情報処理学会論文誌 Vol. 49 No. 6 2129–2143 (June 2008)

ホームネットワークシステムにおける競合問題の検証

松尾尚文[†1] パッタラ リーラープルット[†2]
土屋達弘[†1] 菊野亨[†1]

近年，家電機器をネットワークで接続して実現するホームネットワークシステムが利用され始めている．ホームネットワークシステムでは，ネットワークに接続された複数の家電機器を協調させ，様々なサービスの提供が可能となる．しかし，単体ではサービスが正しく動作しても，複数のサービスを並行に動作させると予期せぬ動作をしてしまう場合がある．これは競合問題と呼ばれる．本論文では，ホームネットワークシステムにおいて発生する競合問題の検出方法について検討する．具体的には，競合を定義するためのホームネットワークシステムのモデルを導入する．また，導入したモデルをもとに，モデル検査を用いて競合を自動検出する手法を提案する．さらに，サービスの例に対し提案法を実験的に適用することにより，その有効性を示す．

Verifying Feature Interactions in Home Network Systems

Takafumi Matsuo,[†1] Pattara Leelaprute,[†2]
Tatsuhiro Tsuchiya[†1] and Tohru Kikuno[†1]

As home appliances are becoming increasingly interconnected, the use of home network systems is being expanded. Home network systems provide value-added services by integrating the features of different appliances. Concurrent execution of these services, however, can cause unexpected behaviors of the system, even when each service is independently correct. This problem is called the feature interaction problem. In this paper, we propose an approach for detecting feature interactions in home network systems. Specifically, we propose a model of home network systems that is specialized for specifying feature interactions and devise a method of automatically detecting interactions by means of model checking. To demonstrate the usefulness of the proposed approach, we show the results of an experiment using an example of practical services.

1. はじめに

近年，家電機器をネットワークで接続して実現するホームネットワークシステムが活発に利用され始めている[6),14),16),21)]．ホームネットワークシステムでは，ネットワークに接続された複数の家電機器を協調させ，様々なサービスの提供が可能となる．たとえば，煙センサと換気扇を利用して，煙を自動的に検出・除去する煙除去サービスや，エアコンと換気扇を利用し，外気を取り込むことで空調の効率化を図る HVAC（Heating, Ventilation and Air-conditioning）サービス等がある．

しかし，単体ではサービスが正しく動作しても，複数のサービスを並行に動作させると予期せぬ動作をしてしまう場合がある．上記のサービスの例においても，この問題は発生する．煙除去サービスが室内の煙を取り除くために換気扇を回していたとする．ここで，HVAC サービスが冷房をかけ，空調の邪魔になると判断し，換気扇を止めた場合，室内の煙が除去されることなく，換気扇が止められてしまう．これは競合問題[7)]と呼ばれる．

複数のサービスを並行して実行した場合，それらに含まれる命令の実行順序により，システムは非常に多くの実行パターンを持つ．そのために，テスト等によって競合を検出することは困難である．こうした複雑な動作をするシステムの正しさを保証する方法としてモデル検査[5)]がある．

モデル検査とは，検証対象となるシステムの動作を有限状態空間で表現し，その有限状態空間内を探索して，検証項目である性質が満たされるかどうかを確認する手法である．モデル検査を用いることにより，自動的・網羅的な検証が可能となる．そのため，多くの実行パターンを持つ複数のサービスの動作の検証に，モデル検査は非常に有効である．また，性質が満たされなかった場合，反例を得ることができる．反例を分析することにより，どのような実行の結果，システムが性質を満たさなくなったのかが調べられる．

本論文では，ホームネットワークシステム上のサービス間の競合問題をモデル検査を用いることで検出する方法について述べる．具体的な貢献は以下の 2 つに要約される．

- ホームネットワークシステムと競合の表現に適したモデルを提案する．このモデルを用いることにより，ホームネットワークシステムの競合問題に特徴的な環境および環境へ

†1 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
†2 カセットサート大学（タイ）工学部情報科学科
Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Thailand

の複数の異なった影響を扱うことができる．また，このモデル上での競合を明らかにし，それらを形式的に定義した．提案したモデルはサービスレベルにのみ注目しているため，システムがどのようなネットワークプロトコルを用いて実装されているかにかかわらず適用可能である．

- モデルの記述からの自動的な競合の検出を実現する．そのため，モデルの記述をモデル検査器 SPIN[8] の入力へと変換する手法を提案し，それを実現するプログラムを実装した．また，各競合を検出するための性質を明らかにした．これにより，提案モデルを用いたシステムの記述から，自動的に競合の検出が可能となる．

ホームネットワークシステムは通常，温度等の環境要素と直接的な相互作用を有するため，競合を検出するためには，それぞれの環境要素に対して相反する動作をしていないかを調べる必要がある．システムの状態遷移とは異なり，このような環境要素の状態は，機器の影響を受けつつ，漸次的に変化するという性質を持つ．これまで競合問題は，電話通信や工場の自動制御等の分野で広く扱われてきた[7),9)]．たとえば文献 2) では，SPIN を用いてシステムの状態を変更する複数の命令間で矛盾があった場合を競合として検出する手法が提案されている．しかし，このような従来の競合検出手法では，上記の性質を有する環境に関連した競合を取り扱うことができなかった．本研究では，機器の環境への影響を考慮したモデルを提案し，そのモデルに対しモデル検査を適用することで，環境に関する競合の検出を可能にしている．

SMV[12)] や SAL[1)] 等の多くのモデル検査器では，検証対象のシステムの動作を，状態遷移を直接記述することで表現する．そのため，サービス記述に現れる，一連の手続き的処理を表現することが困難である．本研究では，モデル検査器として SPIN を用いることで，この問題を解決している．SPIN の入力言語は手続き的言語である Promela であり，これにより，提案モデルに従って表現されたサービスを，自動的に Promela プログラムへと変換することが可能になっている．

本論文の構成を以下に示す．まず，2 章でホームネットワークシステムおよびサービスについて具体例を用いて説明する．3 章で提案するシステムモデルを導入し，そのモデル上での競合の定義を 4 章に示す．5 章でモデルをもとに検証を行うための方法を示す．提案手法の有効性を示すために行った実験の結果を 6 章に示す．7 章で本研究と関連研究の位置づけについて述べる．8 章で本論文をまとめる．

2. 準　　備

2.1 ホームネットワークシステム

本論文中で用いるホームネットワークの例を**図 1** に示す．図 1 のシステムは，エアコン，換気扇，煙センサ，温度計（室内と室外），DVD プレイヤ，テレビ，電灯，ブラインド，照度計の計 10 個の機器と 1 つのサーバから構成されている．

2.2 サ ー ビ ス

ホームネットワークシステムでは複数の機器を連携させることにより，有用なサービスを実現する．本論文では，文献 10) の例をもとにして，以下の 4 つのサービスを考える．

HVAC サービス：部屋の空調管理を効率的に行うためのサービスで，エアコン，換気扇，温度計を用いる．ここでは，冷房をかける機能に注目する．エアコンの設定温度より室温が高い場合にエアコンで冷房をかける．ここで室外の温度が室温よりも低いと，換気扇を回し，涼しい外気を取り込むことにより，冷房の効率化を図る．

煙除去サービス：煙センサと換気扇を連動させて室内の煙を自動的に除去するサービスで，煙センサが煙を検知すると，換気扇の電源を入れる．そして，煙がないことが確認されるまで，換気扇を回し続ける．煙が検知されなくなれば換気扇を止める．

ホームシアタサービス：DVD を見る際に部屋の明かりを調整するサービスで，テレビ，DVD プレイヤ，照度計，ライト，ブラインドを用いる．DVD の電源が入れられた際に，テレビの電源をつけ，チャネルを DVD 用に設定する．同時に，ブラインドを閉め，部屋の明かるさが設定されたレベルになるように，ライトを調節する．

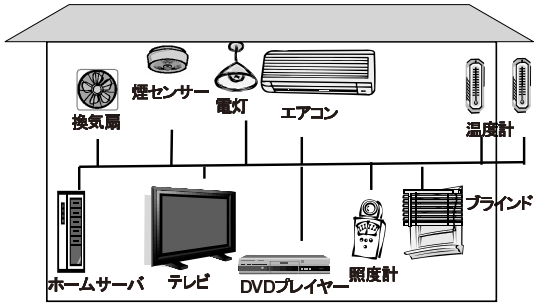

図 1　ホームネットワークシステム
Fig. 1　Home network system.

