2010年度 SIGSC研

センサ駆動連携サービスのための サービス競合検出手法に関する検討

神戸大学
稲田 卓也，池上 弘祐，まつ本真佑，中村 匡秀

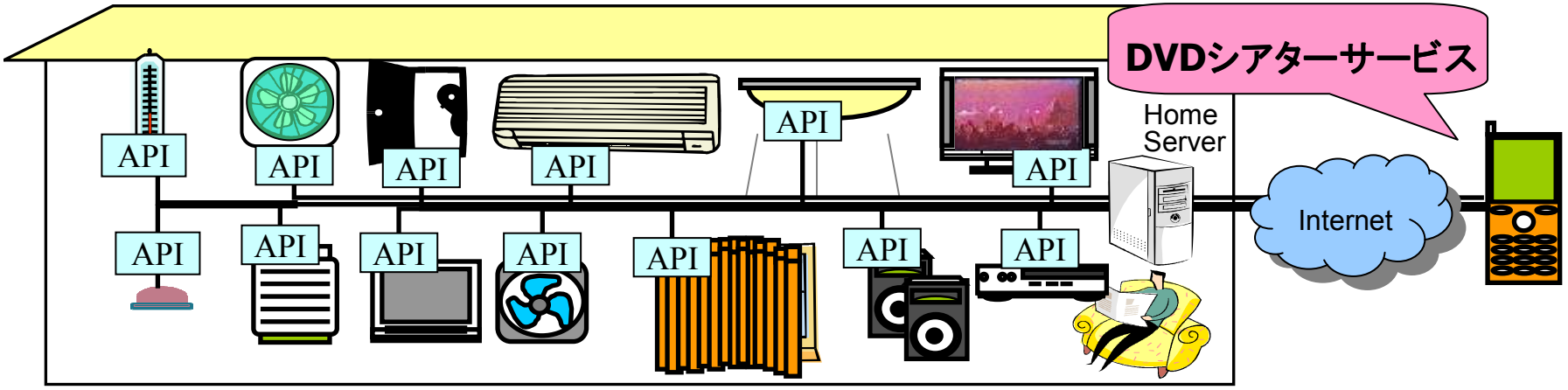
東京工科大学
井垣宏

ホームネットワークシステム(HNS)

