Vol.2020-DPS-182 No.49
Vol.2020-CSEC-88 No.49
2020/3/13

Web of Thingsを利用した Trigger-Action Programming環境における 不適切なルール実行の防止手法の検討

村上 恒[1,a)]　掛井 将平[1]　齋藤 彰一[1]

概要：近年，様々なIoT機器が家庭を含め多くの場所で普及している中，機器同士の相互運用性を高めるため，IoT機器向けのオープンな規格・アーキテクチャであるWeb of Things (WoT) と，それを利用したIoTゲートウェイであるWebThings Gatewayが提案されている．WebThings Gatewayでは，機器の制御のため，事前に利用者が定義したルールに基づき，機器・サービスAの状態に応じて機器・サービスBに特定の動作を与えるTrigger-Action Programming (TAP) と呼ばれる手法を用いている．しかし，TAPでは制御ルールを人間が設定する必要があるが，こうしたルールを本来意図した通りに条件の漏れなく設定するのは人間にとって困難を伴う．また，設定ミスによる機器の不適切な挙動が問題を引き起こす可能性がある．そこで本研究では，Web of ThingsとWebThings Gatewayの両方を拡張し，設定ミスが疑われる挙動を自動的に判別，中断させる手法を検討する．

キーワード：Web of Things，Trigger-Action Programming, 不適切なルールの実行防止

1. はじめに

近年，様々なIoT機器が普及してきている．スマートフォンのような従来から存在する通信端末から産業用システムまで，多彩な機器がインターネットに接続するようになってきた [1]．また，International Data Corporation (IDC) の調査によれば，スマートホーム市場も今後大きく伸びていくと予想されており [2]，今後数年で日常生活の中により多くのIoT機器が溶け込んでいくと考えられる．

しかし，IoT機器の数が増えていく一方で，IoT機器同士の相互運用性は高まっておらず，特定のプラットフォームでしか動作しない『サイロ化』が問題となっている．サイロ化がある状態では，利用者は機器がどのプラットフォームに対応していてどの機器と連携可能であるかを把握せねばならず，負担が大きい．また，機器のベンダーによるサポートが終了した際，機器がプラットフォームごと利用できなくなる懸念もある．したがって，サイロ化は利用者にとって不利益である．そこで，W3Cを中心に，IoT機器向けのオープンな規格・アーキテクチャであるWeb of Things (WoT) の策定が進められている [3]．また，WoTを利用したIoTゲートウェイの実装として，MozillaがWebThings Gatewayをリリースしている [4]．こちらは自由ソフトウェアとしてオープンな形で提供されており，Raspberry Piのような安価なコンピューター上でも動作させることができる．これらのことから，WoTを中心としたエコシステムは今後さらに発展していくことが予想される．

これらより，IoT機器開発者は機器をWoTに対応させることで，相互運用性の問題を解決することができる．しかし，WoTやWebThings Gatewayはいずれも発展途上であり，利用者が安心して利用できるようにするために，安全面での機能向上が必要である．

例えば，WebThings Gatewayの機能のうち，機器同士の制御に用いる自動化ルール処理部分，Trigger-Action Programming[5] (TAP) について考える．WebThings Gatewayでは，機器の制御のため，事前に利用者が定義したルールに基づき，機器・サービスAの状態に応じて機器・サービスBに特定の動作を与えるためにTAPを用いている．TAPを利用することで，プログラミング言語の記述に慣れていない利用者でも，簡単なルールを設定するだけで機器を制御できるようになる．しかし，TAPではこうした制御ルール（以降，これをTAPルールと呼ぶ）を人間が設定する必要があるが，TAPルールを本来意図した通りに条

[1] 名古屋工業大学
Nagoya Insitute of Technology, Nagoya, Aichi 466–8555, Japan
[a)] ckl15149@nitech.jp

件の漏れなく設定するのは人間にとって困難を伴う．例えば，外が晴れていたら窓を開ける，という動作を設定したい場合を考える．この時，人間は『外が晴れならば窓を開ける』というルールを直感的に考えがちである．しかしながら，実際は『外が晴れ かつ 住居内に人が存在する かつ 部屋の冷暖房が有効でない かつ ... ならば窓をあける』といったように，人間が無意識的に想定している条件も正しく TAP ルールに組み込まなければ，利用者が意図しない不適切な挙動となり不適切である．また，設定ミスによる機器の不適切な挙動が大きな問題を引き起こす可能性がある．例えば，設定ミスにより人がいない状態で窓が開いてしまった場合，不審者による住居への侵入を許しかねない．