節電サービス：これは不要な家電機器の電源を切るサービスである．たとえば，テレビの電源が切れている状態で DVD プレイヤの電源が入っていると，DVD プレイヤの電源を切る．

3. ホームネットワークシステムのモデル

この章ではホームネットワークシステムのモデルを提案する．ホームネットワークシステムは環境，機器，サービスの 3 つの要素から構成される．

3.1 環　　境

環境を構成する温度や明るさ等の各要素 $E = (e, Eff)$ は，その状態を表す 1 つの変数 e と影響の集合 Eff から構成される．

表 1 に本論文中で用いる環境のモデルを示す．Temp_in，Temp_out はそれぞれ，室温と室外の温度を表している．また，Smoke は室内の煙の有無を表し，Brightness は部屋の明るさを表す．Temp_in の変数 temp は現在の室温を表している（たとえば temp=28°C）．また，室温への影響の種類は $\{up, down\}$ であり，温度の上昇と下降の 2 種類の影響があることを示している．

3.2 機　　器

機器 $A = (F, M, EW)$ は，その状態を表す変数の集合 F と，動作を表すメソッドの集合 M，環境の要素への影響の有無 EW からなる．

メソッドは後述のサービスから呼び出される．メソッド呼び出しの際には引数が与えられる場合がある．メソッドは，環境の要素の変数の読み込み，自身の機器の変数の読み込み・書き込みを行う．また，実行後に，呼び出し元のサービスに返り値を返すこともできる．

各メソッド $m \in M$ には，その実行前に成立すべき事前条件 $Pre(m)$ と，実行後に成立すべき事後条件 $Post(m)$ が関連付けられる．事前条件および事後条件は，機器の状態に関する述語である．メソッド m の実行開始時に，事前条件 $Pre(m)$ が満たされていない場合は，メソッドの動作は不定となる．

表 1　環境のモデル
Table 1　Environment model.

環境の要素	変数	変数の型	影響の種類
Temp_in	temp	int	{up, down}
Temp_out	temp	int	{up, down}
Smoke	smoke	{not-exist, exist}	{occurrence, removal}
Brightness	brightness	{High, Middle, Low}	{up, down}

機器が環境へ与える影響は，機器の状態と環境の状態に依存して決まる．環境の要素への影響の有無 EW は，環境の要素への影響 $eff \in Eff$ に対して，機器および環境がどのような状態であれば，影響を有するかを規定する．つまり，$EW(eff)$ は，機器および環境の状態に関する述語であり，この述語が成り立つ場合に，機器は環境の要素に影響 eff を与える．

例として換気扇のモデルを考える．換気扇は変数として電源の状態を表す `Power` を持つ．メソッドの例として，電源を設定する `SetPower(power)` を考える．このメソッドはいつでも実行することができるので，事前条件 Pre(`SetPower(power)`) はつねに `true` となる．事後条件 $Post$(`SetPower(power)`) としては，`Power = power` が設定できる．また，換気扇は，電源が on で，外気温が室温よりも低い場合に，室温 `Temp_in` に対して down という影響を持つ．これは，EW(`Temp_in.down`) = `[Power = ON` $\wedge$ `Temp_out` $<$ `Temp_in]` となる．

3.3 サービス

各サービスは，機器のメソッドを実行することによって機器・環境の値を更新し，その目的を達成する．サービスの記述は，具体的には，メソッドの実行，ローカル変数への値（メソッドの実行によって得られる機器や環境の状態）の代入，条件分岐やループといった制御構造から構成される．

複数のサービスが並行実行された場合，サービスは任意の順序で実行されていく．

具体的な HVAC サービスの記述例を**図 2** に示す（サービス記述の文法は，過去の文献 11) に従う）．このサービスでは室内の温度計 `Thermometer_in`，室外の温度計 `Thermometer_out`，エアコン `Air-conditioner` と換気扇 `Ventilator` が用いられる．

6 行目は，メソッド実行により室内の温度計の電源をオンにしている．8，9 行目では，メソッドを用いて室温，室外の温度をローカル変数へと代入している．これらのローカル変数の値は，12，14 行目のように分岐の際に用いられる．12 行目で，室温と設定温度の比較を行い，設定温度の方が低い場合，12 から 24 行目を繰り返し実行して，室温を下げる．

4. サービスの競合問題

ここでは，競合を機器に関する競合と環境に関する競合の 2 種類に分け，形式的に定義する．説明のため，6 章で検出した競合を例として示す．競合の具体的な例，および説明は 6 章で行う

4.1 機器に関する競合

ある機器 A において，あるサービスによって実行されたメソッド m の動作が，他のサー

```
1   SERVICE HVAC(tTemp user_temp) {
2    VAR
3     tTemp Ti_temp,To_temp; # Local variables
4    CONTENT
5     WHILE (true) {
6      Thermometer_in.SetPower(ON);
7      Thermometer_out.SetPower(ON);
8      Ti_temp := Thermometer_in.Measure();
9      To_temp := Thermometer_out.Measure();
10     Air-conditioner.SetPower(ON);
11     Air-conditioner.SetTemp(user_temp);
12     WHILE (Ti_temp > user_temp) {
13      Air-conditioner.SetMode(COOLING);
14      IF (Ti_temp > To_temp) {
15       WHILE (Ti_temp > To_temp && Ti_temp > user_temp) {
16        Ventilation.SetPower(ON) ;
17        Ti_temp := Thermometer_in.Measure();
18        To_temp :=Thermometer_out.Measure();
19       }
20       Ventilation.SetPower(OFF);
21      }
22      Ti_temp := Thermometer_in.Measure();
23      To_temp := Thermometer_out.Measure();
24     }
25     Air-Conditioner.SetMode(FAN);
26    }
27  }
```

図 2　HVAC サービス
Fig. 2　HVAC service.

ビスによって実行されたメソッドの動作により阻害されることを，機器に関する競合とする．ここで，動作が阻害される場合を，以下のように 3 つに分けて定義する．ただし，サービスが単独で実行されると，事前条件，事後条件は必ず満たされることを仮定する．

A1：他のメソッドが実行されることにより，メソッド m の実行終了時に，$Post(m)$ が成立しなくなる．

A2：他のメソッドが実行されることにより，メソッド m の実行開始時に，$Pre(m)$ が成立しなくなる．

A3：メソッド m の返り値が変数 f の値を用いて決定される場合に，他のメソッドの実行により，メソッド m によって読み込まれた f の値とメソッド m 終了時の f の値にずれが生じる．

[例 1] HVAC サービスと煙除去サービスを同時に実行すると競合 A1 が発生する．室温が設定温度以上で，かつ室外の温度が室温以上，室内に煙があるような場合を考える．このとき，煙除去サービスは室内の煙を除去するために換気扇を回す．その後，HVAC サービスは冷房をかけ，その際に暖かい外気の進入を防ぐため，換気扇を止める．このような順でサービスが実行された場合，室内に煙がある状態で換気扇が止められてしまう．