- HNSは家電機器やセンサを家庭内のネットワークに接続し，制御するシステム．

- 複数の機器を協調動作させることで，連携サービスを実現．

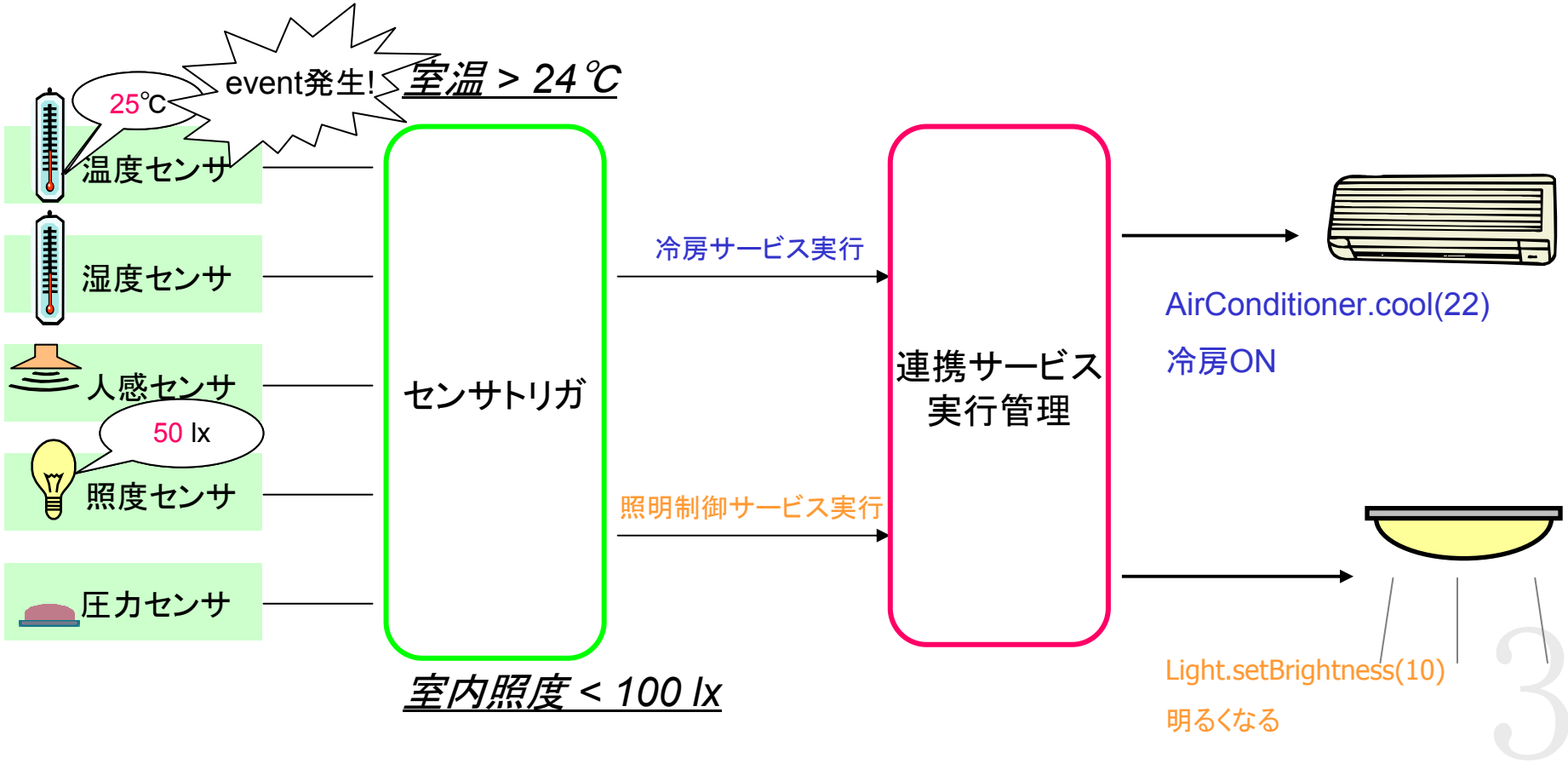
→ 家電機器を単体で動作させるより，付加価値が高い．



API:外部ソフトウェアから機器を制御するインタフェース(機器メソッド)

