.2 話題沸騰ポットの検査モデル

検査モデル A.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{
10     if
11     ::(t0>=1500)->t0=1500;
12     ::else->t0++;
13     fi;
14     }
15   ::d_step{
16     if
17     ::(t0<=-100)->t0=-100;
18     ::else->t0--;
19     fi;
20     }
21   :: break;
22   od;
```

```
23 }
24
25 /*** 状態 ****/
26 /* ステートタスク */
27 mtype = {s_idle,s_warming,s_boiling}
28 /* ヒータ制御タスク */
29 mtype = {s_off,s_pid,s_heating}
30 /* 蓋監視タスク */
31 mtype = {s_opn_cvr,s_cls_cvr}
32
33 /*** イベント ***/
34 mtype = {
35     m_stopWrmCtrl /* 温度制御停止 */
36    ,m_compBoiling /* 沸騰処理完了 */
37    ,m_reqBoiling /* 沸騰要求 */
38    ,m_doBoiling /* 沸騰行為 */
39    ,m_doWarming /* 保温行為 */
40    ,m_stopDoBoiling /* 沸騰行為停止 */
41    ,m_stopDoWarming /* 保温行為停止 */
42    ,m_setTemp98 /* 温度設定(98℃)*/
43    ,m_setTemp90 /* 温度設定(90℃)*/
44    ,m_setTemp60 /* 温度設定(60℃)*/
45    ,m_tempSetBtnOn /* 温度設定ボタンON */
46    }
47
48 /* メッセージ */
49 mtype = { st,ht,tm,ft }
50 chan msg = [5] of { mtype,mtype }
51
52 /***************/
53 /* システム記述 */
54 /***************/
55
56 /*** タスク定義 ***/
57 /* ステートタスク */
58 active proctype st_tsk()
59 {
60   mtype state = s_idle;
61   mtype event;
```

```
62    do
63    ::d_step{
64      msg?st(event)->
65      if
66      ::(state==s_idle)->
67        if
68        ::(event==m_reqBoiling)->
69          state=s_boiling;
70          msg!ht(m_doBoiling);
71        ::else->skip;
72        fi;
73      ::(state==s_warming)->
74        if
75        ::(event==m_stopWrmCtrl)->
76          msg!ht(m_stopDoWarming);
77          state=s_idle;
78        ::(event==m_reqBoiling)->
79          state=s_boiling;
80          msg!ht(m_doBoiling);
81        ::else->skip;
82        fi;
83      ::(state==s_boiling)->
84        if
85        ::(event==m_stopWrmCtrl)->
86          msg!ht(m_stopDoBoiling);
87          state=s_idle;
88        ::(event==m_compBoiling)->
89          state=s_warming;
90          msg!ht(m_doWarming);
91        ::else->skip;
92        fi;
93      fi;
94      }
95    od;
96  }
97
98  /* 温度捜査量の算出 */
99  inline calcHeatPower(tg,t0,t1,t2,m)
100 {
```

```
101     int dm;
102     int m0;
103     int m1;
104     getTemp(t0); /*温度取得*/
105     m1=m;
106     d_step{
107     dm=(t1-t0)+(tg-t0)+(2*t2-t0-t2);
108     m0=m1+dm;
109     if
110     :: (m0<=0)->m0=0; /*加熱しない*/
111     :: (m0>=100)->m0=100; /*加熱する*/
112     :: else->skip; /*現状維持*/
113     fi;
114     t2=t1;
115     t1=t0;
116     m=m0;
117     }
118 }
119
120 /* ヒータ制御タスク */
121 active proctype ht_tsk()
122 {
123     mtype state = s_off;
124     mtype event;
125     int tg=98;
126     int t0=0; /* X^0 */
127     int t1=0; /* X^1 */
128     int t2=0; /* X^2 */
129     int m0,m1,m2; /* Y^0,Y^1,Y^2 */
130
131     do
132     ::msg?ht(event)->
133       d_step{
134       if
135       ::(event==m_setTemp60)->tg=60;
136       ::(event==m_setTemp90)->tg=90;
137       ::(event==m_setTemp98)->tg=98;
138       ::else->
139         if
```

```
140         ::(state==s_off)->
141           if
142           ::(event==m_doWarming)->
143             state=s_pid;
144           ::(event==m_doBoiling)->
145             state=s_heating;
146           ::else->skip;
147           fi;
148         ::(state==s_pid)->
149           if
150           ::(event==m_stopDoWarming)->
151             state=s_off;
152           ::else->skip;
153           fi;
154         ::(state==s_heating)->
155           if
156           ::(event==m_stopDoBoiling)->
157             state=s_off;
158           ::else->skip;
159           fi;
160       fi;
161     fi;
162     }
163   ::(state==s_heating)->
164     if
165     ::calcHeatPower(tg,t0,t1,t2,m0);
166       if
167       ::!(((t0!=t2))&&(m0>m2))->assert(false);
168       ::else->m1=m0;m2=m1;
169       fi;
170     ::state=s_off;
171       msg!st(m_compBoiling);
172     fi;
173   od;
174 }
175
176 /* 蓋監視タスク */
177 active proctype ft_tsk()
178 {
```