[例 2] 競合 A2 はたとえばホームシアタサービスと節電サービスの間に発生する．ホームシアタサービスが DVD 再生のために DVD プレイヤを操作しようとした際に，節電サービスにより DVD プレイヤの電源が切られたとする．すると，DVD プレイヤを操作できなくなる．

[例 3] ホームシアタサービスと節電サービス間で発生する競合 A3 について説明する．節電サービスによってテレビの電源が調べられている間に，ホームシアタサービスによってテレビの電源が ON にされたとする．結果として，節電サービスはテレビが OFF であると認識しているが，実際のテレビの電源は ON という状況になる．そのため，節電サービスが正しく動作しなくなる．

4.2　環境に関する競合

環境に関する競合は，複数の機器が 1 つの環境の要素に対し相容れない影響を有する状況に相当する．

ある環境の要素への影響 eff に対して，影響を有する機器があるか否かは，すべての機器の持つ影響の有無を規定する以下の述語で表現できる．

$$\mathcal{EW}(\mathit{eff}) = \bigvee_{A} A.EW(\mathit{eff})$$

ただし，$A.EW$ は機器 A の持つ環境の要素への影響の有無 EW を表す．環境の要素への影響 eff を与えている機器がシステム中に存在する場合，$\mathcal{EW}(\mathit{eff})$ が成り立つ．

環境の要素 E に関する競合は，以下のいずれかの条件が満たされる場合に発生する．

E1：ある環境の要素 E に対する異なった 2 つの影響 eff と eff' について，ある時点以降に $\mathcal{EW}(\mathit{eff}) \wedge \mathcal{EW}(\mathit{eff}')$ が成立し続ける．

E2：ある環境の要素 E に対して，ある機器のメソッド m によって環境の要素 E が読み込まれると同時に，$\mathcal{EW}(\mathit{eff})$ が成り立つ影響 eff が存在する．

競合 E1 はある環境の要素 E に対して異なった 2 つ以上の影響（たとえば，温度の上昇と下降等）が存在する場合に発生する．競合 E2 は，ある環境の要素に対し，要素への影響と読み込みが同時に発生した場合に起こる（これにより，読み込まれた値と実際の値の間にずれが生じてしまう）．

[例 4] 室温が設定温度以上でかつ外気が室内の温度よりも暖かく，室内に煙が存在するとする．このとき，HVAC サービスが実行されると，冷房がかけられる．その後煙除去サービスが実行されると，煙除去のために換気扇が回される．この換気扇を回した副作用として，室内に暖かい外気が入ってくる．このとき，室温に対して換気扇による室温を上昇させる影響とエアコンによる室温を下降させる影響が同時に発生するという競合 E1 が起こる．これは冷房の効率悪化につながる．

[例 5] 室温がエアコンの設定温度以上なので，HVAC サービスが冷房をかけたとする（このとき室温が HVAC によって読み込まれている）．同時に煙除去サービスによって換気扇が回されたと仮定し，換気扇が室温を設定温度より下げたとする（このとき換気扇による室温への影響がある）．結果として，HVAC が認識している室温と現実の室温にずれが生じ，室温が設定温度よりも低いにもかかわらず，冷房をかけることになる．これが HVAC サービスと煙除去サービス間で発生する室温に対する競合 E2 の例である．

例 5 の競合 E2 は，サービスが正常に実行されているのであれば，ただちに冷房が切られ問題がないように考えられる．しかし，何らかの要因により動作に正常時よりも時間がかかってしまう場合，たとえば，サーバが過負荷の場合を考慮すると，上記の状況が発生する可能性が考えられる．

5. モデル検査器 SPIN による競合検出

モデル検査器 SPIN を用いて検証を行うには，システムを Promela と呼ばれる言語で記述する必要がある．Promela では，システムは並行動作する複数のプロセスとして記述される．検証すべき性質は時相論理式の一種である LTL 式を用いて記述する．Promela 記述と，LTL 式を入力することにより，SPIN は自動的，網羅的にシステムが LTL 式で記述されている性質を満たしているかを検証する．もし，性質が満たされていない場合は，反例を得ることができる．

以下，5.1 節では，3 章で提案したホームネットワークシステムのモデルから，Promela 記述への変換法について述べる．我々は，この変換法を，bison と flex を用いてプログラムとして実装している．

次に，5.2 節では，4 章で与えた競合の定義に従い，競合が発生しないことを表す LTL 式を示すとともに，この LTL 式を用いて，どのように競合の検出を行うかについて説明する．

5.1 Promela によるシステムおよびサービスの記述

5.1.1 型 の 定 義

各変数の型の定義を行う．以下に型定義の例を示す．ここでは，温度を表す変数の型 `tTemp` と部屋の煙の有無を表す変数の型 `tSmoke` を定義している．型はすべて int 型とし，とりうる値を個別の整数とする．

```
#define tTemp int
#define tPower int /* ON or OFF */
#define OFF 0
#define ON 1
```

5.1.2 環境の定義

環境の要素の持つ変数はグローバル変数として定義する．以下に室温，室外の温度を表す変数を定義している．

```
tTemp temp_in; tTemp temp_out;
```

これらの変数の値は，機器の変数とは異なり，つねに変化する可能性がある．たとえば，室温はシステムからの影響とは関係なく変化する．そのため，各環境の要素の値は，機器がその値を読み込んだ際に任意の値をとるものとする[3)]．

5.1.3 機器の定義

環境の要素と同様に，機器の変数もグローバル変数として定義する．たとえば，エアコンの持つ変数 `power` と `temp`（設定温度），`mode` は以下のように定める．

```
tPower AC_power=OFF; tTemp AC_Temp=28;
tMode AC_Mode=FAN;
```

また，各メソッドは Promela のマクロとして定義する．すべてのメソッドは**図 3** に示す形で定義される．

`pre_condition` は事前条件を表し，2 行目で事前条件が成り立つ場合，3 行目以降に記述されたメソッドの本体が実行される．逆に成り立たない場合は，`false` 文により，このメソッドを呼んだサービスの実行が停止される．このことは，事前条件が成り立たない場合にメソッドの動作は不定とした 3 章のモデルと矛盾しているように見えるが，事前条件の違反は検証の過程で競合として検出されるため，問題とはならない．

メソッドが返り値を持つ場合，各サービスが持つローカル変数 `rvalue` に返り値を格納す

```
1  #define Appliance_method(argument){\
2   if::(pre_condition == true) ::else->false;fi;\
3   ... /* メソッドの動作記述 */
4   rvalue = return_value;}
```

図 3　機器のメソッドの記述
Fig. 3　Description of a method.

ることで，値をサービスへと返す．ここで，return_value はメソッドによって読み込まれた機器または環境の変数の値とする．

機器の環境の要素への影響の有無 EW は環境と機器の変数を用いて表現する．たとえば，エアコンの持つ室温への温度を低下させるという影響の有無は以下のように表現される．

```
(AC_power == ON && AC_Mode == COOLING)
```

機器ごとに定義されたこれらの論理式を用いることにより，システム全体からの影響の有無 $\mathcal{EW}(eff)$ を定義できる．たとえば，室温を下げる影響の有無は以下の式によって表現される．

```
#define temp_in_down
   ((AC_power == ON && AC_Mode == COOLING)
     ||(ventilator_power == ON
        && temp_in > temp_out))
```

5.1.4　サービスの定義

各サービスの動作は Promela の動作主体であるプロセスとして定義する．サービスの持つローカル変数はプロセスのローカル変数として定義できる．また，返り値を格納するための特殊なローカル変数として rvalue を用いる．サービスの動作は，メソッドの実行，ローカル変数への代入，条件分岐・ループのための制御文からなる．メソッドの実行はマクロの実行に変換する．代入，制御文は Promela 記述内のものと置き換えることにより記述する．

変換した HVAC サービスの一部を**図 4** に示す．これは図 2 の 13 行目までを Promela 記述へと変換したものである．3 行目はローカル変数の定義を変換し（図 2 の 2，3 行目），4 行目以降がサービスの動作記述を変換している．図 4 の 4 行目は，while 文を Promela での制御文である do 文へと変換したものである．図 4 の 5 から 10 行目は，図 2 の 6 から 11 行目のメソッドの実行に対応するマクロの呼び出しへと変換している．

```
1   proctype HVAC(tTemp user_temp){
2    int rvalue;
3    tTemp Ti_temp,To_temp; /* Local variables*/
4   do ::(end == 0)->
5    Thermometer_in_SetPower(ON);
6    Thermometer_out_SetPower(ON);
7    AC_SetPower(ON);
8    AC_SetTemp(user_temp);
9    Thermometer_in_Measure(); Ti_temp = r_value;
10   Thermometer_out_Measure(); To_temp = r_value;
11   do ::(Ti_temp > user_temp)->
12    AC_SetMode(COOLING);
13     :
14   }
```

図 4　HVAC サービスの Promela 記述
Fig. 4　HVAC service described in Promela.