本研究では，Web of Things と WebThings Gateway の両方を拡張し，設定ミスが疑われる挙動を自動的に判別，中断させる手法を検討する．

本論文の構成は次の通りである．二章では研究で利用する要素技術とその課題について述べる．三章では TAP における問題点について述べる．四章では関連研究について述べる．五章で提案手法について述べた後，六章でその手法の実装，七章で評価について述べる．八章で課題，九章でまとめを述べる．

2. 要素技術

2.1 Web of Things

Web of Things (WoT) とは，IoT 機器の相互運用性を高め，プラットフォーム間での分断を防ぐために策定されているオープンな規格・アーキテクチャである．W3C 主導で策定が進められている．

WoT に対応した IoT 機器は，自身の状態や操作を提供する API を公開する．API は Thing Description[6] と呼ばれる JSON-LD フォーマットのデータを返し，各状態の取得や操作を行うためのエンドポイントを提示する．エンドポイントは，HTTP や CoAP など，様々なプロトコルを用いて提供され得る．他の IoT 機器や端末は提示されたエンドポイントにアクセスし，データ取得や操作を行うことができる．

2.2 WebThings Gateway

WebThings Gateway とは，WoT を利用した IoT ゲートウェイの実装であり，Mozilla がリリースしている．FLOSS で Raspberry Pi 向けイメージも公開されており，誰でも無償で利用可能である．Add-on と呼ばれる拡張機能を利用・開発することで，WoT を利用していない機器やサービスを仮想的な WoT 対応機器として認識し，その他の機器と連携させることができる．

WebThings Gateway は，Thing Description を HTTP を利用した API 経由で公開する．また，管理下の機器に対し HTTP を利用した RESTful API のエンドポイントを提供する．したがって，本論文では特に指定が無い限り，各 IoT 機器へのアクセスには HTTP を用いるものとする．

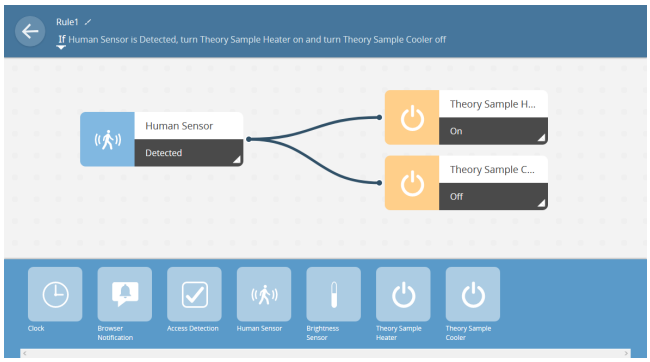

図 1 WebThings Gateway における TAP ルール編集画面

IoT 機器同士の連携ルール作成は WebThings Gateway が提供する管理画面より，TAP を用いて行う．この実際の画面を図 1 に示す．TAP は管理画面から編集が可能であり，図 1 の通り，プログラムを一切記述する必要がないのが特徴である．ルールを作成するには，必要なアイコンをドラッグアンドドロップし，対象の機器に実行させたい動作の内容を決めるだけでよい．

2.3 課題

WoT と WebThings Gateway はいずれも発展途上であり，セキュリティや機能の面で完全ではないと考えられる．例えば，WoT のセキュリティとプライバシーについてのガイドラインは，あくまで Group Note[7] に留まっている．また，WebThings Gateway に関しては，機器同士の連携に TAP を用いるため，TAP が抱える課題をそのまま受け継いでいる．本研究では，主にこの TAP 部分の改善を取り扱う．TAP における問題の具体例は次章で述べる．

3. TAP における問題

3.1 問題の概要

図 1 の画面のように，TAP はプログラミングを知らない人間でも手軽に制御ルールを作成可能という点で優れている．しかし，TAP には問題がある．それは，バグのない TAP ルールを作ることは難しい，という点である．

例えば，天気が晴れの時に窓を開ける，というルールを考える．一見これは正しそうに思えるが，

- 外出中の開閉の是非
- 深夜等，時間帯ごとの開閉の是非
- エアコン等が稼働している際の開閉の是非

など，さらに考慮を要する点は多い．これらを人間がすべて把握した上で，正しく指定するのは容易でない．

3.2 既存の解決手法と欠点

これを防ぐための手法の一つとして Capability を利用したものが考えられる．Capability とは，機器が持つ機能を

表 1 WebThings Gateway に実装されている Capabilities

Capability 名	示す種別
Alarm	アラーム
BinarySensor	二値センサー
Camera	カメラ
ColorControl	色の制御
ColorSensor	色センサー
DoorSensor	ドアセンサー
EnergyMonitor	電力使用量モニター
LeakSensor	漏電センサー
Light	電灯
Lock	電子錠
MotionSensor	モーションセンサー
MultiLevelSensor	レベルセンサー
MultiLevelSwitch	レベル制御スイッチ
OnOffSwitch	オンオフスイッチ
PushButton	押下型ボタン
SmartPlug	スマートプラグ
TemperatureSensor	温度センサー
Thermostat	サーモスタット
VideoCamera	ビデオカメラ