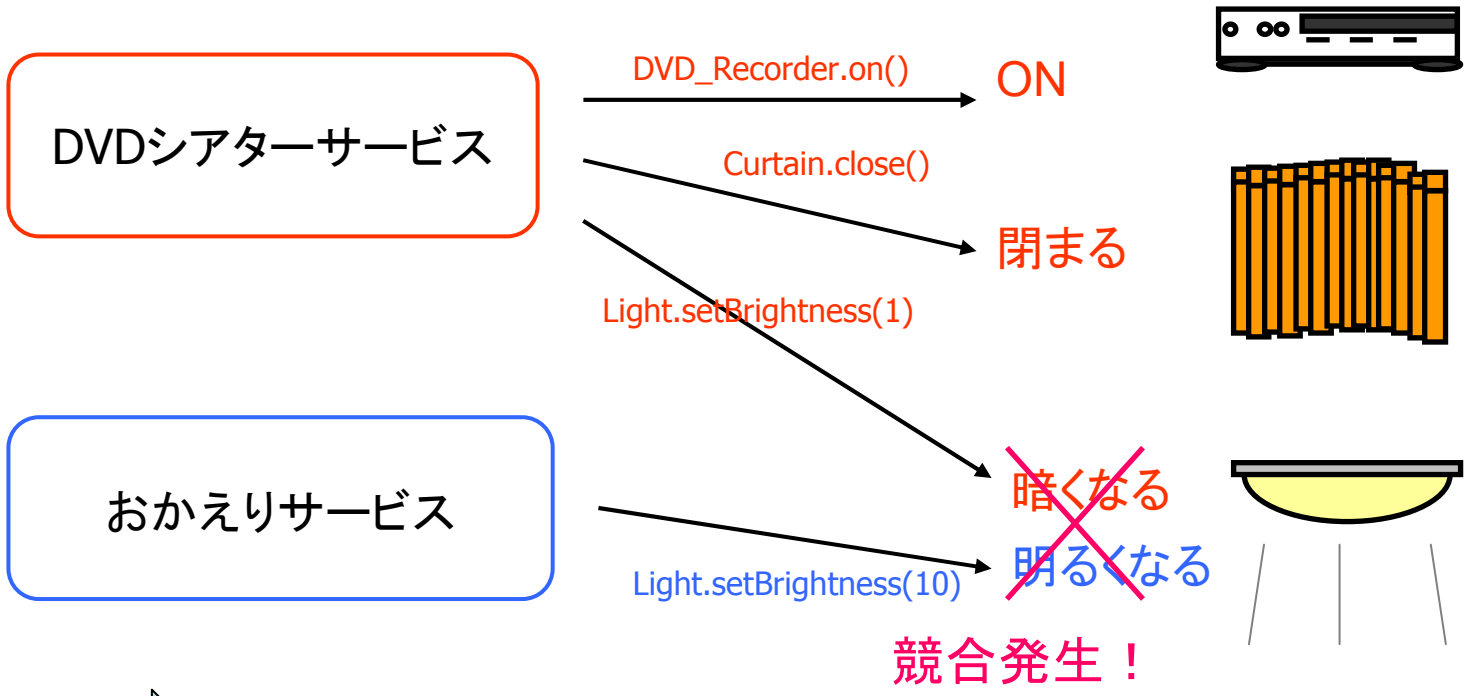
センサ駆動連携サービス

- センサによって環境の変化をイベントとして検知し，イベントに応じた振る舞いを提供するサービス．(センササービス)
- あらかじめ登録された条件式を環境値が満たせば，連携サービスを呼び出す．



先行研究：HNSにおけるサービス競合

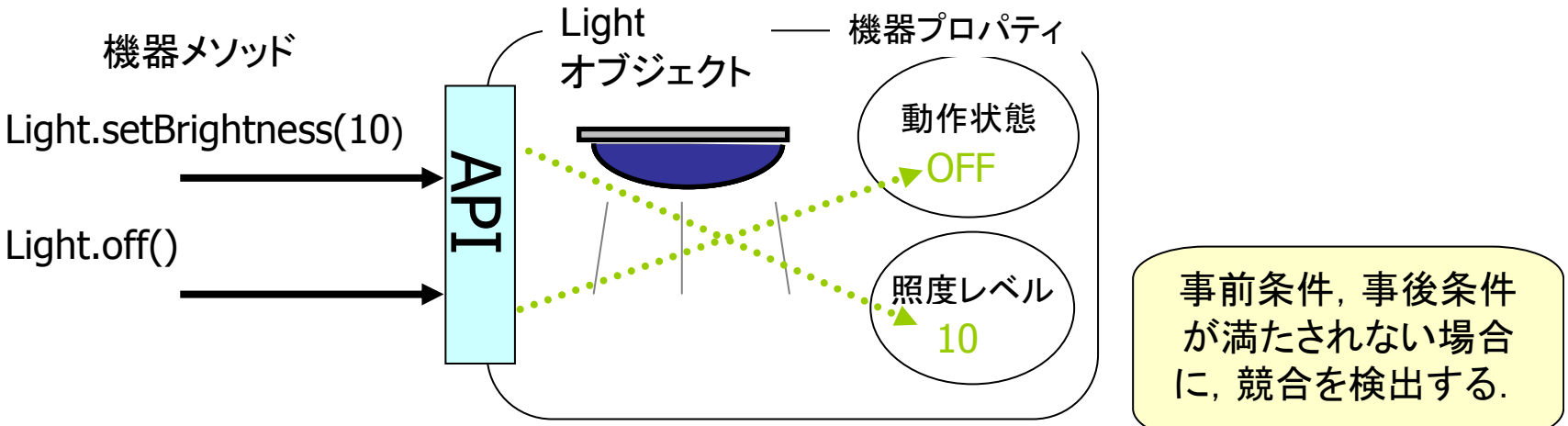
- 複数の連携サービスを同時に実行すると，
  互いに干渉・衝突を起こし，
  ユーザの意図した通りに動作しなくなる現象．
  (Feature Interaction)