5.2　検証する性質

SPIN を用いて競合の検出を行うため，各競合が発生しないことを表す LTL 式を与える．本論文では 2 種類の演算子 [] と<>を用いる．[] P は，P がシステムのすべての実行列でたえず成り立つ場合に真となる．一方<> P は，P がすべての実行列でいずれかが成り立つ場合に真となる．

5.2.1　機器に関する競合

機器に関する競合が発生していることを確認するため，新たな変数 appliance_error を機器ごとに定義する．競合が発生したときにこの変数の値を変更する．競合が発生していない場合の値を 0 とし，値が 1 で A1，値が 2 のときに A2，値が 3 のときに A3 が発生したことを表す．

図 5 は，図 3 に 9，4，10，7 行目を追加したものである．9，4，10 行目は，それぞれは競合 A1，A2，A3 を検出するために挿入されている．7 行目は，後述の競合 E2 の検出に用いる．

変数 appliance_error の値を確認することで競合の検出を行う．たとえば，競合 A1 が発生していないことを確認するために，以下の LTL 式を用いる．

```
LTL: ! <> (appliance_error == 1)
```

```
1  #define Appliance_method(argument){\
2   if::(pre_condition == true)
3     ::else-> \
4       appliance_error=2;\
5       false;fi;\
6   ...;
7   e_read = 1; e_read = 0;\
8   rvalue = return_value;\
9   if::(post_condition == true) ::else-> appliance_error=1;fi}\
10  if::(rvalue == return_value) ::else-> appliance_error=3;fi}\
11 }
```

図 5　競合検出のために変更したメソッド記述
Fig. 5　Description of a method for interaction detection.

5.2.2　環境に関する競合

E1 の検出：競合 E1 は，環境の要素への影響の有無を表す論理式を用いて検出する．具体的には以下の LTL 式を用いて検出する．

```
LTL:!<> [](e_eff1 && e_eff2)
```

`e_eff1` は 5.1.3 項で定義した，環境の要素 `e` に対して影響 `eff1` が発生していることを表す式 $\mathcal{EW}(eff_1)$ である（たとえば，`temp_in_down`）．`e_eff2` は同じ環境の要素 `e` に対して異なった影響 `eff2` が発生していることを表す．よって，この LTL 式は環境の要素 `e` に対し異なった影響 `eff1` と `eff2` が同時には発生しないことを示す．同じ環境の要素への影響が 3 種類以上存在する場合，すべての組合せについて調べる．たとえば，影響が 3 種類存在すると以下のようにする．

```
LTL:!<>( [](e_eff1 && e_eff2)
      || [](e_eff1 && e_eff3)
      || [](e_eff2 && e_eff3) )
```

E2 の検出：競合 E2 は，環境の要素の読み込みがあった際に，システムからの影響の有無を調べることで検出できる．環境の読み込みが行われたことを示すグローバル変数 `e_read` を新たに導入する．そして環境の読み込みを行うメソッドの記述に図 5 の 7 行目のように，この値を変化させる命令文を追加する．

図 5 の 7 行目では，一時的に `e_read` の値を 1 とすることで，環境の要素 `e` が読み込まれたことを表す．そして以下の LTL 式を用いることにより，競合 E2 を検出する．

```
LTL:!<> (e_read && (e_eff1 || .. || e_effn))
```

`e_read` は，いずれかのメソッドが環境の要素 `e` を読み込んだことを表す．そして `(e_eff1 || .. || e_effn)` は，環境要素に対するすべての影響の有無の論理和をとっている．したがって，この LTL 式は，環境要素 `e` を読み込んだ際に何の影響も発生していないことを示している．

6. 適 用 実 験

手法の有効性を示すため，図 1 で示したシステムと 2.2 章で示した 4 つのサービスに対して手法を適用した．実験には 900 MHz PentiumIII，512 MB メモリを有する WindowsXP PC を用いた．

システムの環境要素は室温，室外の温度，煙の有無，明るさのレベルの 4 つ，機器は図 1 で示されている 10 個とした．機器はそれぞれ 1 から 3 個の変数，2 から 6 個のメソッドを持つものとして記述した．煙除去サービス，ホームシアタサービスは，図 2 と同程度の行数で記述できた．今回，節電サービスは，テレビの電源を確認し，DVD の電源を切るだけのサービスとしたので，10 行程度となった．

作成したシステム記述を提案法を用いて変換して Promela 記述を得た．そして，2 つのサービスを並行実行した際に発生する競合を調べた．HVAC サービスは実行時にエアコンの設定温度を決める必要があるが，今回の実験では，28°C とした．機器に関する競合の検出は，システムに含まれるすべての機器について，競合 A1，A2，A3 を検証した．つまり，機器の数が 10 であるので，サービスの組ごとに 30 回の検証を行った．環境に関する競合に関しても，環境の要素ごとに競合 E1，E2 を調べた．4 つの環境要素を想定しているため，検証はサービスの組ごとに 8 回行った．また，検証を行う際には，SPIN で利用可能な Partial order reduction と呼ばれる手法を用いて状態数の削減を行った．

検証は「競合が発生しない」という性質を確認することで行う．そのため，検証結果は `true`（競合が発生しない）または，`false`（競合が発生する）となる．`false` の場合は，反例（競合が発生するまでの実行列）が得られる．

6.1　実 験 結 果

検出できた競合を**表 2** にまとめる．表中の A1，A2，A3，E1，E2 は，サービスの組に対して発生した競合の種類を示している．() 内は，その競合が発生した機器または環境を示す．たとえば，HVAC サービスと煙除去サービス間では，換気扇において競合 A1 が発生した．また，室温に対して競合 E1 と競合 E2 が，煙に対して競合 E2 が発生した．

表 2　各サービスの組で検出された競合の種類
Table 2　Feature interactions detected between two services.