カテゴライズしたものである．例えば，機器に WindowControl という Capability が与えられているとき，その機器は窓の制御機能を持つということを示す．WebThings Gateway においても Capability の概念があり，いくつか実装されている [8]．一覧を**表 1** に示す．

Capability を用いれば，AirConditioner が有効のとき，WindowControl を open にする操作を禁止する，といった指定が可能になる．しかし，この手法では，プラットフォームが提供する Capability の範囲でしか IoT 機器をカテゴライズできないという制約がある．例えば WebThings Gateway では，テレビ等，表 1 に存在しない機能を持つ機器についてカテゴライズできない．したがって，WoT という共通規格を利用する様々な IoT 機器が開発される上で，プラットフォームが各機器に適切な Capability を追加し，IoT 機器がその Capability を実装して対応しない限り，3.1 節で述べた問題を解決できない．また，Capability は機器の機能の種別であるから，WoT 対応の機器の数が増えていくにつれて定義も増加することが見込まれる．そのため，成熟した場合に膨大な量の定義が追加される可能性がある．

4. 関連研究

TAP は既に様々な観点から研究がなされているが，そのうち以下の二種類については本研究と関係性が深い．具体的にどの部分と関係性が深いのかについては 5 章で述べる．

4.1 TAP ルール自体の改善

TAP ルール自体を改善することで不正な挙動の除去を試みている研究がある．例えば AutoTap[9] では，通常の TAP ルールに加え，ルール実行に際して満たすべき条件であるプロパティという概念を新たに定義し，プロパティから生成した TAP ルールと既存の TAP ルールを Büchi オートマトンを利用して自動的に合成することで，条件を確実に満たす TAP ルールを生成する．

4.2 動的解析による不正な挙動の検出

TAP の実行環境を動的解析することで，不正な挙動の検知を試みている研究がある．例えば IoTGuard[10] では，SmartThings[11] や IFTTT[12] における TAP の実行環境であるアプリに着目し，アプリ自体にコードを注入した上でそれを動的解析することにより，予め定義されたポリシーに違反する不適切な挙動を検出する．

5. 提案手法

本章では，5.1 節で提案手法のアプローチを述べ，提案手法において新たに導入する概念について詳しく説明する．次に，5.2 節で関連研究に対する提案手法の利点を述べる．次に 5.3 節で提案手法の構成について述べ，5.4 で提案手法の挙動について述べる．

5.1 アプローチ

Capability による機器のカテゴライズは，機器が持つ機能を抽象化して把握するうえで有用である．しかし，機器が何に対して作用するかを表す指標ではないため，機器間で阻害し合うような動作を形式的に表現できない．本研究では，機器が直接作用する対象で機器をカテゴライズすることで，機器の協調動作を目指す．具体的には，作用する対象として『環境』という概念を導入する．

環境とは，天候，湿度，温度，明るさ，開放の有無といった要素である．IoT 機器が動作するということは，機器の機能がこうした環境に対して作用するということである．この環境同士の関係性を定義することで，上記に示した TAP の課題を解決する．例えば，温度に変化が加えられているとき（自動的な）開放はされるべきでない，といった具合である．

このとき，どのような環境が存在するのかの定義は当然必要である．提案手法では，この定義を『定義データ』という概念を導入することで管理する．

さらに，定義された環境を機器の機能や動作と紐付ける必要がある．提案手法では，これを『評価式』という概念を導入することで実現する．

また，各 IoT 機器を WebThings Gateway に登録する際，該当機器を設置する空間の指定も要求するよう変更する．これらにより，ユーザーが本来意図していない不適切なルールの実行を防止する．新たに導入する三つの概念について，以下で詳しく説明する．

5.1.1 環境

環境とは，IoT 機器が設置された空間において機器が作用する対象である．例えば天候，湿度，温度，明るさ，開放の有無などが挙げられる．環境はグローバル環境とローカル環境の二つに分類される．グローバル環境とは，ユーザーが管理する複数の空間で共通して有効な環境のことである．例えば，天候という環境は家の各部屋の空間だけでなく家全体で共通である．したがって，これはグローバル環境である．ローカル環境はその逆で，特定の空間のみで有効な環境のことである．例えば，温度という環境は家の各部屋の空間ごとに異なる．したがって，これはローカル環境である．

