Towards Planning the History of a Virtual Agent

Lucie Kučerová, Cyril Brom, Rudolf Kadlec

Charles University, Faculty of Mathematics and Physics
Malostranské náměstí 2/25, Prague, Czech Republic
lucie.kucerova@mensa.cz, brom@ksvi.mff.cuni.cz, rudolf.kadlec@gmail.com

Abstract

Episodic memory is an important component of "minds" of many long-living virtual agents, for instance non-player characters from role-playing games. So far, research on episodic memory modeling in the context of these agents has focused mostly on producing the memory content on-line, that is, when the agent is being simulated. In this paper, we address a complementary issue: automatic generation of the memory content off-line. The intended use is generating memories that anticipate the start of the simulation, that is, generating history of an agent. We propose here a complex design method enabling a designer to specify high-level requirements on the agent's history and use planning to generate this history according to these requirements. We also detail that part of this method that concerns itself with the planning, and describe a prototype implementation and the first results.

Introduction

Recently, there has been increasing interest in modeling episodic memory (memory for the past events) for intelligent virtual agents. Some of that work focuses on on-line storage and later recall, including explaining by the agent (e.g. Rickel and Johnson 1999; Dias et al. 2007; Brom, Pešková, and Lukavský 2007). For example, a typical RPG (role-playing game) features a vast virtual world inhabited by tens or hundreds of virtual agents (non-player characters, NPCs) and it may be vital to equip them with episodic memory abilities (Castellano et al. 2008; Doherty and O'Riordan 2008; Brom and Lukavský 2009).

Other work in this field focuses on learning from past experience (e.g. Ho and Watson 2006; Nuxoll 2007), which can improve the performance of a virtual agent dramatically. For instance, an agent with the hunger drive can satisfy his eating need more quickly when he remembers where he saw food recently.

To our knowledge, no one has addressed yet other issue concerning episodic memory modeling – the automatic off-line generation of the memory content in cases when it is impossible or inconvenient to create it manually. For instance, if an RPG designer wants the NPCs to have memories for events that anticipate the start of the game, she has to write them manually. While this approach is befitting in the case of main characters, it can get very irritating in the moment when the designer has to model many NPCs of low importance (background characters throughout). Creating the history of the 125th palace guard is not very amusing. Another possibility for the designer is to have at her disposal a method to generate these supplementary NPCs automatically according to her wishes, which is precisely the aim of our research now.

How to approach this issue? According to some, one of the main functions of episodic memory[1] is social (e.g., Williams et al., 2008). In a nutshell, people tend to exchange their personal stories for various social reasons. As Hirst and Manier (1995, pp. 271) put it: "We cannot divorce the act of remembering from the act of communicating. ... Recollections arise... from a desire to communicate with others about the personal past." The conceptualization of (some) recollections as personal narratives brings us to the idea of generating these recollections similarly to how narratives are generated in the field of virtual storytelling.

There are more approaches to generating narratives, but arguably the most promising is planning. In our domain, using of planning would mean generating the content of episodic memory, i.e. the underlying representation enabling the agent to tell stories about himself, based on a planning domain and planning problem. This content would be generated based on the agent's initial state, his possible memories and a designer's requirements constraining the to-be-generated agent's memory. The core of a planning domain is formed by predicates and operators. Predicates can serve to describe possible agent's states, Operators have preconditions and effects, which makes them a suitable representation of actions in the virtual world, thereby of the agent's possible memories. A planning problem consists of description of the initial state

[1] Recent psychological literature tends to use the term "autobiographic memory" for what we refer to as "episodic memory". We use here the latter term for the reasons of consistency with our previous papers on this topic.

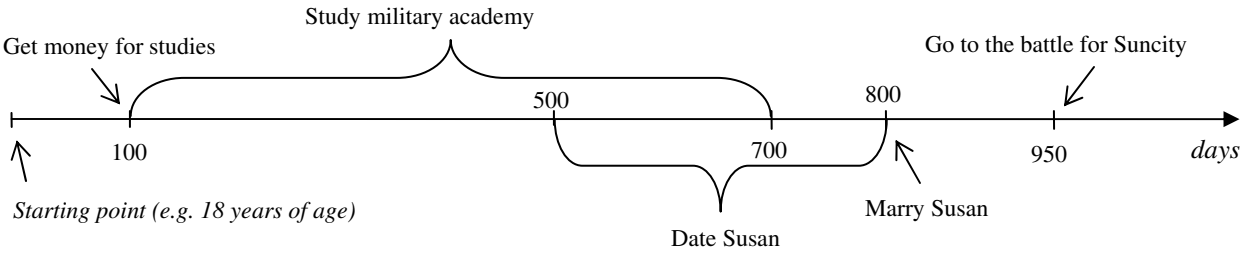

Fig. 1. Schema of content of episodic memory.
A real schema would be more complex; this one is simplified for intelligibility.

and the goal state. These two can serve to represent the designer's constraints on the to-be-generated memory.