	節電サービス	ホームシアタサービス	煙除去サービス
HVAC サービス	E2(室温)	E2(室温) E2(明るさ)	A1(換気扇) E1(室温) E2(室温) E2(煙)
煙除去サービス	E2(煙)	E2(煙) E2(明るさ)	
ホームシアタサービス	A1(DVD) A2(DVD) A3(テレビ) E2(明るさ)		

ここで，HVAC サービスと煙除去サービスに対して行った検証の詳細を**表 3** に示す．(A) は機器に関する競合，(B) は環境に関する競合の検証結果である．実行時間，状態数はそれぞれ検証にかかった時間とその際に探索した状態数を表す．どちらのサービスでも利用されていない機器に関する競合の検証の結果は省略した．実験結果より，実行時間はどれも現実的な時間に収まっていることが分かる．

また，反例を用いることにより，それぞれの競合が発生した状況を調べることができた．以下にその一部について説明する．

HVAC サービスと煙除去サービスの間の競合 A1（換気扇）：煙除去サービス中の換気扇の電源をオンにするメソッド `SetPower(ON)` の事後条件 `Power=ON` がメソッド実行終了時に成り立たない．

反例を調べることにより，以下の状況が分かった．まず，室内に煙があるとする．さらに，HVAC サービスが室温を下げるために冷房をかけていると仮定する．この状況下で，煙除去システムが実行されると煙を除去するために換気扇が回る．その後，室温が冷やされ，室外の温度以下になり，HVAC サービスが暖かい空気の進入を防ぐために換気扇を止める．その結果，室内に煙が残っているにもかかわらず，換気扇が止められてしまう．

HVAC サービスと煙除去サービスの間の競合 E1（室温）：反例により，室温に温度上昇の影響が発生していることを示す `temp_in_up` と温度下降の影響が発生してることを示す `temp_in_down` が同時に真となることが分かった．

具体的には，この競合は以下のような状況で発生していた．室温が室外より低く，エアコンによって冷房がかけられており（`AC_Power` == `ON` かつ `AC_Mode` == `COOLING`）室内に煙があると仮定する．この状況下では，煙除去サービスが換気扇を回す．ここで，室温が室外の温度より低い（`Temp_in` < `Temp_out`）ため，換気扇は室温を上昇させる働きをしてしまう．その結果，エアコンによる室温を下げる影響と，換気扇による室温を上げる影響が，同時に発生する．

HVAC サービスと煙除去サービスの間の競合 E2（室温）：まず，HVAC サービスが温度計のメソッド `Measure()` を実行する．このメソッドにより，環境の要素の室温が読み込まれる．同時に，煙除去サービスが換気扇を回すメソッド `SetPower(ON)` を実行したとする．

表 3　HVAC サービスと煙除去サービスについての検証結果
Table 3　Verification results between HVAC service and Air-cleaning service.

(A) 機器に関する競合
(A) Appliance interactions

機器	競合	検証結果	実行時間 (秒)	状態数
エアコン	A1	true	1.31×10^{0}	9.4×10^{4}
	A2	true	1.17×10^{0}	9.4×10^{4}
	A3	true	1.23×10^{0}	9.4×10^{4}
温度計（室内）	A1	true	1.23×10^{0}	9.4×10^{4}
	A2	true	1.23×10^{0}	9.4×10^{4}
	A3	true	1.22×10^{0}	9.4×10^{4}
温度計（室外）	A1	true	1.20×10^{0}	9.4×10^{4}
	A2	true	1.16×10^{0}	9.4×10^{4}
	A3	true	1.23×10^{0}	9.4×10^{4}
煙センサ	A1	true	1.17×10^{0}	9.4×10^{4}
	A2	true	1.19×10^{0}	9.4×10^{4}
	A3	true	1.16×10^{0}	9.4×10^{4}
換気扇	A1	false	7.81×10^{-1}	2.5×10^{4}
	A2	true	1.84×10^{0}	7.3×10^{4}
	A3	true	1.92×10^{0}	7.3×10^{4}

(B) 環境に関する競合
(B) Environment interactions

環境要素	競合	検証結果	時間 (秒)	状態数
室温	E1	false	1.00×10^{0}	2.3×10^{4}
	E2	false	4.03×10^{-1}	2.4×10^{1}
室外の温度	E1	true	1.17×10^{0}	9.4×10^{4}
	E2	true	1.14×10^{-1}	9.4×10^{4}
煙	E1	true	1.20×10^{0}	9.4×10^{4}
	E2	false	4.06×10^{-1}	1.1×10^{1}
明るさ	E1	true	1.19×10^{0}	9.4×10^{4}
	E2	true	1.14×10^{-1}	9.4×10^{4}

すると，換気扇によって室温を下げる影響が発生する．そのため，HVAC サービスが返り値として得た室温と，実際の室温の値にずれが生じてしまう．ここで，HVAC サービスが得た値が設定温度以上であり，実際の室温が設定温度以下だとする．すると，設定温度よりも室温が低いにもかかわらず，冷房がかけられてしまう．

ホームシアタサービスと節電サービスの間の競合 A2（DVD）：ホームシアタサービスが DVD を再生するためにメソッド `SetMode(PLAY)` を実行したときに，その事前条件が成り立たたない．これは，節電サービスによって，DVD の電源が消されることでこの競合が発生する．結果として，メソッド `SetMode(PLAY)` が正しく実行されなくなる．

ホームシアタサービスと節電サービスの間の競合 A3（テレビ）：この競合は以下のような状況で発生した．まず，テレビの電源が OFF のときに，節電サービスのメソッド `CheckPower()` がテレビの電源を確認し，返り値を `OFF` と決めた．その間に，ホームシアタサービスがテレビのメソッド `SetPower(ON)` を実行して，実際の電源の状態と節電サービスが得た状態の間にずれが生じる．

6.2 考　　察

ホームネットワークシステムの競合問題において，環境をどのように扱うかが非常に重要である．既存の環境を扱った競合の検出法（たとえば文献 13)，15)）では，サービスが機器を操作した際，環境要素に対して影響があるかないかを定義することで環境を扱っている．そして，複数のサービス内に同一の環境要素に対する影響が含まれるかどうかを確認することにより競合の有無を確認している．しかし，この方法だと実際には問題とならないような状況も競合と判断する可能性がある．

提案手法では，このような問題を扱うため，影響の方向性を考え，それらが異なった場合のみ，競合と扱うことにより，この問題の解決を図った．

たとえば，煙除去サービスは換気扇を操作し，煙を除去しようとする．換気扇は煙の有無という環境の要素に対して影響を持つ．一方，換気扇の操作は HVAC サービス内にも含まれる．これら 2 つのサービスが煙の有無という環境の要素に対して影響を持つ換気扇を操作する．これは既存の競合検出手法では，競合として扱われるしまう．しかし，実際にはサービスの動作が阻害されることはなく問題とならない．

適用実験において，提案手法では HVAC サービスと煙除去サービスの間で，煙の有無に対して環境に関する競合 E1 は発生しないという結果が得られた．これは，煙の有無への影響に，煙を除去する，煙を発生させるという種類を導入したため，煙を除去するという同一の影響であるために競合ではないと判断できたためである．環境への影響の種類を考慮することにより，より正確に競合の判断ができる．

また，HVAC サービスと煙除去サービス間で室温において発生した環境に関する競合についても，室温に対して温度を上昇させる影響と下降させる影響が発生しているときのみを競合としているため，競合が発生する状況を正確にとらえることができた．

本論文で提案した手法は，サービスの動作にのみ注目して競合の検出を行っている．そのため，システムがどのようなネットワークプロトコルを用いて実装されているかにかかわらず適用可能である．ここでは例として ECHONET 規格の機器およびサービスへの提案手法の適用法について簡単に説明する．

ECHONET では，機器は状態変数からなるオブジェクトとして表現される．機器に対する操作は，個々の状態変数の値を変更・参照することで行われる．そこで，ECHONET 規格の機器の持つ状態変数を提案モデルの機器の持つ変数として定義できる．また，ECHONET 規格内の各状態変数への更新・参照を，提案モデル内のメソッドとして記述することができる．環境に対する影響については ECHONET 規格では定められていないため，各機器がどのような機器の状態と環境状況において，環境に対してどのような影響を与えるかを検証時に付与する必要がある．つまり提案法を適用するために必要となるのは，ECHONET 規格から得られる機器の状態変数をもとに，変数とメソッドの定義を行うことと，定義した機器に対して環境への影響を付与すること，さらにサービスの記述を提案したモデルの記述に変換することである．