各環境は，値とベクタという二種類の状態を持ちうる．値は，その環境がどうなっているのかの姿を示す．例えば温度であれば，暑い，中庸，寒いといった具合である．ベクタは，その環境の変化を示す．例えば温度であれば，上昇，下降，外気取り込み中といった具合である．ただし，変化を一意に定義できない，あるいは困難である環境も存在する．その場合，該当の環境はベクタの定義を持たない．環境の具体例を**表 2** に示す．

環境はゲートウェイによって管理される．IoT 機器の提供する値の変更がある度，後述する評価式を利用して自身の環境を更新する．

5.1.2 定義データ

定義データとは，5.1.1 で述べた環境を定義するデータである．また，環境ごとに定義された基本的な遷移禁止ルールも持つ．遷移禁止ルールとは，ある環境が状態 A であるとき，状態 B になることを禁止するルールである．この例を**表 3** に示す．これにより，例えば部屋が暑い状態にも関わらずヒーターを有効にする，といった挙動を禁止できる．これらは事前に定義され，IoT ゲートウェイ起動時に何らかの方法で読み込ませる．一例として，定義データの JSON ファイルを予め配布しておき，起動時に読み込ませる手法が考えられる．

5.1.3 評価式

評価式とは，各 IoT 機器から提供される実行可能な JavaScript コードである．評価式は各 IoT 機器の RESTful API のエンドポイントごとに存在し，Thing Description の一部として提供される．評価式を含むエンドポイントの定義の具体例を**図 2** に示す．評価式をゲートウェイが事前に取得しておき，評価式にエンドポイントから取得したセンサー等の値を代入し実行することで，新しい環境の値やベクタを得ることができる．図 2 では，5 行目の vector ブロックがベクタに相当する．ゲートウェイにより，vector ブロック内の評価式は各リクエストの直前とセンサー値の変化時に実行される．$input にはリクエストの値（機器へ送信する値）が入り，$value には現在の値が入る．$isE はゲートウェイ側がリクエスト直前に評価式を実行し，実行可否を判定する場合に true となる．r() の引数に undefined 以外が与えられたとき，それを新たなベクタとする．ただし，ベクタの更新は$isE が false の場合のみ（結果が確実な場合のみ）行われる．各変数と関数はゲートウェイ側で代入され，実行される．

```
1  // 電源を管理するエンドポイント
2  on: {
3    '@type': 'OnOffProperty',
4    // 環境(温度)のベクタを更新する評価式
5    vector: {temperature: `if ($isE) { r($input ?
         'up' : undefined) } else { r($value ? 'up
         ' : undefined) }`},
6    // リクエスト禁止を明示する評価式
7    blockOn: {temperature: `r($input && ($status.
         temperature === 'hot' || $vector.
         temperature !== undefined))`},
8    label: 'On/Off',
9    name: 'on',
10   type: 'boolean',
11   value: false,
12 },
```

図 2 評価式を含むエンドポイントの定義例

また，この仕組みを応用して，各 IoT 機器はエンドポイントごとにリクエストに対する禁止ルールとなる評価式を設定できる．例えば，環境 1 が状態 A のとき，電源を入れるリクエストのみを禁止するといったことを評価式を通して明示できる．これにより，機器の開発者がその機器にとって不適切な動作というものを定義可能になり，危険・不適と考えられる用途を制限することが可能になる．図 2 では，7 行目の blockOn がこれに相当する．ゲートウェイは他の IoT 機器などからリクエストを受け付けると，blockOn 内の評価式を実行し，結果に応じてそのリクエストを受け付けるかを判断する．$status, $vector にはゲートウェイが保持している環境の値が入る．関数 r() の引数に true が指定されたとき，リクエスト禁止と解釈される．各変数と関数はゲートウェイ側で代入され，実行される．

5.2 利点

5.1.1 で提案した環境の概念は．Capability と比較して相対的に量が少なくて済むと考えられる．機器の種類が増えたとしても，機器が作用する環境の種類は限られるためである．例えば，部屋の温度を変えている最中に窓を開けない，というルールを考える．Capability を使った場合，扇風機，エアコン，ストーブといった機器と窓という製品に対して個々に関係性を定義する必要がある．しかし，環境の概念を利用すると，製品ごとに新たに定義を与える必要はない．各製品は RESTful API のエンドポイントに 5.1.3 で提案した評価式を公開するだけでよい．ゲートウェイがこの評価式を実行することで，その製品がどの環境に影響