The issue of memory contents generation is scalable in at least two ways, the length of content of episodic memory and the number of agents whose histories are being generated. For example, we can generate an agent's memories for the entire lifetime, but we can also omit childhood and/or old age as memories from these life periods and not necessary for many purposes. It is also different whether we generate memories of one agent, several agents that never interacted with each other during the time period in question, or hundreds of interconnected agents, i.e. agents who have relationships among them. Generating such agents separately could cause inconsistencies in their memories.

Each of these concepts brings different complexity. As this is, to our knowledge, the first work addressing this problem, we intentionally focus here on the less complex variants. We create memories of an agent or a few of interconnected agents spanning over several years of their adult life.

A typical agent with the content of episodic memory generated by our method would be a background character, e.g. a soldier, an ordinary mage, a villager or a shopkeeper in a fantasy RPG. Such RPG is just one of the possible applications, however, we will center on the soldier example in this text, for explanatory purposes. This agent would be able to tell the player a brief summary of his life or respond him to basic questions about his life. For instance, the soldier would be able to say in which battles he took part, when and where he got married or when he was ascended to lieutenant, as illustrated in Fig. 1.

However, we cannot expect a game designer to describe the planning domain and the planning problem directly. At the same time, we can expect that the designer may want to fine-tune the generated plan (history) manually. Thus, we foresee a necessity of a complex design method enabling the designer to work with a sort of user-friendly software tool encapsulating the very process of planning.

The goal of this paper is a) to propose such design method, and present b) a proof-of-concept implementation of the planning part and c) results of a small-scale case study.

The rest of this paper proceeds as follows: In the next section, we will introduce our problem in more detail. Then we will discuss the choice of a suitable planner. In the following part, we will present the results we have obtained so far. This will be followed by a discussion of the results and then by a short conclusion.

Problem Detailed

We will describe the problem on two levels. First, we will present the general overview of the design method, and then we shall centralize on the parts concerning generation of appropriate planning domains/problems and planning itself.

General overview

The process of generating content of episodic memory of an agent from a high-level input using our method is captured schematically in Fig. 2.

As the game designer will not want to describe the planning domain and the planning problem directly, we have to enable her to write down her intentions in a high-level, user-friendly language, using appropriate authoring tool. For this reason, we introduce Step 1 in our method workflow. This high-level definitions and requirements then have to be converted by an input-processing tool to a valid planning domain and a corresponding planning problem, which occurs in Step 2. The planner afterwards generates the appropriate plan describing content of episodic memory of the agent (Step 3). Next, the plan has to be converted back to the form suitable for the designer. This is done by an output-processing tool (Step 4). The designer then may want to make some manual changes to the created content of episodic memory. So we have supplied the workflow with Step 5.

While not trivial, Steps 1, 2, 4 and 5 are arguably much simpler than Step 3 (from the technical standpoint). Therefore, we focus on this step, addressing the preceding steps only as much as necessary. We will present them more in detail now.

