Using object memory patterns to make plan-driven help systems more flexible.

Damien CRAM\textsuperscript{a,b}, Alexander KRÖNER\textsuperscript{a}, and Alain MILLE\textsuperscript{b}

\textsuperscript{a}German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, 66123 Saarbruecken, Germany
\textsuperscript{b}Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France

Abstract

There are many strategies to assist the user in pervasive environments. In some cases, the user has to achieve a goal according to a plan known by the environment. We address the issue of enabling the environment to be aware of which task of the plan is currently on-going. We present a strategy that makes use of temporal patterns called memory chronicles and that watches events that occur in object memories to recognize if they match any memory chronicle of any task of the plan. We take the example of a smart kitchen where tools and ingredients are considered as objects with memories and where the cook reproduces a recipe according to a recipe plan known by the kitchen. We explain how the kitchen makes use of memory chronicles to applying cooking task recognition and guide the cook through the recipe plan.

Keywords. Memory Patterns, Task recognition, User assistance, Interaction Traces, Chronicles

1. Introduction

Objects from the outside world that are equipped with sensors allow to keep track of object-centric interaction traces, and then to reuse these traces. In many cases, the reuse of interactions aims to enable intelligent behaviors and to help its owner \cite{1}. In the context of pervasive environments, some other objects continually try to match events occurring in their environments to a task plan in order to infer in real-time where the user steps in that plan. Smart kitchens \cite{2,3} are such pervasive environments.

The smart kitchen of the \textit{SharedLife} project \cite{4} is equipped with a video screen and sensors (e.g. RFIDs, accelerometers) that are attached to each tool and ingredient of the environment. In that way, each object can store its event story, which makes it an object with a memory. The screen displays information and instructions to the cook. To provide the user with help, the smart kitchen is beforehand given a list of linear recipe plans that can be realized again by new cooks. When a cook enters the smart kitchen, he selects a recipe plan he wants to realize. Then the kitchen keeps up to date the knowledge of which tools enter and leave the
cooking workbench and infers in real time which step of the recipe plan the cook has to do next.

As showed in [5], this design brought notable help to users of the smart kitchen, but it also suffers from some inconvenient strictenesses. Flexibility needs to be brought into the system. The main strictness is that the cook has to stick rigorously with the given order of tasks in the recipe plan. Some tasks in cooking activities, as in most of other activities, are actually allowed to be done independently from some others. As a result, the cook was also forced to look both at the workbench and at the screen alternatively so as to ensure that he performs the task the right way.

In this paper, we propose that the recipe plan is no longer a fixed sequence of tasks, but instead a more flexible structure allowing some recipe tasks to be done according to any order (cf. section 3). Section 4 explains how the smart kitchen is still given the ability of inferring where in the recipe plan the cook steps thanks to memory chronicle recognition, despite that the smart kitchen can no longer rely on a unique order of tasks in the recipe plan.

2. Definition of object memory and memory chronicle

Object memory. In this paper, we define an object memory as follows. Given a finite set of event types \( E \), an object memory is a sequence of events, where each event is a pair \((e, t)\) such that \( e \in E \) and \( t \) is an integer called the date. For example, given an object bread-knife in the smart kitchen and \( E = \{bk\text{-appears}, bk\text{-disappears}\} \), bread-knife could have the following memory in the smart kitchen:

\[
M_0 = \langle (bk\text{-appears}, 25), (bk\text{-disappears}, 210), (bk\text{-appears}, 220), (bk\text{-disappears}, 300) \rangle.
\]

(bk stands for bread-knife) The memory \( M_0 \) can be interpreted as follows: “bread-knife appeared into the workbench at date 25 (in the smart kitchen, each date refers to the number of seconds since the cooking session started) and disappeared at date 210. Then it appeared again at date 220 and disappeared again at date 300”.

An interaction trace merges memories from different objects that are used in the context of the same activity, e.g. in the context of the smart kitchen. In the smart kitchen, several digital objects are handled by the user, e.g. bread-knife, cutting-board, bread... For example, considering only two digital objects: bread-knife and bread, the interaction trace of the smart kitchen could be:

\[
T_0 = \langle (bk\text{-appears}, 25), (b\text{-appears}, 27), (b\text{-disappears}, 100), (b\text{-appears}, 110), (bk\text{-disappears}, 210) \rangle.
\]

(b stands for bread) Event types about appearance into the workbench or disappearance are typically those collected from RFID sensors but several other event types, e.g. start-mixing Off turn-oven-on, can be collected as well.

Memory chronicles. We use chronicles [6] to model patterns that can occur in object memories, that is why we will refer to them as memory chronicles. Memory chronicles are structures that represent a typical or recurring episode in an object memory. In the case of the smart kitchen, each memory chronicle represents a cooking task. Defining memory chronicles rigourously would be too long and out of scope of this paper. Basically, a memory chronicle specifies which events must appear in the memory and gives time constraints on these events.
For example, the figure opposite shows a memory chronicle that represents any subsequence that is composed of two bk-appear and one bk-disappear, where bk-disappear occurs whenever between 100 and 200 seconds after the first bk-appear and 10 to 20 before the second bk-appear. There is only one occurrence for this memory chronicle in the memory $\mathcal{M}_0$:

$\langle (\text{bk-appear, 25}), (\text{bk-disappear, 210}), (\text{bk-appear, 220}) \rangle$

3. The hyper recipe model of the smart kitchen

A hyper recipe is a recipe in which each task is not only described textually, but also formally and with an attached video. A hyper recipe is composed of a recipe plan and a task map. (cf. Figure 1)

The recipe plan is similar to a workflow that defines all tasks to be done and their inter-dependencies. Existing workflow models, e.g. Petri Nets and its extensions, are very powerful to express a wide range of task dependencies. However, such formalisms are too complex to be handled by end-users. For this reason, we define a recipe plan as a sequence of parallel tasks. No task composition is allowed. For example on Figure 1, the recipe plan in brackets is non-valid since we could define a compound task $G$ as the sequence of subtasks task $B$ and task $D$.