一般公開されている ECHONET の規格書[6] 内で記述されている空調用の換気扇と，家庭の消費電力を抑えるためのサービスを，本論文で提案したモデルで記述したものを付録に示す．機器のモデルは ECHONET 規格の機器で定義されている状態変数をモデルの変数とメソッドとすることで容易に得ることができた．また，サービスについてもサービスが行う機器の操作部分のみに注目し，機器に対する操作の列として記述できた．

7. 関 連 研 究

電話通信システムの分野において，競合問題は広く扱われてきた[9]．たとえば，文献 2) では，モデル検査器 SPIN を用いた競合の検証法が示されている．ホームネットワークシステムでは，機器を特定の状態にするだけでなく，システムを取り巻く環境をユーザが望む状態へと変化させることを目的としている．そのためシステムの周辺環境を扱うために，環境をモデル内に取り入れる必要がある．しかし，電話通信システムで用いられている手法では，システムを機器の状態のみで表現し，システムを取り巻く環境を表現することができな

い．そのため，システムが環境に対して矛盾する影響を持っているかどうかの判断ができなかった．そこで，本論文ではシステムを取り巻く環境をシステムを構成する要素の一部としてとらえ，モデル化した．そして，環境に対して機器がどのような影響を持つかを規定することにより，環境に関する競合の検出が可能となった．

システム運用の際に ECA ルールと呼ばれるルールで記述されたポリシをもとに，システムの動作を制御することで競合を解決する手法が文献 4)，20) で提案されている．ECA ルールは，システムの状態と条件，アクションの組で表現される．これは，システムがある状態で何らかのイベント（条件）が発生した際に実行されるアクションを示している．同一イベントで衝突する複数のアクションが同時に実行される場合は，事前に決められた対応策の下で競合を解消してシステムが動作していく．また，静的な方法で競合を検出する手法が文献 18) で，対応策としてルールの実行順序を設定する方法が文献 19) でそれぞれ提案されている．

ECA ルール用いた手法をホームネットワークシステムに対して適用するためには，システムの周辺環境への影響をどのように扱うかが問題となる．環境の状態はシステムからの影響により徐々に変化するものであり，アクションの実行にともなって環境の状態が決定的に変化するとは限らない．そのため，実行されるアクションに注目して競合の検出を行う ECA ルールを用いる手法では，環境に関する競合の検出が困難である．本研究では環境への影響を機器の状態から定義し，矛盾した影響が発生しないかをモデル検査を用いることでシステムの全状態について調べることで環境に関する競合の検出を実現した．

環境と関係した競合問題を扱った研究として，インテリジェントビルを対象とした文献 13) があげられる．しかし，環境への影響をその有無のみで扱っているため，環境に対して複数の影響が存在した場合に，それらがどういった方向性を持つ影響なのかが区別できない．そのため，6.2 章で述べた誤検出の問題が発生する．本研究では環境自体の状態を表す変数とは別に，環境への影響の方向性を定義し，異なった方向性の影響が発生していないかを調べることにより競合の検出を行っている．競合問題に関する研究に限らず，一般的な環境のモデルである 4 変数モデル[17] においても，影響の方向性は取り扱われていない．そのため，これらのモデルを適用しようとすると，環境に関する競合の誤検出が起こってしまう．

ホームネットワークシステムのサービス競合については，文献 10)，15) で研究がなされている．文献 10) では，サービスの実行時に競合が発生しないかを調べ，競合が発生する場合は，事前に設定された優先度に従って，その実行を制御している．しかし，優先度設定のために事前にどのサービス間で競合が発生するかが特定されている必要がある．提案手法によってサービス実行に先立った競合の特定が実現されるので，提案手法を前処理に利用することで，文献 10) のオンラインでの解決法をより有効に利用することが可能となる．

文献 15) では，静的解析を用いた検出法が提案されている．しかしながら，静的解析手法では，実際にサービスを動作させた場合には，起こりえないような多数の競合を検出してしまう可能性が高い．たとえば，暖冷房が可能な 2 つのエアコンがあり，それぞれを異なったサービスが制御しているとする．それらのエアコンを同一の設定温度で操作し，かつ暖房の設定温度が冷房の設定温度より低ければ，暖房と冷房が同時にかけられることはない．しかし，文献 15) 内の静的解析では，それぞれのサービス記述内に冷房をかける，暖房をかけるという記述が含まれるため，競合が発生すると判断してしまう．提案手法では，モデル検査を用いて，実際に実行可能な状態のみを探索することにより，こういった実際に起こりえない競合の誤検出を防ぐことができる．

我々は，本研究の予備研究である文献 11) において，文献 15) のモデルをもとに，モデル検査を用いた検出法を提案している．この研究では，モデルとは別に検証すべき特性が与えられることを仮定している．また文献 15) と同様，環境への影響をその有無のみで扱っていた．本研究では，モデルの改善を行うとともに，モデルの定義のみに基づいた形式的な競合の定義を与えた．また，文献 11) では，モデル検査器として SMV[12] を用いていたが，SMV では状態遷移を直接記述することでシステムの動作を指定しなければならず，メソッド呼び出しのような処理の記述が困難であった．本研究ではモデル検査器として SPIN を用いることで，この問題を解決した．

8. おわりに

本論文では，ホームネットワークシステムにおけるサービス間の競合を検出する方法を提案した．具体的には，ホームネットワークシステムと競合の表現に適したモデルを提案した．そして，そのモデル上で競合を形式的に定義した．さらに，提案したモデルの記述をモデル検査器 SPIN の入力へと変換する手法を考案し，それを実現するプログラムを実装した．また，各競合が検出できる性質を明らかにした．これにより，提案モデルを用いたシステムの記述から，自動的に競合の検出が可能となった．

さらに，提案法の有効性を示すために，適用実験としてサービスの例に対して競合の検出を行った．競合検出の結果，サービスの組ごとに 1 から 4 つの競合を検出できた．また，反例を調べることにより，各競合が発生した状況を確認した．その結果，環境への影響の種類という，ホームネットワークシステムにおける競合問題を扱ううえで重要な要素が扱えてい

ることが確認できた．

また，一般公開されている ECHONET の規格書内で記述されている空調用の換気扇と，家庭の消費電力を抑えるためのサービスを，本論文で提案したモデルを用いて記述できることを示した．しかしながら，ホームネットワークのすべてのユーザがこのような記述法に関する知識を持っているわけではない．そこで，システムを構築している機器とサービスの情報をもとに，システム記述を得るための支援ツールの開発が今後の課題にあげられる．