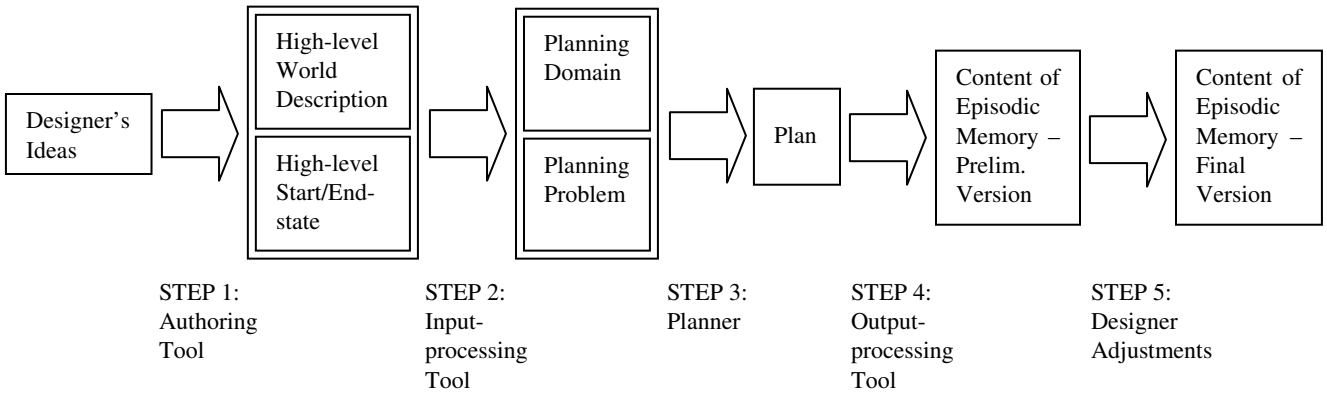

Fig. 2. Method Workflow

Input Design and Planning

Generally, in Step 1, the designer must create high-level description of the virtual world, i.e. define its topology, objects and possible actions which can be executed by an agent. At the same time, she has to specify her requirements on a concrete agent or a group of agents. Then, in Step 2, this high-level input has to be transformed to a planning domain and a planning problem.

We have not formalized the high-level language used by the authoring tool so far. Currently, we generate planning tasks manually, to verify feasibility of Step 3 and hence the method in general.

Now, we will summarize the types of conceptual requirements which can be demanded by the designer to specify the properties of content of an agent's memory and the properties of the virtual world in general. We will use the soldier example.

Requirements on the virtual world. The first essential part of the description of the virtual world is its topology (cities, villages, important places…) and objects the world contains. The designer will need a way to specify all of this. The other part is the specification of actions that are possible in the world, together with their preconditions and effects. Moreover, some real-like actions are naturally durative and permit other actions to occur during them. For example, studying military academy takes some time and a future soldier can perfectly plausibly date a girl or win some money in roulette during his studies. So we have to permit durative actions in the world definition.

Requirements on an agent. Conceptual requirements on the generation of history of an agent can be categorized into these groups:

1. General requirements on the agent's achievements or states.
2. Requirements on a concrete action.
3. Requirements on an action in a concrete time.
4. Randomness for the possibility to generate more agents from just one setting.

We will discuss these requirements in more detail now.

1. General requirements. The designer may wish to specify a general requirement on the end state of the agent, without assigning a particular way to achieve it. For example, she may want the soldier to be rich, letting completely to randomness whether he has gained the money by fighting, winning in a lottery or inheriting it.

2. Requirements on a concrete action. In some cases, the designer may want to specify a more concrete requirement: not only the end state, but also the means to achieve it. For instance, she may want the soldier to earn money by playing roulette, because it is important for the story.

3. Requirements on an action in a concrete time. Sometimes, it is important that an action occur in a concrete time. For example, the designer may want the soldier to take part in a particular battle. Since this can happen only in the moment when the battle takes place, there has to be a mechanism to accomplish this.

4. Randomness. The designer may need the agent to achieve a concrete end state (in a random way) or to perform a particular action. But she also may want to use randomness to generate other memories of the agent. In addition, for the purpose of saving her time, she may need to generate histories of several similar, but not identical, agents from the same high-level specification. Thus we need to introduce randomness in these two ways:

- **Probability of the actions.** For example, it is a lot more probable to earn money for living by working than by finding them in a secret cave. So the designer can assign probability to each of these actions.
- **Insignificant actions to "animate" a virtual agent.** The designer can mark some of the actions as "noise actions", which can be inserted to the history randomly (providing their preconditions are fulfilled). These could be actions like "go for a trip to the capital", "see a falling star" etc. They are absolutely not important for an agent's history, but can make him seem more vivid during the communication with a player.

The planning mechanism should cope with all of these requirements. Indeed, we have been addressing all of them, however, the work is still in progress and we are having several open issues, most notably with Req. 3, as detailed later.

The key decision is the appearance of the interface between Step 2 and Step 3. As PDDL (McDermott et al. 1998) seems to be actually the most used language for specifying planning problems, we decided to describe our domain and problems in this formalism.

Selecting a Planner

Many requisitions on the used planner rise from the requirements described above. We will list them now in the form of PDDL requirements, divided in two groups – essential requirements, which are indispensable for our purposes, and technical requirements, which would be useful, but are not absolutely necessary.