ユーザの利便性，快適性を損なうので，検出・解消が求められる．

先行研究：競合検出のための家電機器モデル

HNS機器は機器の状態と制御インターフェース(API)を持つ．
- 状態→プロパティ，インターフェース→メソッドとするオブジェクトとみなせる．



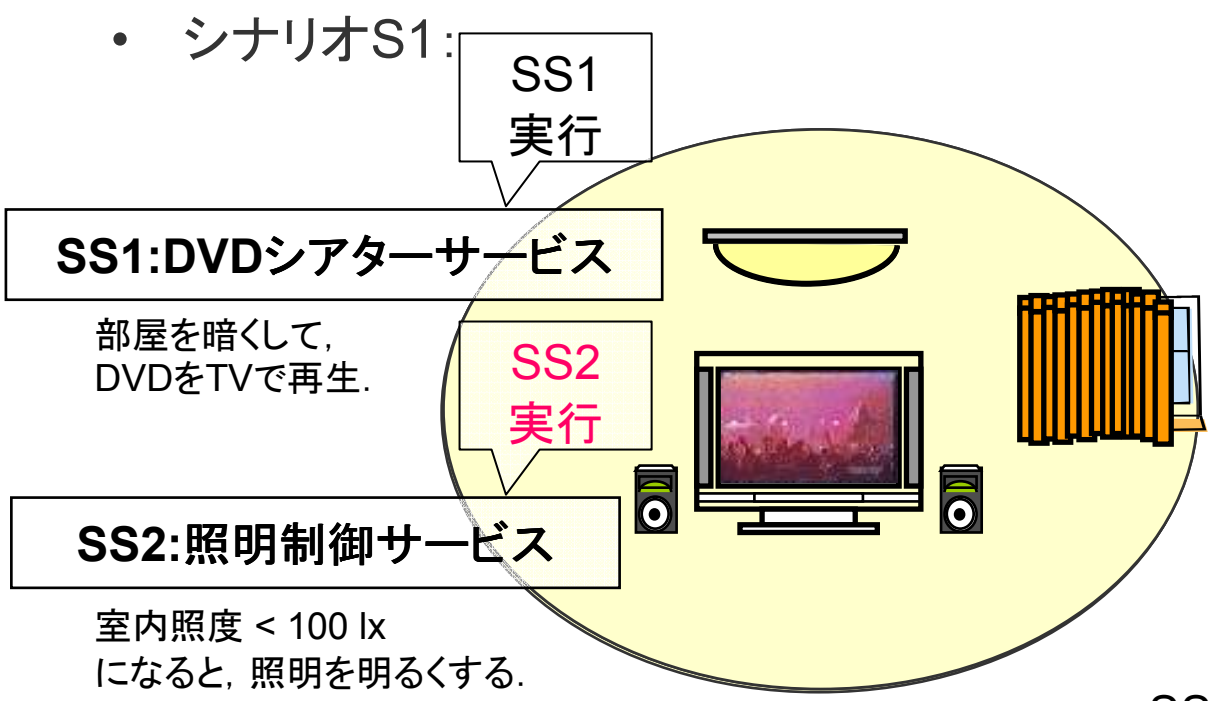
メソッドの実行によって，機器プロパティが参照・更新．
→内部実装はカプセル化し，プロパティの事前条件・事後条件のみでメソッドをモデル化．

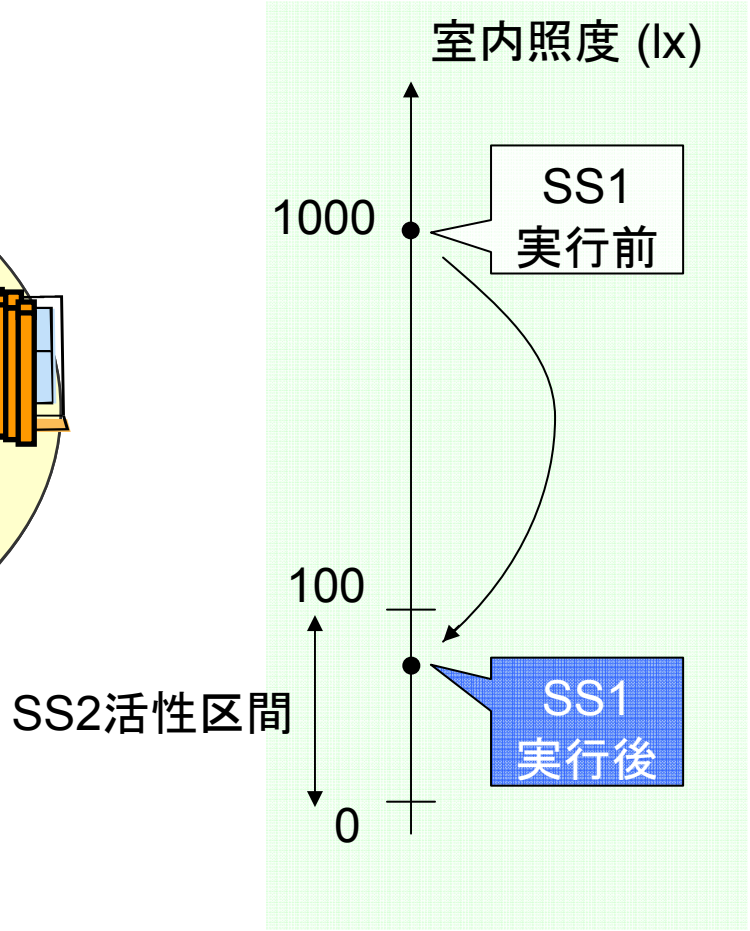
Light.setBrightness(int brightness)：**事前条件**：動作状態＝＝ON　**事後条件**：照度レベル＝＝brightness
Light.off()　：**事前条件**：＊　**事後条件**：動作状態＝＝OFF

機器＝(機器プロパティ，機器メソッド)
機器メソッド＝(事前条件，事後条件)

センササービスにおけるサービス競合

- シナリオS1:





SS1の実行結果がSS2の実行を誘発してしまう.