```
179   mtype state;
180   mtype preState=s_cls_cvr;
181
182   do
183   ::if
184     ::state=s_opn_cvr;
185     ::state=s_cls_cvr;
186     fi;
187     if
188     ::(preState==s_cls_cvr)&&(state==s_opn_cvr)->
189       msg!st(m_reqBoiling);
190     ::(preState==s_opn_cvr)&&(state==s_cls_cvr)->
191       msg!st(m_stopWrmCtrl);
192     ::else->skip;
193     fi;
194     preState=state;
195   od;
196 }
197
198 /* 保温設定ボタン監視タスク */
199 active proctype tm_tsk()
200 {
201   do
202   ::if
203     ::msg!ht(m_setTemp60);
204     ::msg!ht(m_setTemp90);
205     ::msg!ht(m_setTemp98);
206     fi;
207   od;
208 }
```

検査モデル A.6: pot_abst.pml

```
1  /* 外部環境記述 */
2  inline getTemp(t0)
3  {
4    do
5    ::if
6      ::t0++;
7      ::t0--;
```

```
8       fi;
9       if
10      ::(t0>=150)->t0=150;
11      ::(t0<=-10)->t0=-10;
12      ::else->skip;
13      fi;
14    :: break;
15    od;
16  }
17
18  /* システム記述 */
19  active proctype ht_tsk()
20  {
21    int tg=98;
22    int t0=0; /* X^0 */
23    int t1=0; /* X^1 */
24    int t2=0; /* X^2 */
25    int dm;
26    int m0; /* Y^0 */
27    int m1; /* Y^1 */
28    int m2; /* Y^2 */
29    do
30    ::getTemp(t0);
31      m1=m0;
32      dm=(t1-t0)+(tg-t0)+(2*t2-t0-t2);/* PID control */
33      m0=m1+dm;
34      if
35      ::(m0<=0)->m0=0; /* no heat */
36      ::(m0>=100)->m0=100; /* heat */
37      ::else->skip; /* on going */
38      fi;
39      if
40      ::!((t0!=t2)&&(m0<m2))->assert(false);
41      ::else->m1=m0;m2=m1;
42      fi;
43      t2=t1;
44      t1=t0;
45    od;
46  }
```

検査モデル A.7: pot_abst_limit2.pml

```
1  /* 外部環境記述 */
2  inline getTemp(t0)
3  {
4    int loop; loop = 2;
5    do
6    ::if
7      ::(loop==0)->break;
8      ::else->skip;
9      fi;
10     if
11     ::t0++;
12     ::t0--;
13     fi;
14     if
15     ::(t0>=150)->t0=150;
16     ::(t0<=-10)->t0=-10;
17     ::else->skip;
18     fi;
19     loop--;
20   :: break;
21   od;
22 }
23
24 /* システム記述 */
25 active proctype ht_tsk()
26 {
27   int tg=98;
28   int t0=0; /* X^0 */
29   int t1=0; /* X^1 */
30   int t2=0; /* X^2 */
31   int dm;
32   int m0; /* Y^0 */
33   int m1; /* Y^1 */
34   int m2; /* Y^2 */
35   do
36   ::getTemp(t0);
37     m1=m0;
38     dm=(t1-t0)+(tg-t0)+(2*t2-t0-t2);/* PID control */
```

```
39      m0=m1+dm;
40      if
41      ::(m0<=0)->m0=0; /* no heat */
42      ::(m0>=100)->m0=100; /* heat */
43      ::else->skip; /* on going */
44      fi;
45      if
46      ::!((t0!=t2)&&(m0<m2))->assert(false);
47      ::else->m1=m0;m2=m1;
48      fi;
49      t2=t1;
50      t1=t0;
51    od;
52  }
```

検査モデル A.8: pot_abst_limit1.pml

```
1   /* 外部環境記述 */
2   inline getTemp(t0)
3   {
4     int loop; loop = 1;
5     do
6     ::if
7       ::(loop==0)->break;
8       ::else->skip;
9       fi;
10      if
11      ::t0++;
12      ::t0--;
13      fi;
14      if
15      ::(t0>=150)->t0=150;
16      ::(t0<=-10)->t0=-10;
17      ::else->skip;
18      fi;
19      loop--;
20    :: break;
21    od;
22  }
23
```