Essential requirements:

- **:durative-actions** – This requisition stems directly from the requirements on the virtual word.
- **:fluents** – It is necessary to include several numeric variables, e.g. amount of money or a randomly generated number to introduce uncertainty in the generated problem/plan (see Req. 4).
- **:equality** – First, this is needed for randomness (Req. 4). Second, a designer may want an action to include preconditions comparing numeric variables with a predefined value. For example, she may want to specify that a soldier can be ascended to lieutenant only after taking part in a concrete number of battles.
- **:timed-initial-literals** – Timed initial literals are needed for Req. 3 from the previous section.

Technical requirements:

- **:typing** – It is a lot more transparent to describe some preconditions of an action by specifying the type of parameters (e.g. action *get_married(v1, v2)* does not make sense with locations as its parameters). However, if typing is not available, this can also be solved by inserting predicates of type *is_person(v)*, although it makes the domain less human-readable.
- **:disjunctive-preconditions** – Many real-like actions may require disjunctive preconditions, nevertheless, these could be also formally written like several different actions.
- **:negative-preconditions** – There are many possible real-like actions which need their preconditions to include negation of a predicate. However, this can be bypassed by inserting other predicates (for example adding predicate *not_married ?person* to supplement a predicate *married ?person*).

From the listed requirements it is obvious that we need a planner supporting all levels of PDDL3 (Gerevini and Long 2005). In this, we depart from the work in virtual storytelling (e.g. Riedl and Young 2006, Porteous and Cavazza 2009), which usually do not demand so much equipped planner (although it tends to have other requirements).

Implementing a satisfactory planner would be quite demanding, so we decided to use an already existing planner, at least for present purposes. Sadly, we have not found many planners fulfilling all these requisites. At this moment, we are using SGPlan6 (Hsu and Wah 2008), since in addition to meet all our requirements, its authors are still working on it. Thus they are able to address some technical problems we have with the planner (regarding timed initial literals), which are probably caused by the fact that our domain is not a typical planning contest domain.

Results

Our present purpose was to demonstrate that our approach, the complex design method encapsulating planning, can work. This means developing a proof-of-concept implementation and obtaining results of a small-scale case study.

In this moment, we have at our disposal a planning domain created manually, bypassing the authoring and input-processing tools from Fig. 2. The domain contains definitions of about 50 operators and 100 predicates. For practical use and more thorough investigation, the domain has to be extended and we also should design another domain for tests. Nevertheless, we are already able to generate different plans over this domain for problems which differ only in randomness (Req. 4 from the section "Problem Detailed").

Our testing domain serves for generation of a prototypical soldier in a fantasy RPG. A brief extract from the domain is depicted in Fig. 3. We will now comment the marked fragments from the point of view of the designer's requirements.

Requirements on the virtual world. The actions possible in the world are translated to PDDL's durative actions. The rest of the world description is converted to constants (1).

Requirements on an agent. We will discuss the translation of these requirements to the domain one by one.

1. General requirements. Each requirement is converted to a predicate or a function (4, 3). This predicate (or change of the function) then becomes an effect of one or multiple actions (11, 8).

2. Requirements on a concrete action. These requirements are converted to two predicates (5). The first of them is an effect of exactly one action (9); the second is a sufficient precondition of this action (7), to overpower the action's probability, as explained later.