→ 連鎖と定義.

サービスの連鎖によって引き起こされる不具合をセンササービスにおけるサービス競合と呼ぶ.

しかし, 先行研究ではサービスのイベント部分を考慮していなかったため, 連鎖の検出は不可能.

目的とアプローチ

- 目的
  - センサ駆動連携サービスにおける連鎖検出手法の確立.
- アプローチ
  - ECA(Event,Condition,Action)ルールにもとづいた，センササービスモデルの構築.
  - 環境への影響を考慮したメソッドモデルの構築.
  - 新しいモデルを利用した連鎖検出手法の提案.

ECA ルールにもとづくセンササービスモデル

- S_event:
  - サービスを実行する契機となる，機器や環境のプロパティから構成される条件式.
  - 例「室温が28 ℃を超える」

    env.Temperature > 28(℃)

- S_condition:
  - S_eventが検知された際に，S_actionの実行可否を判断するための条件式.

- S_action:
  - 従来の連携サービス記述における連携サービスシナリオ(機器メソッドの系列) .

サービス記述例

SS_1：DVDシアター

```
S_event
 env.ServiceButton == true;

S_condition : *

S_action :
  begin(){
    DVD_Recorder.play();
    TV.on();
    TV.setChageInput(1);
    Curtain.close();
    Light.setBrightness(1);
  }
```

SS_2: 照明制御

```
S_event :
  env.Brightness < 100(lx);

S_condition :
  env.PeopleNum > 0

S_action :
  begin(){
    Light.setBrightness(10);
  }
```

環境への影響を考慮したメソッドモデル

•S_actionが持つメソッド毎に、事前条件、事後条件に加えME_futureを導入.

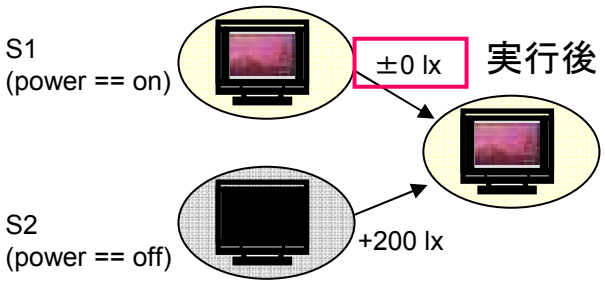
- ME_future:
  - 機器メソッドを実行し続けたときに，特定の環境プロパティに与える最終的な影響の大きさと方向を示す.
  - 状態遷移機械モデルを用いて表現.

表1.TVのME_future

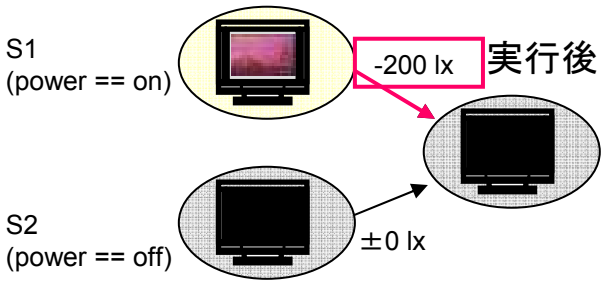
		Method	
		on	off
State	S1 (power == on)	–	ΔB == −200 ／ S2
	S2 (power == off)	ΔB = +200 ／ S1	–

ΔB = Brightnessへの影響値

TV.on() 実行前



TV.off() 実行前



連鎖検出手法(1/2)

SS1:DVDシアターサービスがどのような状況で実行されると，SS2:照明制御サービスに連鎖するかを検出する.

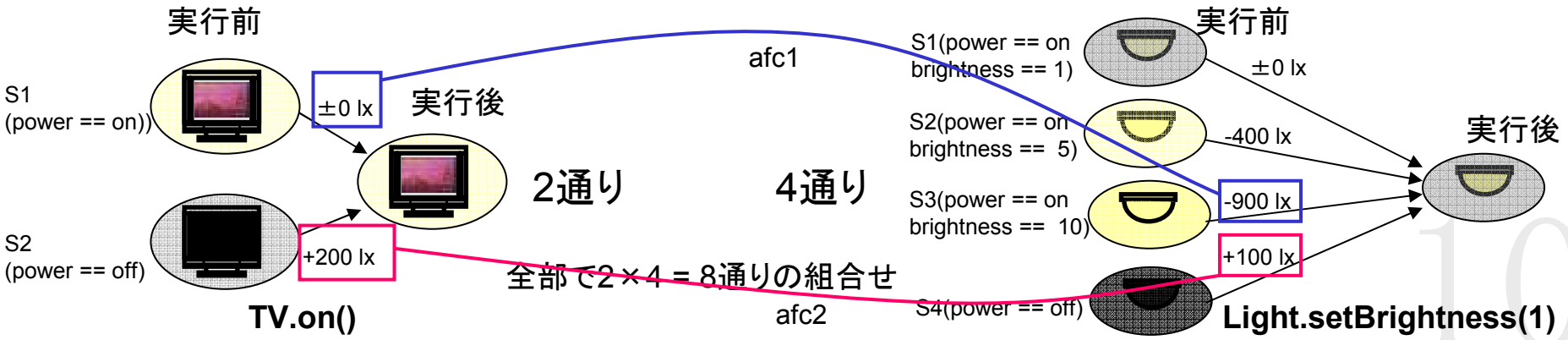
Step1：SS2からイベント **env.Brightness < 100(lx)** を抽出.

