Story Generation in PDDL Using Character Moods: A Case Study on Iliad's First Book

Andrea Marrella and Stavros Vassos

Sapienza University of Rome, Italy
{marrella,vassos}@dis.uniroma1.it

Abstract. In this paper we look into a simple approach for generating character-based stories using planning and the language of PDDL. A story often involves modalities over properties and objects, such as what the characters believe, desire, request, etc. We look into a practical approach that reifies such modalities into normal objects of the planning domain, and relies on a "mood" predicate to represent the disposition of characters based on these objects. A short story is then generated by specifying a goal for the planning problem expressed in terms of the moods of the characters of the story. As a case study of how such a domain for story generation is modeled, we investigate the story of the first book of Homer's Iliad as a solution of an appropriate PDDL domain and problem description.

1 Introduction

In this work we present some preliminary results toward modeling stories as solutions of automated planning problems [6]. Planning systems are problem-solving algorithms that operate on explicit representations of states and actions [3]. We focus on the standardized Planning Domain Definition Language (PDDL) [5]; it allows one to formulate a *planning problem* $\mathcal{P} = \langle I, G, \mathcal{P}_D \rangle$, where I is the initial state, G is the goal state, and $\mathcal{P}_D$ is the planning domain. In turn, a planning domain $\mathcal{P}_D$ is built from a set of *propositions* describing the state of the world (a state is characterized by the set of propositions that are true) and a set of *operators* (i.e., actions) that can be executed in the domain. Each operator is of the form $o = \langle Pre_o, Eff_o \rangle$, where Pre_o and Eff_o specifies the preconditions and effects of o, in terms of the set of domain propositions. A solution for a PDDL planning problem is a sequence of operators—a *plan*—whose execution transforms the initial state I into a state satisfying the goal G. In this paper we focus on the STRIPS subset of PDDL 2.1 [2], which also makes use of the common features of typing and (possibly quantified) conditional effects.

By representing the underlying dynamics of the world in which a story unfolds as a planning domain and the characteristics of the specific target story instance that we intend to generate as a planning problem, then we can use an off-the-shelf planner to generate a story according to the given description. One motivation for this approach is that a wide range of story alternatives can be handled by a concise representation, while these stories can be adapted and tweaked both at design time and real-time in order to satisfy requirements that arise during the execution, e.g., in a video game context.

As a case study, we investigate the knowledge representation and modeling task of generating the story of the first book of Homer's classic work Iliad. Figure 1 shows a

- *Apollo's priest Chryses comes to the Achaian camp and asks to ransom back his daughter Chryseis, who has been captured by Agamemnon.*
- *Chryses, is insulted and sent away. He prays to Apollo to punish the Greeks, which Apollo does by sending a plague upon them.*
- *Achilleus calls an assembly to deal with the plague, and the prophet, Kalchas, reveals that Apollo was angered by Agamemnon's refusal to return the daughter of his priest.*
- *Agamemnon agrees to give her back, but demands compensation. This provokes Achilleus' anger, and, after they exchange threats and angry words, Agamemnon decides to take Achilleus' "prize", the captive woman, Briseis.*
- *The goddess, Athene, prevents Achilleus from killing Agamemnon by promising that he will one day be compensated with three times as many prizes.*
- *Agamemnon's men take Briseis from Achilleus, and Achilleus prays to his divine mother, Thetis, for help. He says he will not fight, and he asks her to persuade Zeus to make the battle go badly for the Greeks so they will see that they should not have dishonored him.*
- *Odysseus leads a group of Greeks to Chryse to return Chryseis to Chryses. Meanwhile, Achilleus isolates himself from the other Greeks.*
- *Thetis, begs Zeus to honor her son, Achilleus, by turning the battle against the Greeks so they will see that they need him.*

Fig. 1. An abstract of the first book of Iliad.

high-level summary of its content. We will focus on modeling the underlying facts and dynamics so that these events are generated as an action sequence that solves a planning problem expressed in PDDL. As we will see next, the emphasis is on the disposition that characters have with respect to each other and how this drives their actions.