3. Requirements on an action in a concrete time. In addition to the predicates from Req. 2, this requirement

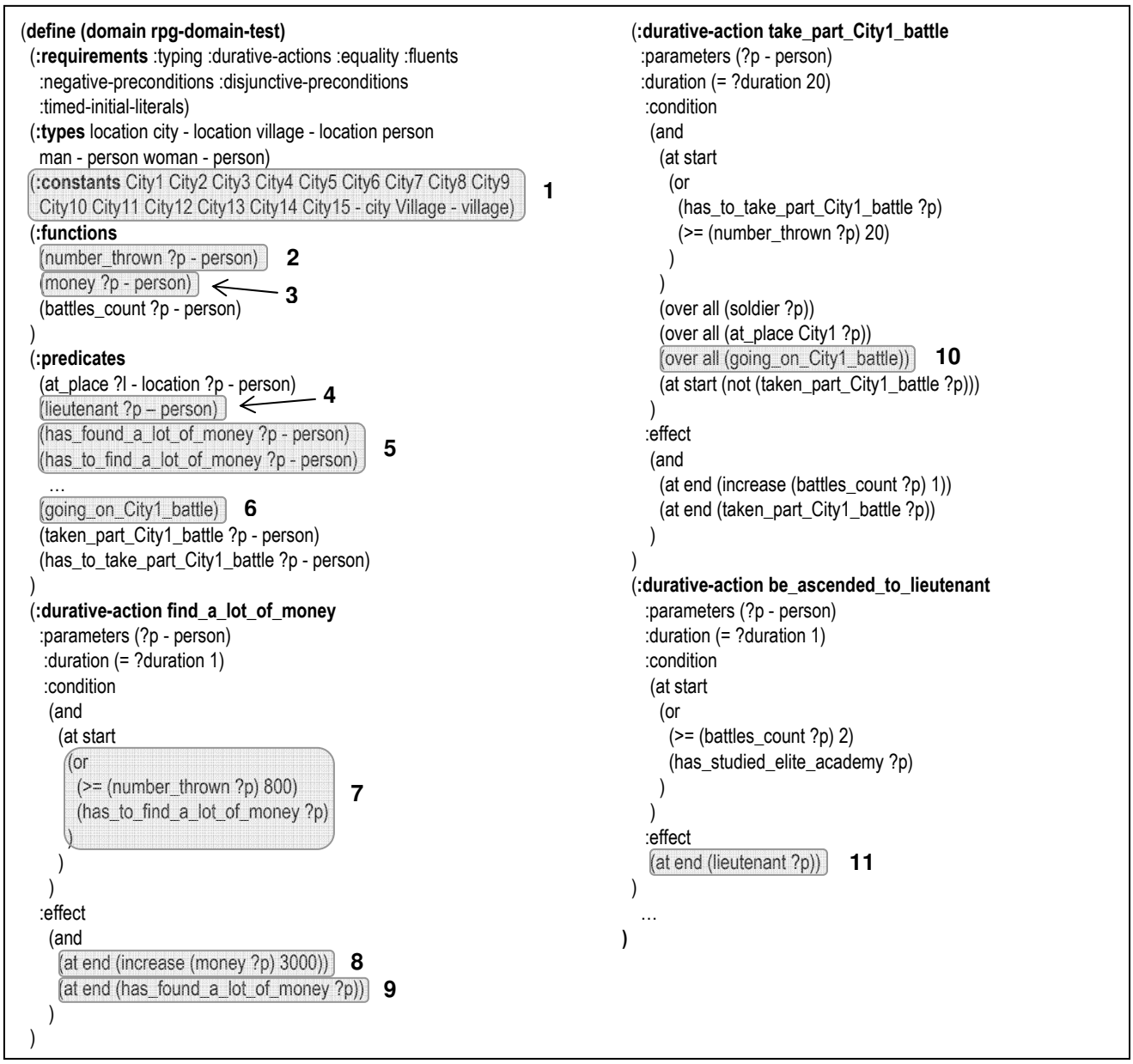

Fig. 3. An extract from the planning domain. Description in the text.

introduces other predicate (6), which is also a necessary precondition of the action in question (10).

4. Probability of the actions. For this purpose, a function representing a random number is inserted (2). It is intended that its value is generated randomly for a particular agent by an input-processing tool to the planning problem definition, as showed later. This random number is used to set the probability of particular actions (7). Presently, we set the number manually.

We will demonstrate our results now on the task of generation of history of two soldiers, which are very similar and are not interconnected. So we will generate their memory contents from one domain and one initial state and goal (with very subtle changes). At the beginning, these future soldiers are about 16 years old (we do not generate memories for childhood) and they have not studied military yet.

The high-level input from the designer that is to be converted to the PDDL could look approximately like this:

Initial state:
Name: John (James for the second soldier)
Sex: male
Initial locality: Village (City10 for the second soldier)
Money: 10
No battles so far.
Other people, who could take part in his history, however will not get in contact with the player: Stacy,

female, Village (Kate, female, City5 for the second soldier).
When the important battles take place (omitted for brevity).
Requirements (at the end of the generated period)**:**
Married. (Req. 1)
Will find a lot of money. (Req. 2)
Will take part in the battle for City1. (Req. 3)
Lieutenant of the army. (Req. 1)
Generate other one or two "noise goals". (Req. 4)