Step2：イベントに含まれるプロパティ **env.Brightness** を抽出.

Step3：SS1の持つメソッドから，env.Brightnessに影響を与えるメソッドを抽出.
Light.setBrightness(1)と**TV.on()**が抜き出される.

Step4，5：メソッドのME_futureを抽出し，全組合わせを求める.

Step6：算出した組み合わせに対してME_futureを加算. (組み合わせが多いので2例扱う)
afc1 = TV.on(){S1->S1} + Light.setBrightness(1){S3->S1} = -900
afc2 = TV.on(){S2->S1} + Light.setBrightness(1){S4->S1} = +300

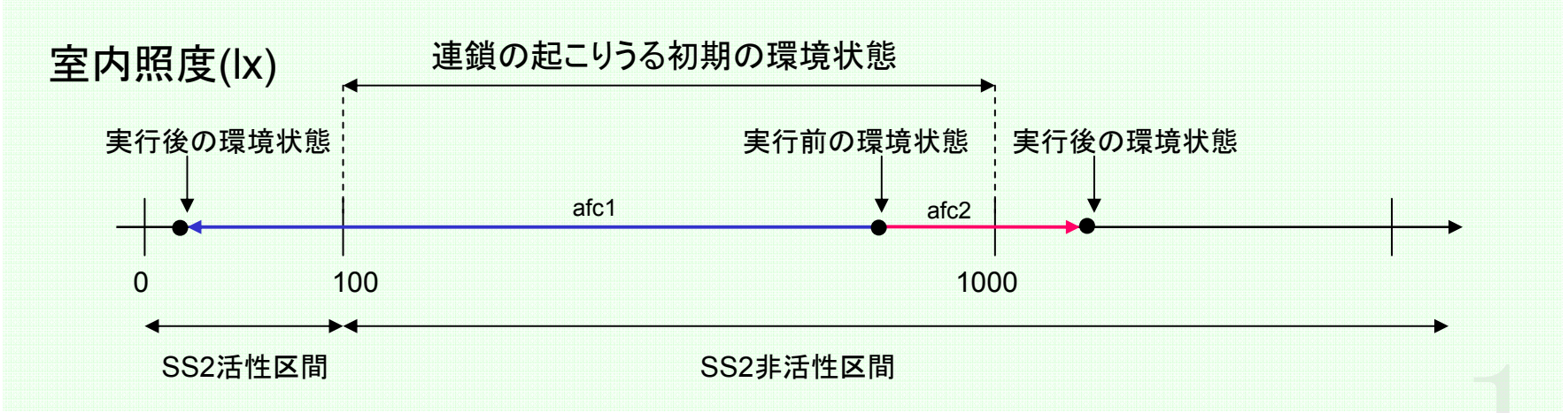


連鎖検出手法(2/2)

Step7：SS2のイベントの補集合 ***env.Brightness >= 100*** を算出し，
env.Brightness < 100 に遷移せしめるafcを抽出．

Step8：連鎖を起こす可能性がある初期の環境状態および機器状態を導出．
*afc*1 は**100 <= *env.Brightnss* < 1000** の環境下で，
TV がS1(power==on),
Light がS3(brightness==10)
の機器状態のときSS1 を実行すれば，SS2 が連鎖することがわかる．

afc1 = -900
~~afc2 = +300~~ → 照度を増加させる方向なので除外



考察

- ケーススタディにおいて検出手法を用いて連鎖の検出を行った.
  - 連鎖(DVD シアター&照明制御)
  - 連鎖(暖房&冷房)

- 考察
  - サービス間の連鎖がどのような状況で発生するのかを具体的に検出することが可能となった.
  - 現在では，環境状態によって*ME_future* の値が変わるような機器メソッド（Curtain.open()等)は扱えない.