```
/* システム記述 */
active proctype ht_tsk()
{
  int tg=98;
  int t0=0; /* X^0 */
  int t1=0; /* X^1 */
  int t2=0; /* X^2 */
  int dm;
  int m0; /* Y^0 */
  int m1; /* Y^1 */
  int m2; /* Y^2 */
  do
  ::getTemp(t0);
    m1=m0;
    dm=(t1-t0)+(tg-t0)+(2*t2-t0-t2);/* PID control */
    m0=m1+dm;
    if
    ::(m0<=0)->m0=0; /* no heat */
    ::(m0>=100)->m0=100; /* heat */
    ::else->skip; /* on going */
    fi;
    if
    ::!((t0!=t2)&&(m0<m2))->assert(false);
    ::else->m1=m0;m2=m1;
    fi;
    t2=t1;
    t1=t0;
  od;
}
```

検査モデル A.9: pot_abst_get_3times.pml

```
/* 外部環境記述 */
inline getTemp(t0)
{
  do
  ::if
    ::t0++;
    ::t0--;
    fi;
```

```
9       if
10      ::(t0>=150)->t0=150;
11      ::(t0<=-10)->t0=-10;
12      ::else->skip;
13      fi;
14    :: break;
15    od;
16  }
17
18  /* システム記述 */
19  active proctype ht_tsk()
20  {
21    int tg=98;
22    int t0=0; /* X^0 */
23    int t1=0; /* X^1 */
24    int t2=0; /* X^2 */
25    int dm;
26    int m0; /* Y^0 */
27    int m1; /* Y^1 */
28    int m2; /* Y^2 */
29    int tLoop = 3; /* t0〜t2 の 3つを対象 */
30    do
31    ::getTemp(t0);
32      m1=m0;
33      dm=(t1-t0)+(tg-t0)+(2*t2-t0-t2);/* PID control */
34      m0=m1+dm;
35      if
36      ::(m0<=0)->m0=0; /* no heat */
37      ::(m0>=100)->m0=100; /* heat */
38      ::else->skip; /* on going */
39      fi;
40      if
41      ::!((t0!=t2)&&(m0<m2))->assert(false);
42      ::else->m1=m0;m2=m1;
43      fi;
44      t2=t1;
45      t1=t0;
46      tLoop--; if ::tLoop==0->break; ::else->skip; fi;
47    od;
```

```
48 }
```

検査モデル A.10: pot_abst_get_3times_other_research.pml

```
1  /* 外部環境記述 */
2  inline getTemp(t0)
3  {
4    do
5    ::if
6      ::t0++;
7      ::t0--;
8      fi;
9      if
10     ::(t0>=150)->t0=150;
11     ::(t0<=-10)->t0=-10;
12     ::else->skip;
13     fi;
14   :: break;
15   od;
16 }
17
18 /* システム記述 */
19 active proctype ht_tsk()
20 {
21   int tg=98;
22   int t0=0; /* X^0 */
23   int t1=0; /* X^1 */
24   int t2=0; /* X^2 */
25   int dm;
26   int m0; /* Y^0 */
27   int m1; /* Y^1 */
28   int m2; /* Y^2 */
29   int tLoop = 3; /* t0〜t2 の 3つを対象 */
30   bool bInit=true;
31   do
32   ::if
33     ::bInit->t0=0;bInit=false;
34     ::else->
35       getTemp(t0);
36       m1=m0;
```

```
37        dm=(t1-t0)+(tg-t0)+(2*t2-t0-t2);/* PID control */
38        m0=m1+dm;
39        if
40        ::(m0<=0)->m0=0; /* no heat */
41        ::(m0>=100)->m0=100; /* heat */
42        ::else->skip; /* on going */
43        fi;
44        if
45        ::!((t0!=t2)&&(m0<m2))->assert(false);
46        ::else->m1=m0;m2=m1;
47        fi;
48        t2=t1;
49        t1=t0;
50      fi;
51      tLoop--; if ::tLoop==0->break; ::else->skip; fi;
52    od
53  }
```

本研究に関する発表論文

[1] 乾 道孝，吉岡 信和: “モデル検査基準を用いたモデル検査プロセスについて”，電子情報通信学会技術報告，CPSY09-2，DC09-2(平成 21 年 4 月).

[2] 乾 道孝，吉岡 信和，落水 浩一郎: “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”，情報処理学会論文誌 52(12)，3205-3220，2011-12-1.

[3] M.Inui, N.Yoshioka, K.Ochimizu: “A State Reduction Method for Model-checking by Eliminating Unnecessary Contexts”, IIAI AIT ’13. IIAI International Conference on Advanced Information Technologies (ICASEIS) 採録.