Open World Story Generation for Increased Expressive Range

David Thue, Stephan Schiffel, Tryggvi Þór Guðmundsson,
Guðni Fannar Kristjánsson, Kári Eiríksson, and Magnús Vilhelm Björnsson

School of Computer Science, Reykjavik University
Menntavegur 1, Reykjavik, 101, Iceland
{davidthue,stephans,tryggvi15,gudni14,kari14,magnusvb14}@ru.is

Abstract. To let authors shape the set of experiences that can occur when a generative Interactive Storytelling (IS) system is used, the process of authoring for the system must support specifying constraints over how different stories can progress. We present an extension to an existing IS system that both allows authors more flexibility in specifying the constraints and gives the generator more freedom in filling in the parts of the story that the authors leave unconstrained. Our approach is based on open-world planning using the IndiGolog action programming language and heuristic search for plan generation.

1 Introduction

One of the primary goals of Interactive Storytelling (IS) is to create playable experiences that are both narratively rich and richly interactive. Given this ambition, it is essential to support a wide range of expression for the designers and authors of such experiences, since the systems and content they create will ultimately define the set of experiences that can occur [1]. To let authors shape the set of experiences that can occur when a generative IS system is used, the process of authoring for the system must support specifying constraints over how different stories can progress [1–4]. The way in which a generative IS system represents its constraints can create conflicts between expression and implementation. For example, while authors often give expressive freedom to the generator by *omission* (i.e., letting it determine any unconstrained aspects of the story's state), leaving the story's state unconstrained is at odds with a common assumption of AI Planning (that all unstated facts are necessarily false). Supporting this expressive freedom thus limits the kinds of planners that can be used. Prior work on "open world" story generation allowed authors to leave some story world facts undetermined [5], but true and false facts could only be asserted about the initial state of the story world. In a story world driven by a simulation, the initial state might represent only a small fraction of the circumstances under which an author might want one of their stories to start.

In this paper, we describe new improvements to our previous work on this topic [6], in which authored constraints could be applied at any time in the

The final publication is available at Springer via https://doi.org/10.1007/978-3-319-71027-3_33

story world (unlike [5]), but were limited to expressing only true or undetermined values. To address this limitation, we changed the planner in our existing system to support "open" worlds and improved its performance using a heuristic.

Illustrative Example. To illustrate the problem, we present a small example that uses terminology from our prior work [6]. An *outline* is a set of constraints that restrict both how and when a story should start (*initial conditions*) as well as how it should ultimately end (*final goals*). Authors create outlines in terms of *abstract entities*. Our generative IS system determines a story's *concrete entities* (which exist in the simulated world) at runtime; each abstract entity is automatically *mapped* to a concrete entity that satisfies the former's constraints. We add the notion of *intermediate goals*, which are similar to *landmarks* [2].

Given this terminology, consider a world with 3 concrete characters (*Sarah, John, Sam*), one article (*Necklace*), and 2 relations between them: *Has(Sarah, Necklace), ParentOf(Sarah,John)*. We can define an outline in which a villain somehow obtains an heirloom from the mother of the hero, but eventually the mother gets the heirloom back. It has 3 abstract characters (*Villain, Hero, Mother*), one abstract article (*Heirloom*), initial constraints (*Has(Mother, Heirloom), ParentOf(Mother,Hero)*, and ¬*Has(Villain, Heirloom)*), an intermediate goal (*Has(Villain,Heirloom)*), and a final goal (*Has(Mother,Heirloom)*).

The stories that can be generated from this outline and world state depend on the actions and entities that are defined in the world. One example could be: *map(Hero,John)*, *map(Mother,Sarah)*, *map(Heirloom,Necklace)*, *map(Villain,Sam)*, *steal(Villain,Mother,Heirloom)*, *steal(Hero,Villain,Heirloom)*, *give(Hero,Mother,Heirloom)*. The *map* actions in this example associate abstract entities from the story outline with concrete entities in the world.

We can use this example world to demonstrate how the closed world assumption in our previous planner [6] fails to support negative conditions in story outlines. With the closed world assumption, anything not mentioned in the initial state of the planner is considered false. Therefore, the initial state of the planner in our example would be represented simply as *Has(Sarah, Necklace)*. However, this initial state is the same as if we did not have the initial condition ¬*Has(Villain, Heirloom)* at all. That is, it does not distinguish between the two story outlines where (i) the villain must not have the heirloom at the beginning and (ii) the author does not care. Depending on the concrete implementation of the planner, there are two possible failure cases:

1. If the planner simply ignored negative preconditions, it could return mappings that violate these preconditions, e.g., $Villain \mapsto Sarah$, $Heirloom \mapsto Necklace$, because the condition ¬*Has(Villain, Heirloom)* is never checked.
2. If the planner treated all absent conditions as negative, it would fail to return valid mappings. For example, it would never return $Hero \mapsto Sarah$, $Heirloom \mapsto Necklace$, because the $Has(Hero, Heirloom)$ is not an initial condition and thus must be false.