まとめ

- センササービス間の連鎖を検出する手法を提案した.
  - ECAルールにもとづくサービスモデル，メソッド毎にME_fururを導入.
  - サービス間の連鎖がどのような状況でどのサービス間に発生するのかを具体的に検出することが可能となった.
- ケーススタディによって，提案の有効性を確認した.
- 今後の課題
  - 連鎖に起因する連携サービス間競合の定式化.
  - 検出・解消モデルの構築.

ご清聴ありがとうございました.

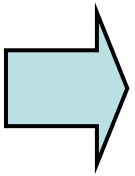
連携サービスシナリオ

- 連携サービスは任意の機器メソッドを組み合わせたもの.
- 機器メソッドの系列を連携サービスシナリオと呼ぶ.

DVDシアターサービス

映画館の雰囲気でDVD鑑賞

ユーザがサービス開始ボタンを押下すると,
DVDレコーダーがDVD再生モードで起動,
TVがDVDレコーダー入力モードで起動,
カーテンが閉まり, 照明が暗くなり,
DVDが再生.



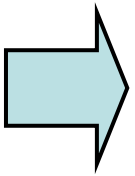
DVDシアターサービスシナリオ

```
DVD_Recorder.on();
DVD_Recorder.changeDVDMode();
TV.on();
TV.changeInput(1);
Curtain.close();
Light.setBrightness(1);
Speaker.changeMode(5.1ch);
DVD_Recorder.play();
```

おかえりサービス

帰宅時に照明を調節

センサーがユーザの帰宅を感知すると,
照明が明るくなる.



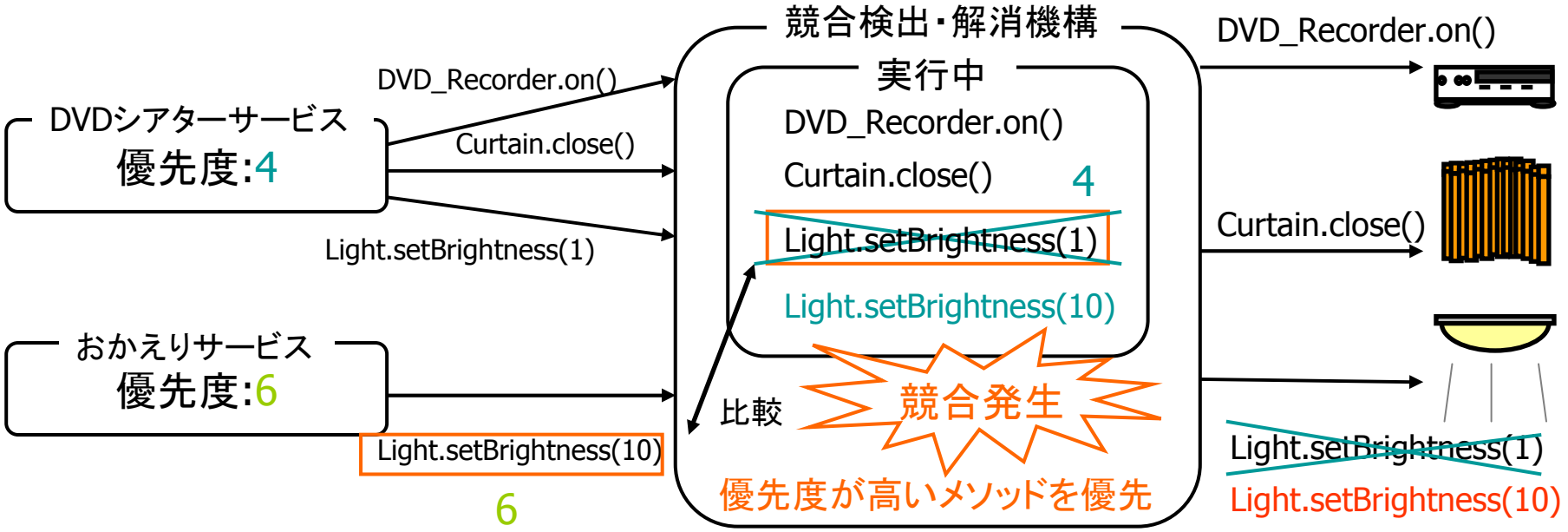
おかえりサービスシナリオ

```
Light.setBrightness(10);
```