The designer is expected to formalize this input using an authoring tool. The two example planning problems which could be generated from the high-level language by an input-processing tool in Step 2 (and which were translated by us manually) are depicted in Fig. 4 and 5, with marked differences between them. Generally, the initial state was translated to the initial state of the planning problem. Most requirements were converted to the goals of the planning problem. Some of them (Req. 2, 3) must also insert some predicates into the initial state. There has been also set the value of the numeric fluent *number_thrown* – to a random number. This is necessary for the first part of Req. 4. Furthermore, there have been added effects of several randomly chosen "noise actions" to the set of goals,

```
(define (problem rpg-problem-test1)
 (:domain rpg-domain-test)
 (:objects John - man Stacy - woman)
 (:init
  (= (number_thrown John) 10)
  (= (money John) 10)
  (= (battles_count John) 0)
  (has_to_find_a_lot_of_money John)
  (has_to_take_part_City1_battle John)
  (at_place Village Stacy)
  (at_place Village John)
  (at 1280 (going_on_City1_battle))
  (at 1300 (not (going_on_City1_battle)))
  (at 1350 (going_on_City2_battle))
  (at 1400 (not (going_on_City2_battle)))
  (at 1480 (going_on_City3_battle))
  (at 1520 (not (going_on_City3_battle)))
 )
 (:goal
  (and
   (married John)
   (>= (money John) 3000)
   (has_found_a_lot_of_money John)
   (taken_part_City1_battle John)
   (lieutenant John)
   (can_ride_a_horse John)
   (can_hunt John)
  )
 )
)
```

Fig. 4. The first generated problem.

```
(define (problem rpg-problem-test2)
 (:domain rpg-domain-test)
 (:objects James - man Kate - woman)
 (:init
  (= (number_thrown James) 20)
  (= (money James) 10)
  (= (battles_count James) 0)
  (has_to_find_a_lot_of_money James)
  (has_to_take_part_City1_battle James)
  (at_place City10 Kate)
  (at_place City5 James)
  (at 1280 (going_on_City1_battle))
  (at 1300 (not (going_on_City1_battle)))
  (at 1350 (going_on_City2_battle))
  (at 1400 (not (going_on_City2_battle)))
  (at 1480 (going_on_City3_battle))
  (at 1520 (not (going_on_City3_battle)))
 )
 (:goal
  (and
   (married James)
   (>= (money James) 3000)
   (has_found_a_lot_of_money James)
   (taken_part_City1_battle James)
   (lieutenant James)
   (has_found_a_quaterfoil James)
  )
 )
)
```

Fig. 5. The second generated problem.

forcing the planner to include these actions in the resulting plan. This makes the output plans more variable, which is the second part of Req. 4.

The generated problems can be supplied directly to the planner, together with the planning domain. This is exactly what we did, using SGPlan6. The resulting plans (i.e. the output of Step 3), presenting the tentative memory content subject to further changes in Step 5, are captured in Fig. 6, 7. On these figures, each line starts with the time when an

```
0.001: (FIND_A_LOT_OF_MONEY JOHN) [1.0000]
0.003: (STUDY_BASIC_MILITARY JOHN) [600.0000]
0.005: (LEARN_TO_RIDE_A_HORSE JOHN) [60.0000]
0.007: (MEET STACY JOHN VILLAGE) [1.0000]
0.009: (LEARN_TO_HUNT JOHN VILLAGE) [60.0000]
1.011: (DATE JOHN STACY VILLAGE) [200.0000]
201.013: (GET_MARRIED JOHN STACY VILLAGE) [1.0000]
202.015: (MOVE VILLAGE CITY5 JOHN) [3.0000]
600.017: (STUDY_ELITE_MILITARY_ACADEMY_IN_CITY5
    JOHN) [600.0000]
1200.019: (BE_ASCENDED_TO_LIEUTENANT JOHN) [1.0000]
1200.021: (MOVE CITY5 CITY1 JOHN) [3.0000]
1280.023: (TAKE_PART_CITY1_BATTLE JOHN) [20.0000]
```

Fig. 6. The first generated plan.

```
0.001: (FIND_A_LOT_OF_MONEY JAMES) [1.0000]
0.003: (FIND_A_QUATERFOIL JAMES) [0.0000]
0.005: (STUDY_BASIC_MILITARY JAMES) [600.0000]
600.007: (STUDY_MILITARY_ACADEMY_IN_CITY10 JAMES)
    [600.0000]
1200.009: (MOVE CITY10 CITY1 JAMES) [3.0000]
1280.011: (TAKE_PART_CITY1_BATTLE JAMES) [20.0000]
1300.013: (MOVE CITY1 CITY2 JAMES) [3.0000]
1350.015: (TAKE_PART_CITY2_BATTLE JAMES) [50.0000]
1400.017: (BE_ASCENDED_TO_LIEUTENANT JAMES) [1.0000]
1400.019: (MOVE CITY2 CITY5 JAMES) [3.0000]
1403.021: (MEET KATE JAMES CITY5) [1.0000]
1404.023: (DATE JAMES KATE CITY5) [200.0000]
1604.025: (GET_MARRIED JAMES KATE CITY5) [1.0000]
```

Fig. 7. The second generated plan.

action started (days after beginning), followed by the action in question and then by its duration.