表 2　環境の具体例

環境名	意味	種別	取り得る値	取り得るベクタ
weather	天候	global	sunny, cloudy, rainy, stormy, foggy	
temperature	温度	local	hot, neutral, cold	up, down, neutralize
fire	火事の有無	global	yes, no	
time	時間帯	global	early_morning, morning, noon, afternoon, evening, night	
sensitivity	繊細な状態の有無	local	specific, areawide	exposing

表 3　遷移禁止ルールの例

対象環境	トリガー	禁止する遷移
temperature	値が hot のとき	ベクタが up となるのを禁止

を与えるのかを判断することができるためである．したがって，Capability の充実を待つことなく，様々な新しい形態の機器に対応が可能になる．

既存研究に対する利点を示す．AutoTap[9] では，条件の考慮漏れがないルールを作成するためにプロパティと呼ばれる条件をユーザーが入力する必要がある．したがって，このプロパティが不十分である場合，作成されるルールも不十分になる可能性がある．本提案であれば，事前に入手している定義データと IoT 機器から提供された評価式さえあれば，自動的に挙動を防止することが可能である．また，IoTGuard[10] では，不正な挙動を判別するためにポリシーが必要であるが，そのポリシーは機器の種別単位で設定する必要がある．したがって，対応する機器種別が増えるに従い，設定するポリシー数も単調に増加する．本研究であれば，環境という単位で判断を行うため，機器種別の単位で判断を行うのに比べて事前定義する必要があるルール数を削減できると考えられる．

5.3　構成

構成図を図 3 に示す．図の中央にあるゲートウェイ，WebThings Gateway がすべての機器・ルールを管理する．IoT 機器の通信はすべてゲートウェイを経由し，NAT により外部のネットワークからの通信は制限する．IoT 機器はゲートウェイに登録されており，設置場所の設定を持つ．各 IoT 機器とゲートウェイは無線 LAN (IEEE 802.11) 経由で接続される．また，IoT 機器の操作のために別途端末が利用され，こちらは WebThings Gateway と同一のネットワーク上に配置されるものとする．操作はすべてゲートウェイの提供する管理画面から行われる．

5.4　動作

提案手法における IoT ゲートウェイの動作は，以下の三つのパターンに大別される．

(1) 機器 A が機器 B に対して WoT の API に対する通信を行う場合，Gateway がそれを中継する．その際，機器 B の設置場所における環境及び機器 B の評価式からゲートウェイが不適切と判定したリクエストは遮断する．

(2) ゲートウェイが，ゲートウェイに設定された TAP ルールに基づき動作を行う場合，環境及び評価式からゲートウェイが不適切と判定した場合は動作を中断する．

(3) 機器自身が，ゲートウェイを介さずに処理を行い，かつそれがゲートウェイから見て不適切であると判断された場合，中断は行わない．これは，例えば IoT 機器に対してリモコン等でユーザーが直接働きかけた場合などが想定される．

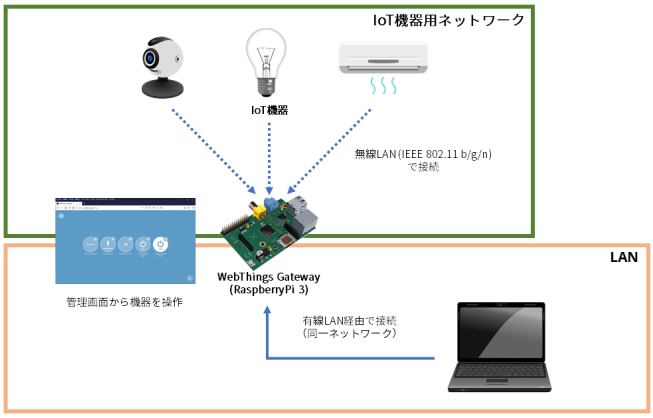

図 3　提案手法構成図

ここで，ゲートウェイにより不適切と判定される処理は，以下の三つの条件のいずれかを満たすものである．

(1) IoT 機器が設置されている空間における現在の環境のベクタと，当該動作によって IoT 機器が空間に及ぼす環境のベクタが異なる場合

(2) 当該動作によって空間に与える影響が，定義データの遷移禁止ルールに該当する場合

(3) 当該動作におけるリクエスト禁止ルール評価式の実行結果が true となった場合

6.　実装

図 4 に実装の模式図を示す．本実装は，WebThings Gateway を拡張し，提案手法の機能を実装した．また，提案手法を実装した IoT 機器の振る舞いをプログラム上でシミュレートするため，当該プログラムを Add-on として実装し，ゲートウェイ上で動作させた．ゲートウェイ，Add-on 共に，プログラミング言語である JavaScript, TypeScript を用いて開発を行い，Node.js 上で動作させた．開発は Docker コンテナ上で行い，そこで簡易的な動作検証を行った後，