先行研究：機器競合の検出・解消の流れ

従来手法

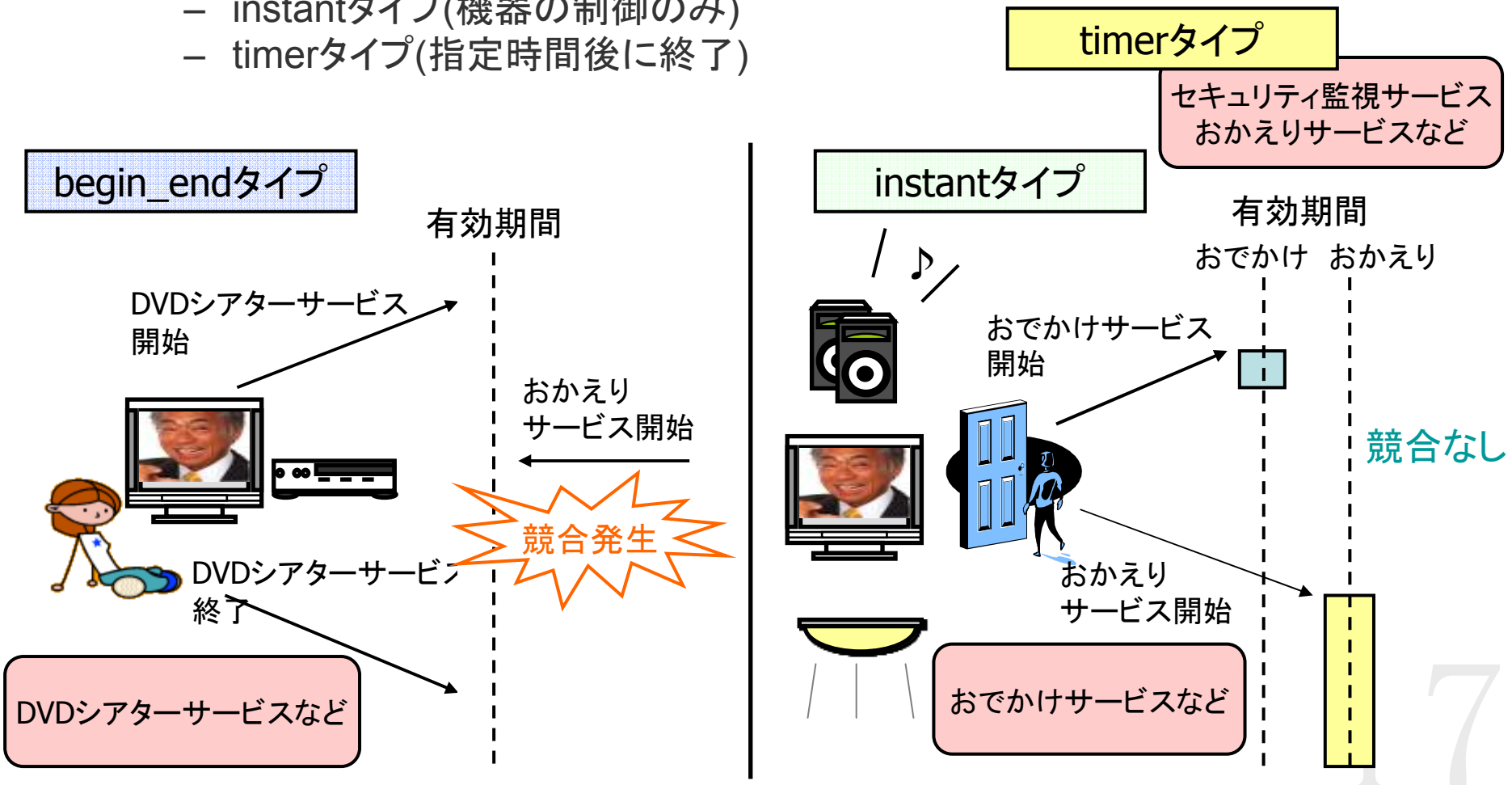
サービス競合の動的な検出と，優先度による単純な解消が可能.



1. 実行中メソッドと新規実行メソッドを比較する.
2. お互いのメソッドの目的が相反すれば競合を検出する.
3. 優先度が高い方のメソッドを優先して実行する.

A1:アクチベーション

- 連携サービスに，競合検出の対象となる有効期間を与える.
  - begin_endタイプ(明示的に終了)
  - instantタイプ(機器の制御のみ)
  - timerタイプ(指定時間後に終了)



ME_futureの加算ルール

- 機器メソッドと環境プロパティの組ごとに加算ルールを設定.
- A1: 単純な足し合わせを行う加算ルール.
  - 例：照明機器の明るさ調節機能による照度プロパティに対する影響.
- A2: 最も大きい値を選択する.
  - 例：複数の空調機器があるようなケースで，暖房機能による室温プロパティに対する影響.
- A3: 最も小さい値を選択する.
  - 例：A2 と同様のケースで，冷房機能による室温プロパティに対する影響.