Benchmarking Smart Spaces Through Autonomous Virtual Agents

Mario Caruso, Francesco Leotta, Massimo Mecella and Stavros Vassos

SAPIENZA Università di Roma — Dipartimento di Ingegneria Informatica, Automatica e Gestionale

Objectives

In the recent years there has been a growing interest in the design and implementation of **smart homes**. The evaluation of related techniques requires massive datasets of measurements from deployed sensors in real prototypes.

CHALLENGE: Freely available smart home datasets are not sufficient for comparing different approaches and techniques in a variety of configurations.

SOLUTION: A smart home dataset generation strategy based on a **simulated environment** populated with virtual autonomous agents, **sensors and devices**.

References

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

[2] M. Pesic, H. Schonenberg, and W. van der Aalst. Declare: Full support for loosely-structured processes. EDOC 2007.

[3] C. Di Ciccio, and M. Mecella. Mining constraints for artful processes. BIS 2012.

1 Modeling daily life

The sensor log produced by a smart home can be considered as a snapshot of the inhabitants performing their **habits** (or activities of daily life [1]).

Habit

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.
- We adopt a declarative approach for the specification of habits, called DECLARE [2].

DECLARE

A declarative modeling language for business processes:

- Focuses on the (minimal) set of rules which must be satisfied in order to correctly carry out a habit.
- Each model (**habit**) is defined by a set of tasks (**h-actions**) and by a set of constraints, whose semantics of is based on Linear Temporal Logic - LTL:

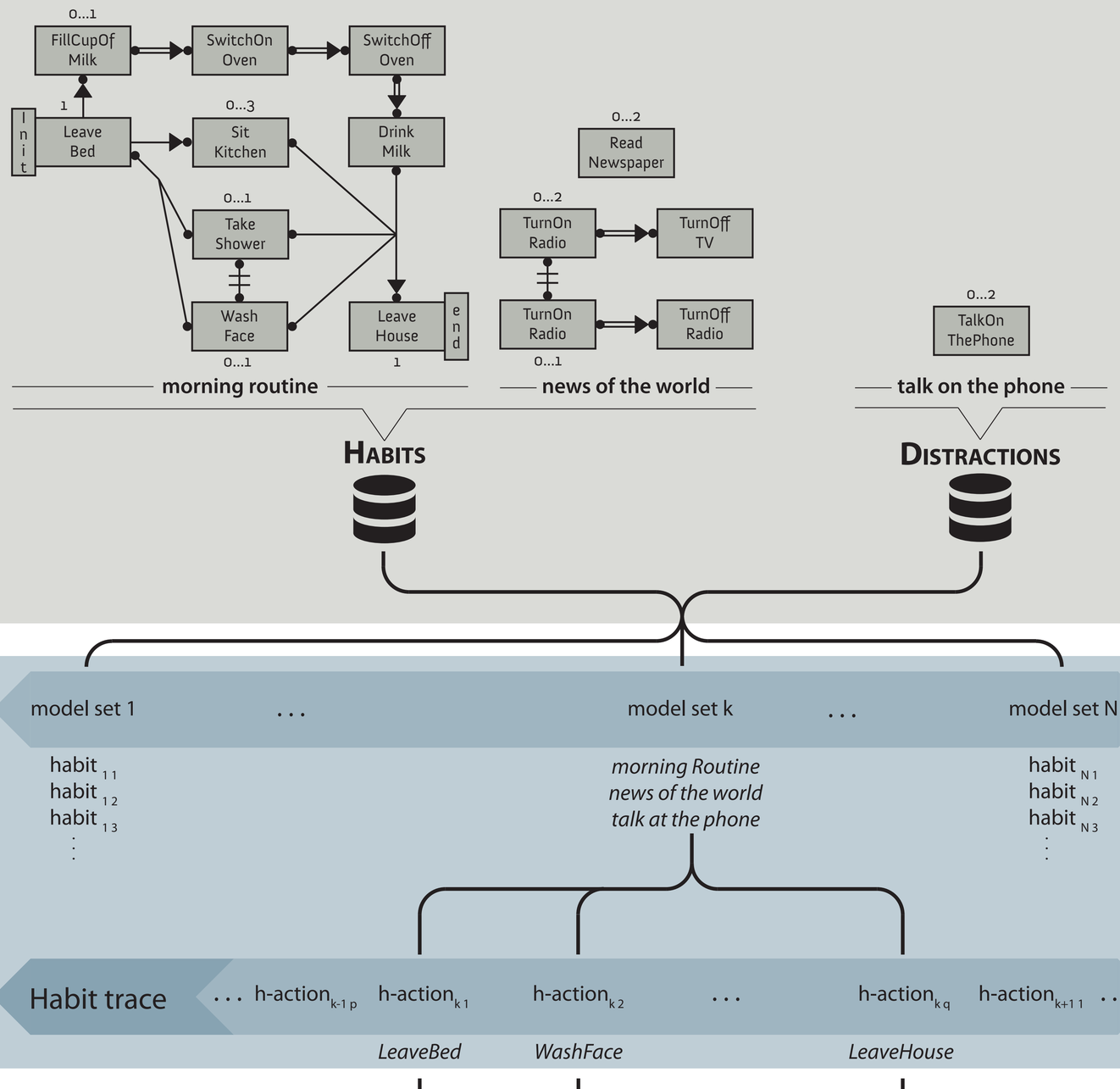
init A $init(A)$	LTL: A Activity A must be the first executed activity	A → B $response(A,B)$	LTL: $\Box(A \Rightarrow existence(B))$ If A is executed, then B must be eventually executed after A
1...* A $existence(A)$	LTL: $\Diamond A$ Activity A must be executed at least once	A → B $precedence(A,B)$	LTL: $existence(B) \Rightarrow ((\neg B)\mathcal{U}A)$ B can be executed only if A has been previously executed
N...* A $existenceN(A)$	LTL: $\Diamond(A \wedge \bigcirc existence(N-1))$ Activity A must be executed at least N times	A → B $succession(A,B)$	LTL: $response(A,B) \wedge precedence(A,B)$ A and B must be executed in succession, i.e., B must follow A and A must precede B
0 A $absence(A)$	LTL: $\Box(\neg A)$ Activity A cannot be executed	A → B $alt\_succession(A,B)$	LTL: $(precedence(A,B) \wedge \Box(B \Rightarrow \bigcirc(precedence(A,B)))) \wedge (response(A,B) \wedge \Box(A \Rightarrow \bigcirc(precedence(B,A))))$ Similar to succession, but A and B must alternate in the sequence of events.
N...* A $absenceN+1(A)$	LTL: $\neg existence(N+1)$ Activity A can be executed at most N times, i.e. the execution trace cannot contain occurrences of A	A ⇸ B $not\_coexistence(A,B)$	LTL: $(\Diamond A \Rightarrow \Box(\neg B)) \wedge (\Diamond B \Rightarrow \Box(\neg A))$ A and B exclude each other: if A is executed, then B can never be executed and vice versa