2 Modeling the first book of Iliad in PDDL

One immediate observation is that PDDL focuses on an action-centered representation that models single aspects of the world, namely which properties are *true* at a given moment, while stories such as the war described in Iliad involve many *modalities* over these properties. In other words, while PDDL is typically used to model facts like:

"Agamemnon is at the Greek camp," "Chryseis is kept captive by Agamemnon,"

in the narrative of Iliad perhaps the statements most crucial to the plot are those about what the heros of the story *believe, desire, request*, for example:

"Chryses desires that Chryseis is not kept captive by Agamemnon."

There is a vast amount of work in the literature for representing and reasoning with such modalities, but our intention here is to investigate a practical way to do so while remaining in the simple representation of the STRIPS subset of PDDL. The statements

```
(:action interact-mood
 :parameters (?c1 - character ?c2 - character ?t - types_of_interaction
              ?c3 - character ?loc - location)
 :precondition (at ?c1 ?loc)
 :effect (and
  (when (and (mood ?c2 ?c1 neutral) (= ?t request_release) (related ?c1 ?c3)
             (captured ?c3 ?c2) (at ?c2 ?loc) (at ?c3 ?loc))
        (and (mood ?c1 ?c2 request_release) (mood ?c2 ?c1 bad)))
  (when (and (mood ?c1 ?c2 bad) (mood ?c2 ?c1 request_release) (= ?t refuse_release)
             (captured ?c3 ?c1) (at ?c2 ?loc) (at ?c3 ?loc))
        (mood ?c2 ?c1 bad))
  ... ))
```

Fig. 2. Part of the specification of the action capturing mood interaction between characters.

above can of course be directly represented by means of normal predicates, as for instance, having both a normal *HeldCaptive*(x, y) predicate and a pair of predicates capturing the desire modality of a person p in the story: *DesiresHeldCaptive*(p, x, y) and *DesiresNotHeldCaptive*(p, x, y).[1] It becomes then interesting to identify some common patterns and a representation methodology so that this type of information can be specified in a way that is both intuitive and concise, and can be effectively maintained and extended in order to capture the intended stories that we would like to generate.

In this work we look into an approach that *reifies* such modalities into *objects* in the domain, and relies on a predicate to represent the disposition or *mood* of characters toward the each other with respect to an object reified in this way. We call this predicate `mood` and, for example, the following PDDL literal can be used to model that Agamemnon has a negative disposition toward Achilleus:

`(mood agamemnon achilleus bad)`,

while the following can be used to model that the mood of Agamemnon toward Chryses is particularly related to a former request that Agamemnon has accepted:

`(mood agamemnon chryses accept_release)`.

A number of objects similar to `accept_release` are used to model the (abstracted and overloaded) moods of characters in the first book of Iliad as we will see next, such as `request_release`, `refuse_release`, and `desire_capture`. These reified objects are just syntactic terms that need to be handled appropriately in the actions that are available in the domain in order to make sense and generate the intended outcomes. Moreover, it becomes important that as many as possible of such objects can be used to identify general interactions that may happen in the story, so that we get a concise representation that can be easily adapted and maintained.

The idea then about available actions is that they fall in two categories: i) *physical* actions that modify only physical properties of the world, e.g., the typical move action, and ii) *interaction* actions that modify the mood disposition between two characters with respect to a context and possibly with another character, e.g., the action of asking to release a prisoner which may change the mood between the persons involved.

[1] Note that $\neg$*DesiresHeldCaptive*(p, x, y) is different than *DesiresNotHeldCaptive*(p, x, y).

For the first type of actions, the PDDL modeling includes an abstraction of the physical activities that are described in the summary of Figure 1 in terms of appropriate action schemas for `go`, `capture`, `release`, `punish` (by means of a disaster), and `cease_disaster`. As it is easy to observe in the summary of Figure 1, these actions are a small part of what is really happening in the story. A lot more has to do about who requests what, and how this affects the relationships of the heros and the gods in the story. This form of interaction is captured by means of an `interact-mood` action that operates on the level of the mood disposition between characters.

Part of the action schema for the action `interact-mood` is shown in Figure 2. This action takes 5 arguments as follows: i) the character which initiates the interaction, ii) a second character that participates in the interaction, iii) the type of interaction in terms of a mood or a reified modality, iv) a character which is the object of the interaction, and v) the location where the interaction takes place (there is also action `interact-self` that does not include an object for the interaction as an argument).

