Benchmarking Smart Spaces Through Autonomous Virtual Agents

(Extended Abstract)

Mario Caruso
caruso@diag.uniroma1.it

Francesco Leotta
leotta@diag.uniroma1.it

Massimo Mecella
mecella@diag.uniroma1.it

Stavros Vassos
vassos@diag.uniroma1.it

Sapienza University of Rome, DIAG, 00185 Rome (Italy)

ABSTRACT

In the recent years there has been a growing interest in the design and implementation of smart homes. The evaluation of these approaches requires massive datasets of measurements from deployed sensors in real prototypes. While datasets obtained by real smart homes are freely available, they are not sufficient for comparing different approaches and techniques in a variety of configurations. In this work we propose a smart home dataset generation strategy based on a simulated environment populated with virtual autonomous agents, sensors and devices that allow to customize and reproduce a smart space using series of useful parameters.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Plan execution, formation, and generation*

Keywords

Smart spaces; Virtual agents; Activities of daily life

1. INTRODUCTION

In this work we propose a method for generating synthetic sensor data similar to what produced by a real smart house. Such data are typically reported in a *sensor log* consisting of records, each of which represents a sensor measurement resulting of either a sudden event (e.g., fire detection) or a periodic reading (e.g., temperature).

Sensor logs allow the study of *activities of daily life* [5] or *habits* of people. A habit is a loosely specified sequence of high-level actions aiming at a particular goal, e.g., cleaning the house. The way a habit is performed may portray a high degree of variability between different users or even between the same user in different time frames, which makes a declarative approach to modeling very well suited. In particular, a *habit template* is defined in DECLARE [4] as a set of tasks (depicted as boxes) and a set of temporal constraints (depicted using arrows) as in Figure 2. The semantics is obtained through a translation to Linear Temporal Logic (LTL) such that a habit template corresponds to the logical conjunction of DECLARE constraints expressed in LTL.

An *habit instance* is a sequence of high-level actions that is consistent with the habit template. Each of these actions may be executed in various ways in the environment, e.g., getting food may require moving closer to the fridge, going through a closed door from another room or first switching on the light in the kitchen. In order to generate a synthetic log of sensor events, we couple habit instances with an *action theory* that represents the available low-level actions in the smart home, e.g., moving around and using devices, and their effects. Each of the *high-level actions* of a habit instance then are treated as *goals* that are pursued through planning on the basis of the action theory of *low-level* actions. In our approach we appeal to the STRIPS subset of the Planning Domain Definition Language (PDDL) [1]).

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

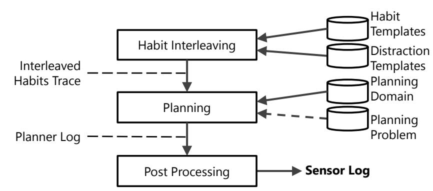


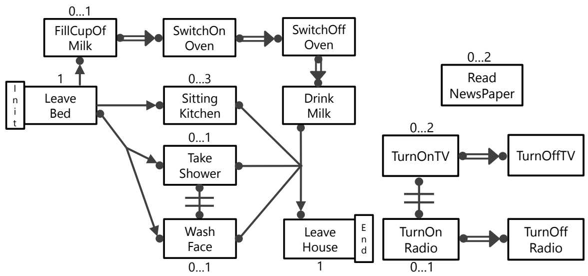
Figure 1: The sensor log generation strategy

2. DATASET GENERATION

Figure 1 depicts our dataset generation strategy that consists of three stages. Our tool takes as input *(i)* a file with habit templates in the DECLARE formalism, e.g., a morning routine; *(ii)* a file with distraction templates, e.g, talk on the phone; and *(iii)* an action theory capturing the interactions in the smart home as a PDDL planning domain.