Vol.2020-DPS-182 No.49
Vol.2020-CSEC-88 No.49
2020/3/13

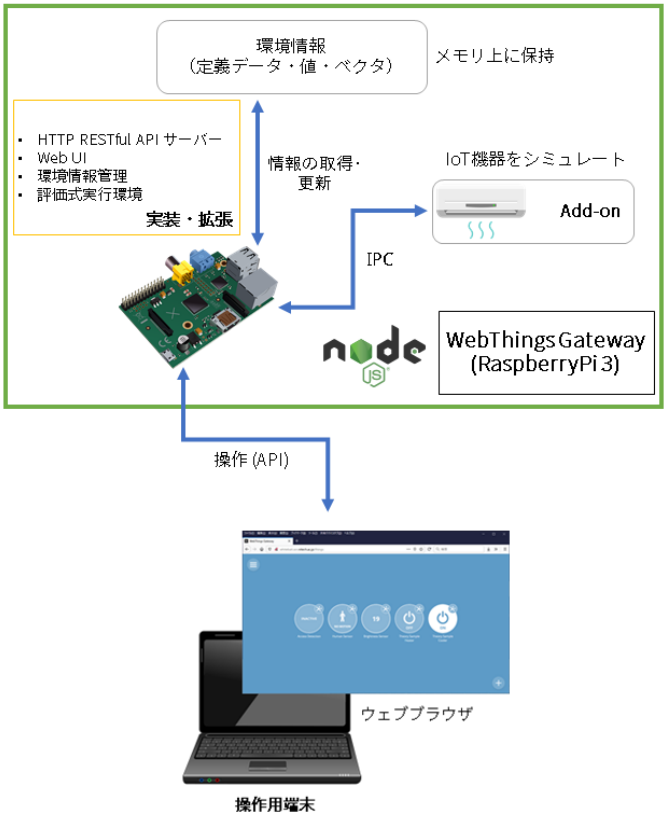

図 4　実装の模式図

実験用の Raspberry Pi へ双方のソースコードを転送，ビルド，実行した．

7. 評価

実装した機能が正しく動作し，また有効であるかを検証するため，**表 4** に示す機器の環境で評価を行った．評価は，WebThings Gateway と同一ネットワーク上に存在する別の端末から管理画面（図 4 のウェブブラウザを参照）を操作することで行った．結果は，操作後の管理画面における表示と，IoT 機器をシミュレートする Add-on の内部状態の動作で確認した．評価の準備として，事前にゲートウェイに On/Off スイッチ，ヒーター，クーラーの機器を登録した．各機器は Add-on によって振る舞いがシミュレートされた仮想機器である．Add-on として記述した各機器の定義を次に示す．On/Off スイッチの定義は**図 5**，ヒーターの定義は**図 6**，クーラーの定義は**図 7** である．vector に記載がある通り，ヒーターは有効時に温度の環境のベクタを up に変化させるが，クーラーは down に変化させると規定している．また，blockOn に記載がある通り，部屋が暑い場合にヒーターの有効化を禁止し，寒い場合にクーラーの有効化を禁止している．ただし，今回の評価では blockOn は結果に影響を与えない．この際，各機器の設置場所にはすべて同一の場所を指定した．

評価に用いた定義データを**図 8** に示す．定義データ中に

表 4　評価環境

機器名	Raspberry Pi 3 Model B+
OS	Raspbian GNU/Linux 10 (buster)
ランタイム	Node.js v8.16.1
WebThings Gateway	v0.10 をベースに提案手法を実装したもの

```
{
  name: 'Simple Switch',
  '@type': ['OnOffSwitch'],
  description: 'Simple Switch Device',
  properties: {
    on: {
      '@type': 'OnOffProperty',
      blockOn: {},
      status: {},
      label: 'On/Off',
      name: 'on',
      type: 'boolean',
      value: false,
    },
  },
}
```

図 5　評価に用いた On/Off スイッチの定義

```
{
  name: 'Theory Sample Heater',
  '@type': ['OnOffSwitch'],
  description: 'Theory Sample Heater Device',
  properties: {
    on: {
      '@type': 'OnOffProperty',
      vector: {temperature: `if ($isE) { r(
          $input ? 'up' : undefined) } else { r(
          $value ? 'up' : undefined) }`},
      blockOn: {temperature: `r($input && (
          $status.temperature === 'hot' ||
          $vector.temperature !== undefined))`},
      label: 'On/Off',
      name: 'on',
      type: 'boolean',
      value: false,
    },
  },
}
```