Finally, the task map associates each task defined in the recipe plan with a set of resources that are used by the flexible assistant (cf. section 4): a textual explanation, a video stream that shows how the author of the recipe performed that task, and a memory chronicle. The memory chronicle describes the pattern of events in object memories that results from the realization of that task.

We keep the global architecture of sharing hyper recipes over a community that was introduced in [4]. This architecture allows two roles in particular: contributor and reuser.

The role contributor consists in adding a new recipe to the shared recipe database: the user can create and post an hyper recipe to the community. Figure 2 illustrates the role contributor. A cook member who wants to publish a new hyper recipe to the community starts with cooking that recipe in the smart kitchen under video recording (cf. step a). Then the cook starts a reflexive phase (cf. step

Figure 1. An example of a hyper recipe. The task map associates task A with Video A, Memory chronicle A and Textual description A, and so on with task B to task E.
Role contributor: creating a hyper recipe and adding it to the shared recipe database.

Role reuser: realizing a hyper recipe selected from the shared recipe database with the help of an assistant.

b) where he reuses all materials (object memories and the video stream) that came out of the cooking session to make a hyper recipe ready to be published at step c.

The role reuser consists in searching the shared recipe database for a hyper recipe to reproduce (cf. Figure 2). Once the reuser cook starts to reproduce the hyper recipe, the assistant uses memory chronicle recognition to infer which task is currently on going and helps the cook.

4. The task recognition-based assistant of the smart kitchen

The assistant tries to match current events occurring in object memories with one of the memory chronicles in the task map. If a memory chronicle matches the current events, then the assistant provides the reuser cook with help like displaying the video segment for that task.

Recognizing all occurrences of a memory chronicle in a memory is a challenging issue that was addressed in [6] through the algorithm CRS (Chronicle Recognition System). CRS keeps for each memory chronicle a set of hypotheses up to date while time passes and new events occur. Each hypothesis is a partially recognized memory chronicle. Basically, an hypothesis is composed of a set of events that have been recognized and a set of event types that are still to be recognized. Figure 3 explains this concept with the example of the memory chronicle $C_{tomatoes}$.

The flexible kitchen assistant must guess which task the cook started before that task comes to end. In our approach, it consists in choosing the best memory chronicle among a set of candidate memory chronicles. To do that, we define a very simple scoring function as follows: $f_{score}(C) = \text{“the ratio of recognized event types in the current hypothesis”}$. Figure 3 gives $f_{score}(C_{tomatoes}) = 4/8 = 50\%$. Thus, we address the partial recognition problem by setting a threshold recognition
Figure 3. Example of a hypothesis. This figure assumes that the cook is currently realizing his recipe and that the current time is 46. The sequence shows the events that occurred in the smart kitchen so far. The memory chronicle showed is the one that was defined for the task "cutting tomatoes". It is composed of 8 events regarding 4 objects. It defines the task as adding the tomatoes (\( t \)-appear) to the workbench and removing them 30 seconds to 200 seconds after, and so does it for objects cutting-board, bread-knife and bowl. So far at date 46, CRS has recognized 4 events of that memory chronicle and has formulated a hypothesis with respect to the temporal constraints of the memory chronicle, e.g. since \( G \) must occur from 30 to 200 after \( C \), and since \( C \) occurred at date 4, \( G \) is then expected between now and date 204.

Figure 4. Screenshot of the flexible recipe assistant that is displayed on the screen of the smart kitchen while the user is cooking. The left part shows the recipe plan, which has one example of parallelism: tasks Tomatoes, Onions, and Red pepper can be realized according to any order. Tasks that have been fully realized are painted in red, those who have not been started are in green, and those who are currently on-going in orange. On that example, our partial recognition system recognized that the user is currently working on task Tomatoes and automatically displayed on the right part the instructions for that task and the video of how the task was performed by the author.

ratio, e.g. 40% and by assuming that the cook is realizing the highest rated task, if there are some over the threshold. (cf. Figure 4 for a screenshot of the help provided)
5. Conclusion

The way we address the issue of “making plan-driven help systems more flexible” implies that the plan on which the help is based allows that flexibility. Since the resulting plan does not have anymore imposed order of tasks, we introduced the notion of memory chronicle partial recognition to match actions currently occurring in object memories to tasks in the recipe plan. The system has been implemented and pre-tested with “fake cooks”, i.e. by replaying actions of past recorded cooking sessions. Our first observation reported that our chronicle recognition system can detect almost every time the right task that is currently ongoing, on condition that several memory chronicles are associated to each task rather than only one, otherwise it appears sometimes that some occurrences of the task cannot be covered by a single chronicle. User studies such as [5] need to be led to observe and measure if the new system actually relaxes constraints and eases the cooking process, to cooks’ opinions.

In the case of the smart kitchen, we can think of even more flexible ways of reusing hyper recipes, like a free mode in which the reuser selects no recipe and cooks his own one. In such a mode, the smart kitchen would match current actions with tasks in all existing recipes. But our next research question concerns the design and implementation of an interface that supports the reflexive phase and gives the contributor the possibility to easily find pertinent memory chronicles. In particular, we are working on a design that uses the discovery of chronicles from interaction traces [7] to make the definition of pertinent chronicles more automatical and hence easier.

References