In the effects of `interact-mood`, the possible cases for interactions are modeled which depend mostly on the moods of the participating characters and result in updating them accordingly. The cases are modeled based on PDDL conditional effects; here the first case models that when a character `c1` requests from `c2` the release of character `c3` and some conditions hold about the moods between the characters, then the mood of `c2` about `c1` becomes `request_release` and `c1` become angry with `c2`. In the second case, something similar is modeled about refusing to release interaction. A number of cases like this can be specified, and one can think of a table that depending on the moods between `c1` and `c2` and the type of interaction, a different outcome takes place.

Action instances of `interact-mood` may be driven by physical properties of the world and the mood disposition between characters, but the important thing to note though is that we require they *always modify only the mood disposition between characters participating to the interaction*. This provides a layer for representing the gist of the story as a game between the relationships of characters, while physical actions are just ways to materialize the tensions expressed in the moods.

The goal of the planning problem is then expressed also solely in terms of the moods of the characters, which in the case of the first book of Iliad could be the following:

```
(and (mood Achilleus Agamemnon bad) (mood Zeus Agamemnon accept_punish)
     (mood Chryses Agamemnon good))
```

I.e., at the end of the first book Achilleus is upset with Agamemnon, Zeus is in the mood of punishing Agamemnon, and Chryses is happy with the former captor of his daughter.

Figure 3 shows a plan that is obtained using version 6 of SGPlan [1].[2] Note that variants of the actual story can be obtained by planning for different goals, tweaking the initial state, or altering the action schemas.

3 Discussion

There is a lot of related work in the literature in employing planning for generating narrative or interactive stories, also taking advantage of the increasing expressiveness

[2] The full PDDL specification and the required instructions to run SGPlan planner can be found at `https://code.google.com/p/the-iliad-project/`

```
(GO CHRYSES APOLLO_TEMPLE GREEKS_CAMP)
(INTERACT-MOOD CHRYSES AGAMEMNON REQUEST_RELEASE CHRYSEIS GREEKS_CAMP)
(INTERACT-MOOD AGAMEMNON CHRYSES REFUSE_RELEASE CHRYSEIS GREEKS_CAMP)
(GO CHRYSES GREEKS_CAMP APOLLO_TEMPLE)
(INTERACT-MOOD CHRYSES APOLLO REQUEST_PUNISH AGAMEMNON APOLLO_TEMPLE)
(INTERACT-SELF APOLLO ACCEPT_PUNISH AGAMEMNON APOLLO_TEMPLE)
(PUNISH APOLLO GREEKS_CAMP PLAGUE)
(INTERACT-MOOD KALCHAS AGAMEMNON REQUEST_RELEASE CHRYSEIS GREEKS_CAMP)
(INTERACT-SELF AGAMEMNON ACCEPT_RELEASE CHRYSEIS GREEKS_CAMP)
(RELEASE AGAMEMNON CHRYSEIS GREEKS_CAMP)
(INTERACT-MOOD AGAMEMNON ACHILLEUS DESIRE_CAPTURE BRISEIS GREEKS_CAMP)
(INTERACT-SELF ACHILLEUS ATTEMPT_TO_KILL AGAMEMNON GREEKS_CAMP)
(INTERACT-MOOD ATHENE ACHILLEUS BLOCK_THE_ATTEMPT_TO_KILL AGAMEMNON GREEKS_CAMP)
(GO ODYSSEUS GREEKS_CAMP APOLLO_TEMPLE)
(GO CHRYSEIS GREEKS_CAMP APOLLO_TEMPLE)
(INTERACT-MOOD ODYSSEUS CHRYSEIS MEET CHRYSES APOLLO_TEMPLE)
(INTERACT-MOOD CHRYSES ODYSSEUS DESIRE_CEASE_DISASTER APOLLO APOLLO_TEMPLE)
(CEASE_DISASTER APOLLO GREEKS_CAMP PLAGUE)
(INTERACT-MOOD ACHILLEUS THETIS REQUEST_PUNISH AGAMEMNON GREEKS_CAMP)
(INTERACT-MOOD THETIS ZEUS REQUEST_PUNISH AGAMEMNON GREEKS_CAMP)
(INTERACT-SELF ZEUS ACCEPT_PUNISH AGAMEMNON OLYMPUS)
```

Fig. 3. The planning solution for the first book of Iliad.

of the evolving versions of PDDL, e.g., [7] that makes use of constraints to specify requirements on the sequencing of events. There are only a few approaches that look into the modalities we discussed. In particular, Riedl and Young [8] look into the case of representing the *intentions* of characters and finding stories that are believable in that characters behave intentionally. They introduce a planning formalism that goes beyond PDDL and demonstrate its power with a specialized planning system. Haslum [4] later showed that this type of representation can in fact be compiled into PDDL.