図 6　評価に用いたヒーターの定義

は本来，遷移禁止ルールを含む transitions というブロックが存在するが，こちらは今回の評価の対象外であり，結果に影響を及ぼさないため割愛する．評価内容とその結果を以下に述べる．

7.1 評価 1: TAP において，単一のルールに望ましくない挙動が含まれるケース

環境に対して相反する影響を及ぼす動作が単一のルールに含まれる場合の挙動を検証するため，以下の通り単一の TAP ルールを定義した．

```
{
  name: 'Theory Sample Cooler',
  '@type': ['OnOffSwitch'],
  description: 'Theory Sample Cooler Device',
  properties: {
    on: {
      '@type': 'OnOffProperty',
      vector: {temperature: `if ($isE) { r(
          $input ? 'down' : undefined) } else {
          r($value ? 'down' : undefined) }`},
      blockOn: {temperature: `r($input && (
          $status.temperature === 'cold' ||
          $vector.temperature !== undefined))`},
      label: 'On/Off',
      name: 'on',
      type: 'boolean',
      value: false,
    },
  },
}
```

図 7　評価に用いたクーラーの定義

```
{
    "laws": {
        "temperature": {
            "states": [
                "hot",
                "neutral",
                "cold"
            ],
            "vectors": [
                "up",
                "down",
                "neutralize"
            ],
            "isGlobal": false
        }
    }
}
```

図 8　評価に用いた定義データ

- スイッチが On のとき，ヒーターの電源を有効化し，かつクーラーの電源を有効化する

5.4 で述べた通り，これは不適切な動作として認識される．したがって，スイッチを On にした際，ヒーターとクーラーの両方が有効になることはあってはならない．

この状態で，WebThings Gateway の管理画面からスイッチを有効にし，ヒーターとクーラーがどのように反応するかを確認する．結果，スイッチを On にしたところ，ヒーターが有効になる一方でクーラーは有効にならなかった．動作を確認したところ，以下の流れでクーラーの動作が拒否されたことが分かった．

(1) スイッチが有効になり，ヒーターの評価式が実行される．このとき環境は何のベクタも保持していないため，そのままヒーターが有効になる．
(2) ヒーターが有効になった段階で温度の環境のベクタが up になる．
(3) そのため，次に実行されたクーラーの評価式から得られたベクタ（down）と，温度の環境のベクタが不一致となる．
(4) ベクタの不一致を理由として，実行を拒否した．

これは想定された挙動である．この結果より，提案手法が機能していることが確認された．

7.2 評価 2: TAP において，複数のルールの噛み合わせにより，望ましくない挙動になるケース

環境に対して相反する影響を及ぼす動作が複数のルールにまたがって含まれる場合の挙動を検証するため，以下の通り複数の TAP ルールを定義した．

ルール 1:
- ヒーターの電源が有効のとき，スイッチを On にする

ルール 2:
- スイッチが On のとき，クーラーの電源を有効化する

7.1 とは異なり，各ルール単体では 5.4 の不適切な動作とはならない．しかし，ルール 1，ルール 2 の順で実行された場合問題となる．したがって，ヒーターを有効にした後，最終的にヒーターとクーラーの両方が有効になることはあってはならない．

この状態で，WebThings Gateway の管理画面からヒーターを有効にし，最終的にヒーターとクーラーがどのように反応するかを確認する．結果，ヒーターを有効にしたところ，スイッチが On になる一方でクーラーは有効にならなかった．動作を確認したところ，以下の流れでクーラーの動作が拒否されたことが分かった．

(1) ヒーターが有効になる．温度の環境のベクタが up になる．
(2) 図 5 より，スイッチは評価式を持たないので，そのままスイッチが On となる．
(3) スイッチが On となり，クーラーを有効にするルールが実行される．
(4) クーラーの評価式が実行され，そこから得られたベクタ（down）と，温度の環境のベクタが不一致となる．
(5) ベクタの不一致を理由として，実行を拒否した．