Since both cases are undesirable, we base our approach on IndiGolog [7], a planner that allows us to represent facts whose value is unknown.

2 Proposed Approach

Using IndiGolog, the initial state of our planning problem consists of all facts in the world as well as the initial conditions in the story outline, where each fact's value is explicitly represented. Thus, the initial conditions of our example in Section 1 become: *init(has(sam,necklace),true)*, *init(has(john,necklace),false)*, and *init(has(villain,heirloom),false)*. Our action definitions in IndiGolog are essentially unchanged from [6]; aside from the syntactic differences between IndiGolog and PDDL. The only addition is that negative preconditions are checked in the map actions. With these changes, our system achieves plan soundness and assignment soundness as defined in [6], while supporting more expressive story outlines by allowing negative initial conditions.

2.1 Planner Improvements and Heuristics

Our planner uses informed search with a heuristic function to estimate the cost of a plan. Common heuristics used in automated planning, such as delete relaxation heuristics [8], require perfect information and an explicit representation of the state (e.g., as a set of facts). However, due to the open-world assumption and the use of IndiGolog, we have neither. We implemented a number of simple heuristic features to improve the performance of the search by using two techniques for state space reduction, as we describe below.

State Space Reduction. The definition of our planning problem allows for plans that are equivalent in several ways. First, observe that once an abstract entity is mapped to a concrete one, actions that use the abstract or the concrete entity have the same effects due to the effect synchronizers defined in [6]. Second, actions involving different entities are independent of each other and can be reordered without changing the resulting state. For example, [*find(John, Necklace), find(Sarah, Ring)*] and [*find(Sarah,Ring), find(John,Necklace)*] result in the same state. To reduce the state space, we only consider actions with concrete entities and discard visited states whose partial plan only differs in terms of the order of mutually-independent actions.

Heuristic. Our heuristic estimates the cost of a plan as a linear combination of four features: the number of actions in the plan (L); the number of unfulfilled final goal conditions (F); the number of unfulfilled intermediate goal conditions (I); and the number of initial conditions that cannot be fulfilled with the mappings that are already in the plan (C).

We tested our heuristic features using six test worlds that we varied in terms of their numbers of concrete entities, abstract entities, initial conditions, intermediate goals, final goals, and steps for the shortest solution plan. All the features in the heuristic turned out to be necessary for finding plans for all six worlds in a reasonable amount of time (10s). With the best weights that we found for the four features ($L = 1$, $F = 2$, $I = 3$, $C = 1$), the solution times for all six worlds

ranged from 5ms to 123ms – nearly two orders of magnitude faster than when the features were ignored. More detailed data from our experiments has been reserved for the poster that will accompany this paper.

3 Discussion and Future Work

Our solution extends prior work to allow positive, negative, and undetermined conditions to be authored inside flexible story outlines, supporting the open world planning of interactive stories. We also introduced extensible heuristics for our new story planner, which achieve good performance for story generation tasks. Although the resulting computing times seem reasonable, there are edge cases of story outlines which cannot be completed in a reasonable amount of time. Specifically, the intermediate goal heuristic (I) can cause the planner to prefer mapping multiple abstract entities to the same concrete entity (when it simplifies meeting certain intermediate goals), but the resulting search path may never lead to a desired goal state in which those abstract entities satisfy mutually exclusive roles. In the future, these heuristics can potentially be modified to steer story generation towards stories of higher quality, such as those that are more interesting to each individual player.

References

1. Riedl, M.O., Young, R.M.: From linear story generation to branching story graphs. IEEE Computer Graphics Applications 26(3), 23–31 (2006)
2. Porteous, J., Cavazza, M.: Controlling narrative generation with planning trajectories: The role of constraints. In: Iurgel, I., Zagalo, N., Petta, P. (eds.) ICIDS 2009, LNCS, vol. 5915, pp. 234–245. Springer Berlin Heidelberg (2009)
3. Thue, D.: Generalized Experience Management. Ph.D. thesis, Department of Computing Science, University of Alberta, Canada (2015)
4. Weyhrauch, P.: Guiding Interactive Drama. Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA (1997)
5. Riedl, M.O., Young, R.M.: Open-world planning for story generation. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. pp. 1719–1720. IJCAI'05, Morgan Kaufmann Publishers Inc. (2005)
6. Thue, D., Schiffel, S., Árnason, R.A., Stefnisson, I.S., Steinarsson, B.: Delayed roles with authorable continuity in plan-based interactive storytelling. In: Nack, F., Gordon, A.S. (eds.) ICIDS 2016. LNCS, vol. 10045, pp. 258–269. Springer, Heidelberg (2016).
7. De Giacomo, G., Lespérance, Y., Levesque, H.J., Sardina, S.: IndiGolog: A High-Level Programming Language for Embedded Reasoning Agents, In El Fallah Seghrouchni, A., Dix, J., Dastani, M., and Bordini, R.H. (eds), Multi-Agent Programming: Languages, Tools and Applications. pp. 31–72. Springer US (2009)
8. Hoffmann, J., Nebel, B.: The ff planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research 14, 253–302 (2001)