The generated plans respect all the requirements asked by the designer and also satisfy all logical preconditions of the actions. As we did not include many timed initial literals (Req. 4), the computation was quick; it took less than a second.

In our experiments, we have trialed with several variants of the soldier domain, and generated more memory contents similar to those from Fig. 6 and 7. Importantly, we have also tried generating memories of a few interconnected agents, in particular two couples connected by an extramarital affair. In the case of so small amount of agents, the computation was also fast, about one second.

Discussion

We have proposed a design method for off-line generation of content of episodic memory of virtual agents using planning. We have developed a proof-of-concept implementation of its central, planning part and managed to generate memories of a single virtual agent using this method. We have also showed that the method can be used to generate memories of several similar agents from the same designer's input, introducing randomness. The method is suitable for generating history of a few interconnected agents as well. However, we are dealing with several issues, which will be discussed now.

Conceptual Issues

1. Scalability. The generation of the plans described in the previous section using SGPlan6 takes about a second. However, for practical use the domain will have to be extended. We do not see it to be a problem for a designer to enlarge the domain; however, the speed of generation may increase too much and become unacceptable. As we want to generate the memories off-line, minutes or even hours of generation are tolerable, but months or years not, obviously. On one hand, in comparison with IPC domains and problems, our domain contains more operators and predicates and also requires more PDDL features than the majority of IPC domains. On the other hand, our problems are not so complicated. Usually, just a few actions are needed to achieve a particular goal and searched plans are shorter than plans in a typical IPC task. This brings the possibility that after enlarging the domain, the speed of computation will be still reasonable. In our opinion, the more serious problem is inter-agent consistence, which will be discussed now.

2. Inter-agent consistence. We are able to generate history of a single agent, as well as histories of a few interconnected agents. However, generating contents of episodic memory of tens or hundreds agents with relationships among them would lead to a very complex planning problem and we will surely encounter performance issues. The remedy to this problem may be to generate memory contents for just small groups of agents using the proposed method and develop a mechanism for correcting inconsistencies between them. This could be achieved by incorporating methods used for solving constraint satisfaction problems. Treating this problem will be one of the next steps in our work.

3. Detailed actions. The content of episodic memory generated by the method contains just brief information about the actions which happened – the name of the action, when it took place and which people, locations etc. were involved. For example, John from the previous section knows that he married Stacy in his village on the 201st day of the simulation, but nothing more. The designer may want to equip her NPCs with more details about the actions, e.g. a detailed description of the mentioned wedding ceremony. In this case, she has to use some suitable mechanisms to accomplish this in the Step 5 of the workflow. One possible approach is to generate details of memories using the hierarchical approach described in (Brom, Pešková, and Lukavský 2007).

4. Categories of actions. It would be convenient to provide the designer with support for easy management of possible actions of agents. For instance, it should be possible to organize the high-level action definitions in packages (e.g. "relationships", "studies", "soldier", "mage").

5. High-level language. At the moment, we are focusing on the planning part of our proposed method, as it is arguably the most important one. So we have not formalized so far the high-level language used by a designer to describe the world and requirements on an agent. This issue also has to be addressed to enable implementation of the method as a whole.

6. Virtual world description. The virtual world used actually for our tests is just a list of places. Later, it could be useful to create a topology of the world. This should be possible conceptually, however, we have not tackled it yet.

Technical Issues

1. Few planners. In this moment, there are not many planners supporting all the features of PDDL we need.

That limits our possibilities to test. We are considering addressing this issue by developing our special-purpose planner, adapted to the general properties of generated domains and problems. This would also enable implementation of some domain-specific heuristics.

2. Timed initial literals. The incorporation of more timed initial literals currently delays the computation a lot. As they are essential for our planning problems, we have to solve this issue in some way. Currently, we are discussing it with the authors of SGPlan6. It is possible that this issue is specific only to this planner; however, it could be an integral property of our planning domains and problems. In that case, we would be obliged to implement a special-purpose planner.