In this work we decided to take a practical approach that allows to represent any type of modality over facts and events, but leaving the semantics of these events to be also specified by means of preconditions and effects given by the person modeling the planning domain. For example, when the object `desire_capture` is used in the planning domain description, there is no guarantee by the framework that it relates to the predicate `captured` in any way or that a character will act based on this desire. Nonetheless, all objects of this sort get their intended meaning by means of the `mood` predicate that specifies how they change by the available interactions, while they also trigger physical actions by being used in their preconditions.

So, in a sense our approach is not really an approach to knowledge representation and reasoning, rather than a simple methodology for expressing the intended meaning of modalities over facts using a separate “moods” layer. This indeed has many drawbacks, the most obvious being that there is no way to verify that the modalities actually get their intended meaning, as this completely relies on the PDDL programmer. Nonetheless, as it is often the case in game developing, simple programming methodologies can go a long way compared to knowledge representation systems, and one reason for this is that the former are easier to understand and control. So, in a sense our approach can be seen as a kind of programming methodology for PDDL programs for generating stories based on two layers: a moods layer and a physical layer.

4 Conclusions

In this work we present a simple idea for modeling stories via planning, and show how this could works for the case of the first book of Iliad, the classic work of Homer. We look into a simplistic approach that reifies modalities over facts (e.g., related to desires and requests) into objects in the domain, and relies on a "mood" predicate to represent the disposition of characters toward the each other and with respect to such objects. Except for physical actions, we introduce a pair of and particular mood interaction actions which affect only the mood of characters. The goal of the planning problem can then be expressed in terms of the moods of the characters of the stories, which in order to be satisfied will drive physical action. Our experimentation with modeling the first book following these ideas showed that it is intuitive and practical, but may also lead to similar stories than the one that is intended. We do not consider this to be a major problem in the sense that these stories are still reasonable and in general would be perhaps welcome as possible alternative outcomes in a video game scenario (unlike here where we were interested in modeling a very specific sequence of events). Finally, we note that this approach is not intended to be thought of as a proper framework for representing modalities, rather than a practical approach that would be intuitive to the story developers as a kind of programming methodology. Finally, we note that this work is inspired by the PDDL modeling challenge in the 6th edition of the Intelligent Narrative Technologies Workshop.

References

1. Chen, Y., Wah, B.W., Hsu, C.w.: Temporal planning using subgoal partitioning and resolution in SGPlan. J. of Artificial Intelligence Research 26 (2006)
2. Fox, M., Long, D.: Pddl2.1: An extension to pddl for expressing temporal planning domains. J. Artif. Intell. Res. (JAIR) 20, 61–124 (2003)
3. Geffner, H., Bonet, B.: A Concise Introduction to Models and Methods for Automated Planning. Morgan & Claypool Publishers (2013)
4. Haslum, P.: Narrative planning: Compilations to classical planning. J. Artif. Int. Res. 44(1), 383–395 (May 2012)
5. Mcdermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D.: PDDL - The Planning Domain Definition Language. Tech. rep., CVC TR-98-003, Yale Center for Computational Vision and Control (1998)
6. Nau, D., Ghallab, M., Traverso, P.: Automated Planning: Theory & Practice. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2004)
7. Porteous, J., Cavazza, M.: Controlling narrative generation with planning trajectories: The role of constraints. In: Proceedings of the 2nd Joint International Conference on Interactive Digital Storytelling: Interactive Storytelling. ICIDS '09, vol. 5915, pp. 234–245. Springer-Verlag, Berlin, Heidelberg (2009)
8. Riedl, M.O., Young, R.M.: Narrative planning: balancing plot and character. J. Artif. Int. Res. 39(1), 217–268 (Sep 2010), `http://www.cc.gatech.edu/~riedl/pubs/jair.pdf`