謝辞　本研究の一部は，文部科学省グローバル COE プログラム「アンビエント情報社会基盤創成拠点」の研究助成によるものである．ここに記して謝意を表す．

参 考 文 献

1) Bensalem, S., Ganesh, V., Lakhnech, Y., Noz, C.M., Owre, S., Rueß, H., Rushby, J., Rusu, V., Saïdi, H., Shankar, N., Singerman, E. and Tiwari, A.: An Overview of SAL, *LFM 2000: 5th NASA Langley Formal Methods Workshop*, Holloway, C.M., (Ed.), pp.187–196, NASA Langley Research Center, Hampton, VA (2000).
2) Calder, M. and Miller, A.: Feature interaction detection by pairwise analysis of LTL properties – A case study, *Formal Methods in System Design*, Vol.28, No.3, pp.213–261 (2006).
3) Chan, W., Anderson, R.J., Beame, P., Burns, S., Modugno, F., Notkin, D. and Reese, J.D.: Model Checking Large Software Specifications, *IEEE Trans. Softw. Eng.*, Vol.24, No.7 (1998).
4) Chomicki, J., Lobo, J. and Naqvi, S.: Conflict Resolution Using Logic Programming, *IEEE Trans. Knowledge and Data Engineering*, Vol.15, No.1, pp.244–249 (2003).
5) Clarke, E.M., Grumberg, O. and Peled, D.A.: *Model Checking*, MIT Press (1999).
6) ECHONET Consortium. http://www.echonet.gr.jp/
7) *Feature Interaction in Telecommunications and Software Systems*, Vol.I-VIII, IOS Press (1992–2005).
8) Holzmann, G.J.: The Model Checker SPIN, *IEEE Trans. Softw. Eng.*, Vol.23, No.5, pp.279–295 (1997).
9) Keck, D.O. and Kuehn, P.J.: The feature and service interaction problem in telecommunicationssystems: A survey, *IEEE Trans. Softw. Eng.*, Vol.24, No.10, pp.779–796 (1998).
10) Kolberg, M., Magill, E.H. and Wilson, M.: Compatibility issues between services supporting networked appliances, *IEEE Communications Magazine*, Vol.41, No.11, pp.136–147 (2003).
11) Leelaprute, P., Nakamura, M., Tsuchiya, T., Matsumoto, K. and Kikuno, T.: Describing and Verifying Integrated Services of Home Network Systems, *The 10th Asia-Pacific Software Engineering Conference* (*APSEC2005*), pp.549–558 (2005).
12) McMillan, K.L.: *Symbolic Model Checking*, Kluwer Academic Publishers (1993).
13) Metzger, A. and Webel, C.: Feature interaction detection in building control systems by means of a formal product model, *Feature Interactions in Telecommunications and Software Systems VII*, pp.105–122 (2003).
14) Moyer, S., Marples, D. and Tsang, S.: A protocol for wide area, secure networked appliances communication, *IEEE Communications Magazine*, Vol.38, No.10, pp.52–59 (2001).
15) Nakamura, M., Igaki, H. and Matsumoto, K.: Feature Interactions in Integrated Services of Networked Home Appliance, *Proc. Int'l. Conf. on Feature Interactions in Telecommunication Networks and Distributed Systems* (*ICFI'05*), pp.236–251 (2005).
16) OSGi Appliance: The OSGi Service Platform. http://osgi.org
17) Parnas, D.L. and Madey, J.: Functional documents for computer systems, *Science of Computer Programming*, Vol.25, No.1, pp.41–61 (1995).
18) Shankar, C. and Campbell, R.: A Policy-based Management Framework for Pervasive Systems using Axiomatized Rule-Actions, *4th IEEE International Symposium on Network Computing and Applications* (*IEEE NCA05*), pp.255–258 (2005).
19) Shankar, C. and Campbell, R.: Ordering management actions in pervasive systems using specification-enhanced policies, *4th Annual IEEE International Conference on Pervasive Computing and Communications* (*PERCOM'06*), pp.234–238 (2006).
20) Srivastava, B. and Kambhampati, S.: The Case for Automated Planning in Autonomic Computing, *2nd International Conference on Autonomic Computing, 2005* (*ICAC 2005*), pp.331–332 (2005).
21) UPnP Forum. http://www.upnp.org/

付　　録

A.1　ECHONET 規格エアコンの記述

以下に ECHONET 規格書内の空調用換気扇を，提案したモデルに基づいて記述したものを示す．ECHONET 規格書の空調用換気扇は，動作状態，室内相対湿度設定値，換気自動設定，室内相対湿度計測値，換気風量設定値，熱交換器動作設定，CO_2 濃度計測値，煙検知状態の 8 つの状態変数を持つ．

動作状態，室内相対湿度設定値，換気自動設定，換気風量設定値，熱交換器動作設定，CO_2 濃度計測値の 6 つについてはその状態を調べる Get と値を設定する Set という操作が可能

であり，室内相対湿度計測値と煙検知状態についてはその値を調べる Get という操作のみが可能である．事前条件，事後条件については，そのような定義が存在しないため，以下のように設定した．事前条件については，すべての操作について true とした．事後条件については，機器の状態変数を変更する Set に関しては状態変数の値が設定値へと変更されていることとし，Get については true とした．事前条件・事後条件は，変数名以外は同一となるため，動作状態 OperationStatus についての Set と Get のみ記述している．

環境への影響については，規格で定義されていないため新たに定義する必要がある．ここでは，以下の 3 つの影響を定義した．換気扇の動作状態が ON のときに煙を除去する．換気扇の動作状態が ON で室温より外気が暖かいときに，室温を上昇させる．換気扇の動作状態が ON で室温より外気が冷たいときに，室温を下降させる．

- Ventilation = (F_V, M_V, EW_V)
- F_V = (OperationStatus, RoomRelativeHumid, VentilatingStatus, MeasuredHumid, VentilatingWindLevel, HeatExchangerStatus, CO2Concentration, SmokeDetectionStatus)
- M_V = (Get_OperationStatus(), Set_OperationStatus(status), Get_RoomRelativeHumid(), Set_RoomRelativeHumid(humid), Get_VentilatingStatus(), Set_VentilatingStatus(status), Get_MeasuredHumid(), Get_VentilatingWindLevel(), Set_VentilatingWindLevel(level), Get_HeatExchangerStatus(), Set_HeatExchangerStatus(status), Get_CO2Concentration(), Set_CO2Concentration(concentration), Get_SmokeDetectionStatus(status))
  - Pre(Get_OperationStatus()) = [true]
  - Post(Get_OperationStatus()) = [true]
  - Pre(Set_OperationStatus(status)) = [true]
  - Post(Set_OperationStatus(status)) = [OperationStatus =status]
- EW_V(eff):
  eff=Smoke_removal の場合： [OperationStatus = ON]
  eff=Temp_in_up の場合： [OperationStatus = ON ∧ Temp_in<Temp_out]
  eff=Temp_in_down の場合： [OperationStatus = ON ∧ Temp_in > Temp_ out]
  上記以外の場合： [false]

このモデルを変換し，Promela 記述を得ることができる．以下にその一部である状態変数の定義と，動作状態 OperationStatus に対する値の更新 Set_OperationStatus と参照 Get_OperationStatus の記述を示す．他の状態変数に対する操作は状態変数名のみ異なる．

```
/* 型の定義 */
#define tStatus int
#define OFF 0
#define ON 1
#define tAuto
#define Auto 2
#define NonAuto 3
#define tSmoke int
#define Found 4
#Define NotFound 5

/* 変数の定義 */
/* 上から順に動作状態，室内相対湿度設定値，換気自動設定，室内相対湿度計測値，換気風量設定値，
熱交換器動作設定，CO2 濃度計測値，煙検知状態 */
tStatus Ventilation_OperationStatus;
int Ventilation_RoomRelativeHumid;
tAuto Ventilation_VentilatingStatus;
int Ventilation_MeasuredHumid;
int Ventilation_VentilatingWindLevel;
tStatus Ventilation_HeatExchangerStatus;
int Ventilation_CO2Concentration;
tSmoke Ventilation_SmokeDetectionStatus;

/* 競合検出用の変数 */
int Ventilation_error=0;

/* 動作状態に対する操作 */
#define Ventilation_Get_OperationStatus(){\
 if::(true) ::else-> \
      Ventilation_error=2; false;fi;\
 rvalue = Ventilation_OperationStatus;\
 if::(true) ::else-> ventilation_error=1;fi}\
 if::(rvalue == Ventilation_OperationStatus)\
   ::else-> Ventilation_error=3;fi}\
}
#define Ventilation_Set_OperationStatus\
(status){\
 if::(true) ::else-> \
```

```
    Ventilation_error=2; false;fi;\
 Ventilation_OperationStatus = status;\
 if::(Ventilation_OperationStatus == status)\
   ::else-> Ventilation_error=1;fi}\
}
```