INPUT — Two repositories

An agent executes a sequence of (possibly interleaving) habits taken from two repositories:

Regular habits: goal-oriented behavior wth high variability such as i) "morning routine," ii) "news of the world"

Distractions: simpler behaviors that satisfy more immediate goals and occur arbitrarily in the daily schedule such as "talk on the phone"



2 Habit Interleaving

INPUT — Habit trace parameters

The habit trace is composed by a fixed number N of model sets. Each model set is is made up by a maximum of max_H habits and max_D distractions.

From each chosen model, a random instance [3] satisfying the constraints is extracted.

- The LTL semantics of DECLARE allow a set of templates to be combined by a simple logical AND.
- In the case of overlapping h-actions between templates, consistency is guaranteed.

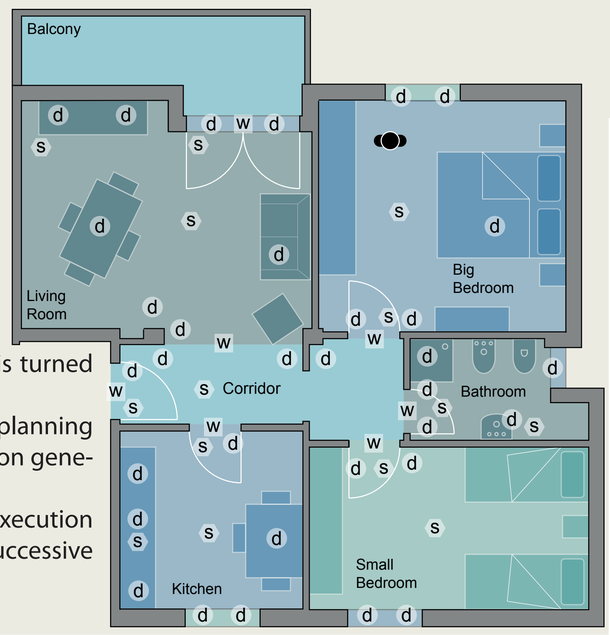
3 Continuous planning

INPUT

The planning domain introduces predicates and actions related to different components of the virtual environment:

- The position of the character expressed by the `charAt` predicate. A character can move to either a specific room or device.
- The topology expressed using the `adjacenct` predicate.
- The devices available into the environment:
  - Stateful devices has an associated state that can be changed.
  - Stateless devices can be only used.

The initial state of the world assigns the fixed predicate whose value will not change during the planning.



Each h-action in the habit traces is turned into a planning goal.

- The **Planner Log** is obtained by planning for each of them. A single h-action generates multiple **p-actions**.
- The final state of one a planner execution is used as initial state for the successive one.

4 Sensing

INPUT — Sensors

The set of sensors available in the house. Each sensor is in charge of monitoring a specific attached device/room. The type of the sensor denotes its ability to sense a certain kind of event (e.g., movement)

OUTPUT — Sensor Log

P-actions that relate to sensors are automatically translated into sensor measurements while the others are filtered out. The result is the Sensor Log that is the output of the system.
The result sensor log strongly depends on the kind and number of sensors that are part of the environment.

- The performance of algorithms for smart homes can be tested over different sets of sensors.
- Robustness of algorithms can be evaluated introducing errors in sensor measurements.
- Sensors installed into real home can be selected maximizing the performance measure