これは想定された挙動である．この結果より，提案手法が機能していることが確認された．

8. 課題

8.1 ゲートウェイを介さない操作に対する挙動について

現在，ゲートウェイを介さない操作が機器に対して行われた場合，特に対応をしない．これは，例えば機器が不具合を起こした時や非常時，物理的に行った操作に対して対

応をすると，かえって危険な可能性があるためである．しかし，この仮説が正しいのか，何らかの手法で検証する必要性がある．

8.2 TAP ルール登録時におけるチェック

現在，TAP ルール登録時にはその他 TAP ルールとの整合性を取る仕組みを導入していない．そのため，実際に不適切なルールであると判定されるには，実際にそのルールが実行される必要がある．しかし，TAP のルールを定義する段階で不適切かどうかを判定することができれば，ユーザーはルール作成段階で修正を行うことができる．TAP のルール修正については AutoTap[9] を含め，いくつかの関連研究が存在する．今後はそういった研究を踏まえ，ルール定義時に確認ができる仕組みを検討する．

8.3 環境の概念の正当性に対する評価

本研究では，機器そのものやその種別ごとに制約を適用するのではなく，環境に対して制約を適用することで，事前に定義する条件を減らすことを提案した．しかしながら，現時点において，機器そのものやその種別ごとの数よりも環境の数の方が少ないという客観的な根拠を提示できていない．

また，本研究では，環境や状態遷移のルールを定義するため，定義データを導入した．しかし，運用する上でこの定義データをどうメンテナンスするのかについて，十分考察できているとは言い難い．したがって，今後はこの仮説が正しいことを証明し，かつメンテナンスコストについても詳細に考察する必要がある．

8.4 網羅的な評価

本研究では，TAP に着目し二種類の評価を実施した．しかし，現実世界のルールに対して網羅的に評価できているとは言い難い．今後は関連研究を踏まえ，さらに多くの評価を実施する．

9. まとめ

IoT 機器向けのオープンな規格・アーキテクチャである Web of Things と，それを利用した IoT ゲートウェイである WebThings Gateway を拡張し，Trigger-Action Programming における不適切な挙動を防ぐ手法を提案した．提案手法では，既存研究を踏まえた上で，新たに『環境』『定義データ』『評価式』の概念を導入した．評価の結果，一定の効果が得られることが確認できた．しかし，多くの課題が残っている．今後はそれらの解決を図ると共に，さらなる機能向上を行う．

参考文献

[1] 総務省：例話元年版 情報通信白書，総務省（オンライン），入手先 ⟨https://www.soumu.go.jp/johotsusintokei/whitepaper/⟩（参照 2020-02-14）．
[2] IDC: Double-Digit Growth Expected in the Smart Home Market, Says IDC, IDC (online), available from ⟨https://www.idc.com/getdoc.jsp?containerId=prUS44971219⟩ (accessed 2020-02-14).
[3] W3C: W3C Web of Things at W3C, W3C (online), available from ⟨https://www.w3.org/WoT/⟩ (accessed 2020-02-14).
[4] Mozilla: WebThings Gateway, Mozilla (online), available from ⟨https://iot.mozilla.org/gateway/⟩ (accessed 2020-02-14).
[5] Ur, B., McManus, E., Pak Yong Ho, M. and Littman, M. L.: Practical Trigger-Action Programming in the Smart Home, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, Association for Computing Machinery, p. 803–812 (online), DOI: 10.1145/2556288.2557420 (2014).
[6] W3C: Web of Things (WoT) Thing Description, W3C (online), available from ⟨https://www.w3.org/TR/wot-thing-description/⟩ (accessed 2020-02-14).
[7] W3C: Web of Things (WoT) Security and Privacy Guidelines, W3C (online), available from ⟨https://www.w3.org/TR/wot-security/⟩ (accessed 2020-02-14).
[8] Mozilla: WoT Capability Schemas, Mozilla (online), available from ⟨https://iot.mozilla.org/schemas/#capabilities⟩ (accessed 2020-02-14).
[9] Zhang, L., He, W., Martinez, J., Brackenbury, N., Lu, S. and Ur, B.: AutoTap: Synthesizing and Repairing Trigger-Action Programs Using LTL Properties, *Proceedings of the 41st International Conference on Software Engineering*, IEEE Press, p. 281–291 (online), DOI: 10.1109/ICSE.2019.00043 (2019).
[10] Celik, Z. B., Tan, G. and McDaniel, P.: IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT, *Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, (online), available from ⟨https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_07A-1_Celik_paper.pdf⟩ (2019).
[11] SmartThings Inc.: SmartThings. Add a little smartness to your things., SmartThings Inc. (online), available from ⟨https://www.smartthings.com/⟩ (accessed 2020-02-14).
[12] IFTTT: IFTTT: Every thing works better together, IFTTT (online), available from ⟨https://ifttt.com/⟩ (accessed 2020-02-14).