A.2　ECHONET 規格の消費電力制御サービス（EMS）

ここでは，ECHONET によって実現できるサービスとして ECHONET 規格書内であげられている EMS（Energy Management Setvice）を，提案したモデルで記述したものを示す．このサービスは全機器の合計消費電流値を調べ，その値が制御開始電流値（Is）以上になった際に，各機器に与えられた優先度に従って，登録された機器の利用を中断することで，消費電流値を Is 以下に減少させる．一方，合計消費電流値が制御終了電流値（Ie）以下であり，中断されている機器が存在する場合，その機器を復帰させるサービスである．具体的には，まず，ユーザが制御開始電流値，ならびに制御終了電流値を設定する．そして以下の動作を繰り返し実行しサービスを実現する．

(1)　対象となる機器すべての電流値を測定する．値が Is 以上である場合 (2) へ．Ie 以下である場合は (3) へ．どちらにも一致しない場合は (1) へ．

(2)　優先度の一番低い機器を中断させ，(1) へ戻る．中断させることのできる機器が存在しない場合アラームを鳴らしサービスを終了する．

(3)　中断されている機器がある場合復帰させる．存在しない場合何も行わない．その後 (1) へ戻る．

ECHONET の規格書[6] をもとに，エアコン，換気扇，冷蔵庫，電子レンジ，電気ストーブ，洗濯機により構成されているホームネットワークを考える．この場合 EMS を提案モデルで記述すると以下のようになる．ただし，各機器の優先度および消費電流値はすでに与えられていると仮定している．またすべての機器がサービスに登録されているとする．提案した変換法を用い，この記述から SPIN で検証可能な 100 行程度の Promela プログラムが得られた．

```
int priority[6]; /* 各機器の優先度，エアコン，換気扇，
    冷蔵庫，電子レンジ，電気ストーブ，洗濯機の順に格納*/
int consumption[6];/* 消費電流値 優先度と同一の順で格納*/

SERVICE EMS(int Is, int Ie){
 VAR
  int It;     # 現在の消費電流値
  bool cut[6]; # 各機器が中断されているかを表す
               # 1 で中断中
  int maxpriority, minpriority;
    #max:中断されている機器の持つ優先度の中で最大の優先度
    #min:中断されていない機器の持つ優先度の中で最小
  tStatus Status_tmp;

CONTENT
 # cut の初期化
 cut[0]=0;cut[1]=0;cut[2]=0;
 cut[3]=0;cut[4]=0;cut[5]=0;

 while(true){
  #全ての機器の総消費電流値の測定
  #maxpriority, minpriority の測定
  It = 0; maxpriority = 0; miniproority = 255;
#エアコンの電流値の測定
  Status_tmp :=
    Air_conditioner.Get_OperationStatus();
  if(Status_tmp == ON){
   It := It + consumption[0];
   if(cut[0] = 0 && priority[0] < minpriority){
     minpriority := priority[0];}}
  if(cut[0] = 1 && priority[0] > maxpriority){
   maxpriority := priority[0];}
#換気扇の電流値の測定
 Status_tmp := Ventilater.Get_OperationStatus();
 if(Status_tmp = ON){
   It := It + consumption[1];
   if(cut[1] = 0 && priority[1] < minpriority){
     minpriority := priority[1];}}
 if(cut[1] = 1 && priority[1] > maxpriority){
  maxpriority := priority[1];}
#冷蔵庫の電流値の測定
 Status_tmp := Freezer.Get_OperationStatus();
 if(Status_tmp = ON){
  It := It + consumption[2];
  if(cut[2] = 0 && priority[2] < minpriority){
    minpriority := priority[2];}}
 if(cut[2] = 1 && priority[2] > maxpriority){
```

```
  maxpriority = priority[2];}
#電子レンジの電流値の測定
 Status_tmp := Microwave.Get_OperationStatus();
 if(Status_tmp = ON){
  It := It + consumption[3];
  if(cut[3] = 0 && priority[3] < minpriority){
    minpriority := priority[3]; }}
 if(cut[3] = 1 && priority[3] > maxpriority){
  maxpriority = priority[3];}
#電気ストーブの電流値の測定
 Status_tmp := Heater.Get_OperationStatus();
 if(Status_tmp = ON){
  It := It + consumption[4];
  if(cut[4] = 0 && priority[4] < minpriority){
    minpriority := priority[4];}}
 if(cut[4] = 1 && priority[4] > maxpriority){
  maxpriority := priority[4];}
#洗濯機の電流値の測定
 Status_tmp := Washer.Get_OperationStatus();
 if(Status_tmp = ON){
  It := It + consumption[5];
  if(cut[5] = 0 && priority[5] < minpriority){
     minpriority := priority[5];}}
 if(cut[5] = 1 && priority[5] > maxpriority){
  maxpriority := priority[5];}

 if(It>Is){/* 機器の中断 */
  if(minpriority = priority[0]){
   Air_conditioner.Set_OperationStatus(OFF);
   cut[0] := 1;}
  else if(minpriority = priority[1]){
   Ventilater.Set_OperationStatus(OFF);
   cut[1] := 1;}
  else if(minpriority = priority[2]){
   Freezer.Set_OperationStatus(OFF);
   cut[2] := 1;}
  else if(minpriority = priority[3]){
   Microwave.Set_OperationStatus(OFF);
   cut[3] := 1;}
  else if(minpriority = priority[4]){
   Heater.Set_OperationStatus(OFF);
   cut[4] := 1;}
  else if(minpriority = priority[5]){
   Washer.Set_OperationStatus(OFF);
   cut[5] := 1;}
  else {Alarm.Set_OpertionStatus(ON);}}
 ::(It<Ie) -> /* 機器の復帰 */
  if(maxpriority = priority[0]){
   Air_conditioner.Set_OperationStatus(ON);
   cut[0] := 0;}
  else if(maxpriority = priority[1]){
   Ventilater.Set_OperationStatus(ON);
   cut[1] := 0;}
  else if(maxpriority = priority[2]){
   Freezer.Set_OperationStatus(ON);
   cut[2] := 0;}
  else if(maxpriority = priority[3]){
   Microwave.Set_OperationStatus(ON);
   cut[3] := 0;}
  else if(maxpriority = priority[4]){
   Heater.Set_OperationStatus(ON);
   cut[4] := 0;}
  else if(maxpriority = priority[5]){
   Washer.Set_OperationStatus(ON);
   cut[5] := 0;}
 }
}
```

(平成 19 年 10 月 5 日受付)
(平成 20 年 3 月 4 日採録)

松尾　尚文

平成 18 年 3 月大阪大学大学院情報科学研究科博士前期課程修了．修士（情報科学）．平成 18 年 4 月同大学大学院情報科学研究科博士後期課程入学，現在に至る．IEEE，電子情報通信学会各会員．

パッタラ　リーラープルット

平成 13 年大阪大学基礎工学部情報科学科卒業．平成 15 年同大学大学院修士課程了．平成 18 年同大学院博士後期課程修了．平成 18 年カセットサート大学（タイ）工学部情報科学科助手となり，現在に至る．博士（工学）．通信サービスとサービス競合，ホームネットワークシステムのサービス等の研究に従事．IEEE，電子情報通信学会各会員．

土屋　達弘

平成 7 年大阪大学大学院基礎工学研究科博士前期課程修了．博士（工学）．現在，大阪大学大学院情報科学研究科准教授．IEEE，ACM，電子情報通信学会各会員．

菊野　　亨（フェロー）

昭和 50 年大阪大学大学院基礎工学研究科博士後期課程修了．工学博士．広島大学工学部助教授，大阪大学基礎工学部教授を経て，現在，大阪大学大学院情報科学研究科教授．情報処理学会フェロー．電子情報通信学会フェロー．IEEE，ACM，電子情報通信学会，日本信頼性学会各会員．