Conclusion

We proposed a complex method for automatic generation of episodic memory for virtual agents, intended to be used for instance by game designers. When finished, it will enable the designer to specify her requirements on the history of the agent being created. It will also ensure that generated content of episodic memory will comply with predefined logical constraints of the events. Besides, some level of randomness will be included, to create "more animated" agents and also to enable creation of more agents from just one setting. After the automatic generation of memories of the agents, the designer will be enabled to perform her adjustments to create the final version of these memories.

As the first step, we have developed a prototype implementation of key parts of the method, to verify its feasibility.

Results we gained so far show that even with relatively simple testing domain, the problems created using this method enable the planner to generate brief content of episodic memory of an agent or even several agents with relationships between them. These results suggest that the picked approach to the problem is promising.

Our work in progress is to test the method on more and larger domains and problems. In future, we also want to implement its remaining parts. We are currently having some conceptual and technical issues; nevertheless, we think that the method in general will enable fast enough automatic generation of widely parameterized history of virtual agents.

Acknowledgements

This work was partially supported by the research project MSM0021620838 of the Ministry of Education of the Czech Republic, by the student research grants 201/09/H057 (GA ČR) and GA UK 21809, and by project P103/10/1287 (GA ČR).

References

Brom, C.; Pešková, K.; and Lukavský, J. 2007. What does your actor remember. Towards characters with a full episodic memory. In *Proc. of 4th ICVS*, LNCS 4871. 89-101. Berlin, Springer-Verlag.

Brom, C.; and Lukavský, J. 2009. Towards Virtual Characters with a Full Episodic Memory II: The Episodic Memory Strikes Back. In *Proc. Empathic Agents* 1-9. AAMAS workshop.

Castellano, G.; Aylett, R.; Dautenhahn, K.; Paiva, A.; McOwan, P. W.; and Ho, S. 2008. Long-term affect sensitive and socially interactive companions. In *4th Int. Workshop on Human-Computer Conversation*.

Dias, J.; Ho, W.C.; Vogt, T.; Beeckman N.; Paiva, A.; and Andre, E. 2007. I Know What I Did Last Summer: Autobiographic Memory in Synthetic Characters. In *Proc. of ACII* 606-617. Springer-Verlag.

Doherty, D.; and O'Riordan, C. 2008. Toward More Humanlike NPCs for First-/Third-Person Shooter Games. In *AI Game Programming Wisdom IV* 499-512. Charles River Media.

Gerevini, A.; and Long, D. 2005. Plan Constraints and Preferences in PDDL3, Technical Report, RT 2005-08-47, Dept. of Electronics for Automation, University of Brescia, Italy.

Ho, W. C.; and Watson, S. 2006. Autobiographic knowledge for believable virtual characters. In *Proc. of Intelligent Virtual Agents*, LNCS 4133, 383-394. Springer-Verlag.

Hsu, C.-W.; and Wah, B. W. 2008. The SGPlan planning system in IPC6. In *6th International Planning Competition Booklet* (ICAPS-08).

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL – The Planning Domain Definition Language – Version 1.2, Techical Report, CVC TR-98-003, Yale Center for Computational Vision and Control.

Nuxoll, A. 2007. Enhancing Intelligent Agents with Episodic Memory. Ph.D. thesis, The University of Michigan.

Porteous, J.; and Cavazza, M. 2009. Controlling Narrative Generation with Planning Trajectories: The Role of Constraints. In *Proc. Of 2nd Int. Conf. on Interactive Digital Storytelling*. LNCS 5915, 280-291, Springer.

Rickel, J.; and Johnson, W. L. 1999. Animated Agents for Procedural Training in Virtual Reality: Perception, Cognition, and Motor Control. *App. Artificial Intelligence* 13(4-5): 343-382.

Riedl, M. O.; and Young, R. M. 2006. Story Planning as Exploratory Creativity: Techniques for Expanding the Narrative Search Space. *New Generation Computing* 24(3): 303-323.

William, H.; and Manier, D. 1995. Remembering as communication: A family recounts its past. In *Remembering Our Past: Studies in Autobiographical Memory* 271-290. Cambridge University Press.

Williams, H.L.; Conway, M.A.,; and Cohen, G. 2008. Autobiographical memory. In *Memory in the real world* 21-90. Psychology Press.