In the first stage, an *interleaved habit trace* is produced. Our tool randomly selects a fixed number (given as a parameter) of habit and distraction templates, and generates an instance for the conjunction of the constraints of the selected templates. An example instance for the combination of the templates shown in Figure 2 along with the distraction `TalkOnThePhone` is the following:

```
(1) LeaveBed (2) TurnOnRadio (3) TalkOnThePhone (4) TakeShower
(5) ReadNewspaper (6) FillCupOfMilk (7) StartOven
(8) ReadNewspaper (9-11) SittingKitchen (12) StopOven
(13) DrinkMilk (14) TurnOffRadio (15) LeaveHouse
```



(a) `MorningRoutine` (b) `NewsOfTheWorld`

Figure 2: DECLARE models for two habits

This is repeated until a trace of the desired size is achieved, consisting of high-level actions that we call *h-actions*.

In the second stage, our tool makes use of a PDDL planning domain that specifies a virtual smart space, and produces a *planner log*. For each *h*-action in the interleaved habit trace, a corresponding planning problem is formulated, and a planner is employed to generate a sequence of atomic *p-actions* that satisfy the goal. These actions (when a solution exists) are executed online, updating the planning environment and triggering sensor measurements.

Let us consider, as an example, a PDDL domain specifying four different types of objects: `room`, `device`, `state` and `sensor`. The `device` type abstracts objects that the agent can interact with in the smart home including doors and windows, as well as sinks, showers, light switches, and appliances. Some of them are *stateful*, having a `state` associated through the `deviceState` predicate. Devices that can be used as passages between rooms are specified by a `waypoint` predicate. The arrangement of the house is specified using the predicates `(adjacent ?r1 - room ?r2 - room ?d - device)` and `(deviceAt ?d - device ?r - room)` with the obvious meaning. Finally, the smart home contains a set of `sensor` objects, each attached to either a device or a room, specified using the `senses` and `sensesRoom` predicates.

The available actions model how the agent moves in the environment (`moveToRoom*` and `moveToDevice*`) and how it interacts with devices (`changeDeviceState*` and `useDevice*`). For each of these actions, a sensorless version exists whose name does not contain a trailer asterisk (e.g., `moveToRoom`). For example, the specification of the `moveToRoom*` follows:

```
(:action moveToRoom*
:parameters (?r1 - room ?d1 - device ?r2 - room ?w - device ?s - sensor)
:precondition (and (characterAt ?r1 ?d1) (waypoint ?w)
 (adjacent ?r1 ?r2 ?w) (deviceState ?w open) (sensesRoom ?s ?r2))
:effect (and (not (characterAt ?r1 ?d1)) (characterAt ?r2 null)))
```

Similarly, `moveToDevice*` allows the character to move from one device to another belonging to the same room.

The interaction with devices can be performed either by changing the state of a device by means of a `changeDeviceState*` action, or by a `useDevice*` action that represents the execution of a task over the device without changing its state. The effect of the latter is to change the `usedDevice` predicate, which is reset after each planner execution to allow reuse of devices.

Note that for each *h*-action, one or more goal conditions are specified, into the habit templates, using the predicates of the PDDL domain (e.g., *h*-action `LeaveHouse` may be realized by pursuing the goal `(and (deviceState entranceDoor closed) (characterAt outside null)))`).

Finally, the output of our tool is the *sensor log* that lists only the generated *p*-actions; Figure 3 shows the one corresponding to the trace we saw earlier.

```
(1) LeaveBed ⇒ GOAL: usedDevice bigBedroomBed
(2) TurnOnRadio ⇒ GOAL deviceState livingRoomRadio switchedOn
 • moveToDevice* bigBedroom bigBedroomBed
      bigBedroomDoor bigBedroomDoorSensor
 • changeDeviceState* bigBedroom bigBedroomDoor
      closed open bigBedroomDoorSensor
 • moveToDevice* livingRoom null livingRoomRadio radioSensor
 • changeDeviceState* livingRoom livingRoomRadio
      switchedOff switchedOn radioSensor
(3) TalkOnThePhone ⇒ GOAL usedDevice telephone
(4) TakeShower ⇒ GOAL usedDevice bathroomShower
 • moveToDevice* corridor null bathroomDoor bathroomDoorSensor
 • changeDeviceState* corridor bathroomDoor closed open bathroomDoorSensor
(5) ReadNewspaper ⇒ GOAL usedDevice livingRoomNewspaper
(6) FillCupOfMilk ⇒ GOAL devicestate cupOfMilk filled
 • moveToDevice* corridor null kitchenDoor kitchenDoorSensor
 • changeDeviceState* corridor kitchenDoor closed open kitchenDoorSensor
(7) StartOven ⇒ GOAL deviceState kitchenOven switchedOn
 • moveToDevice* kitchen cupOfMilk kitchenOven kitchenOvenSensor
 • changeDeviceState* kitchen kitchenOven switchedOff
      switchedOn kitchenOvenSensor
(8) ReadNewspaper ⇒ GOAL usedDevice livingRoomNewspaper
(9) SittingKitchen ⇒ GOAL usedDevice kitchenTable
(10) SittingKitchen ⇒ GOAL usedDevice kitchenTable
(11) SittingKitchen ⇒ GOAL usedDevice kitchenTable
(12) StopOven ⇒ GOAL deviceState kitchenOven switchedOff
 • moveToDevice* kitchen kitchenTable kitchenOven kitchenOvenSensor
 • changeDeviceState* kitchen kitchenOven switchedOn
      switchedOff kitchenOvenSensor
(13) DrinkMilk ⇒ GOAL deviceState cupOfMilk empty
(14) TurnOffRadio⇒ GOAL deviceState livingRoomRadio switchedOff
 • moveToDevice* livingRoom null livingRoomRadio radioSensor
 • changeDeviceState* livingRoom livingRoomRadio
      switchedOn switchedOff radioSensor
(15) LeaveHouse ⇒ GOAL (and (deviceState entranceDoor closed)
                             (characterAt outside null))
 • moveToDevice* corridor null entranceDoor entranceDoorSensor
 • changeDeviceState* corridor entranceDoor closed open entranceDoorSensor
 • moveToDevice* outside null entranceDoor entranceDoorSensor
 • changeDeviceState* outside entranceDoor open closed entranceDoorSensor
```

Figure 3: A concrete example of sensor log

3. FUTURE AND RELATED WORK

By customizing the input to our tool one can tweak parameters in the high-level layer of behavior expressed through habit templates, in the low-level layer of planner actions (which specify how habits may be realized in the smart space), as well as in the configuration of the available sensors. Data generated in this way can be used as a way to experiment, test, and validate the effectiveness of different techniques related to smart spaces.

Generic tools for generating datasets of agents moving into pervasive environments are presented in [2] and [3]. The former tool works only at habit level not reaching the level of detail needed to generate a sensor log. The latter tool shows a suitable degree of detail but lacks of an evaluation over whatsoever algorithm for smart spaces; on the other hand, this work introduces the concept of daily schedule that we will incorporate in future versions of our tool.

4. REFERENCES

[1] M. Fox and D. Long. Pddl2.1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20, 2003.

[2] A. Helal, A. Mendez-Vazquez, and S. Hossain. Specification and synthesis of sensory datasets in pervasive spaces. In *Symp. on Computers and Communications*, 2009.

[3] D. Merico and R. Bisiani. An agent-based data-generation tool for situation-aware systems. In *Intl. Conf. on Intelligent Environments*, 2011.

[4] M. Pesic, H. Schonenberg, and W.M.P. van der Aalst. Declare: Full support for loosely-structured processes. In *Intl. Conf. on Ent. Distributed Object Comp.*, 2007.

[5] J. Ye, S. Dobson, and S. McKeever. Situation identification techniques in pervasive computing: A review. *Pervasive and Mobile Computing*, 8(1), 2012.