HeadSpace: Incorporating Action Failure and Character Beliefs into Narrative Planning

Rushit Sanghrajka 1, R. Michael Young 1,2, Brandon Thorne 3

1 School of Computing, University of Utah
2 Entertainment Arts and Engineering Program, University of Utah
3 Spectra Logic
Salt Lake City, UT, USA
rush.sanghrajka@utah.edu, young@eae.utah.edu

Abstract

Previous work on story planning has lacked a knowledge representation for characters that attempt actions that fail because of the characters’ misconceptions about the world state. This work describes HeadSpace, a state-space heuristic search planning system that generates stories that track and manipulate characters’ beliefs about the story world. The planner produces story plans with actions that are attempted but fail. We show an example story plan that contains failed-action content that cannot be generated by typical planning-based approaches to story creation, and we provide an analytical evaluation that characterizes our planner’s increased expressive range relative to other narrative planners addressing character belief and/or failed action execution.

Introduction

In stories, characters commonly attempt to perform actions that fail (Lenhart et al. 2008). For example, when Thanos the Mad Titan attempts to remove half of the life in the universe by snapping his fingers in Avengers: Endgame (Russo, A. and Russo, J. 2019), he’s surprised when his finger snap has no effect. He realizes too late that the Infinity Stones, which he had assumed were in place along the back of the gauntlet he’s wearing, were missing, removing the gauntlet’s powers.

When authors include actions that fail in their stories, it is not simply due to emergent properties of complex story worlds. Quite often, characters’ action failures are designed intentionally by authors for narrative effect. They build tension, prolong efforts around goal achievement, or highlight the disparities between the knowledge states of a story’s characters. These functions played by action failure are central to many plot-level narrative constructs. The work we describe here seeks to outline a principled means to generate story lines with failed actions and advance a broader goal of automatically creating more expressive, natural, and compelling narratives.

One of the strengths of recent planning-based narrative generation methods (e.g. (Young et al. 2013; Porteous and Cavazza 2009; Coman and Munoz-Avila 2012; Wane et al. 2014; Bahamón, Barot, and Young 2015; Teutenberg and Porteous 2013)) is that they retain many of the benefits of classical planning while also increasing the expressive range (Smith and Whitehead 2010) of narrative generators. One limitation of planning approaches for story line creation arises from their inability to generate plans containing actions that fail. In this paper, we extend previous preliminary work (Thorne and Young 2017) to provide the design of an algorithm for story generation that explicitly plans for character actions that fail. The algorithm uses a knowledge representation that provides context for this failure based on the limitations of characters’ beliefs about the story world around them (e.g., Thanos’ false belief that the all the Infinity Stones were in place in his gauntlet and he had the resulting power to change the universe). The algorithm, called HeadSpace, produces story structure that has many of the advantageous properties found in other planning-based approaches and is more parsimonious than previous approaches to story generation that also address character belief dynamics.

As we describe below, the HeadSpace narrative planning algorithm can generate stories where a) agents may operate under mistaken beliefs that lead them to attempt actions which fail, and these attempted actions do not produce the expected effects; b) agents performing actions observe the success or failure of their actions’ execution; c) agents revise their belief states in response to an observed failure as well as both passive and active sensing actions.

Related Work

Narrative planning research has incorporated additional constructs into the planning process to expand the expressive range of plan representations to support aspects of character decision-making. IPOCL (Riedl and Young 2010) adds intentional structures to plan requirements that ensure actions taken by characters appear to be coherent to readers. Extending IPOCL, Ware and Young’s (Ware and Young 2011) CPOCL adds a representation of conflict in story plans where one character might take actions that interfere with or thwart the execution of actions taken by other characters. In the Mask planning system, Bahamón and his collaborators (Bahamón, Barot, and Young 2015; Bahamón and Young 2017) incorporate a model of character personality that is used to drive character choice for action that expresses character personality traits.

These extensions to the classical planning approach have assumed two things. First, they make no distinction between
Full text of the paper: The knowledge held by characters and that held by the planning system. Second, the algorithms are based on a long tradition of planning research outside of narrative planning where the soundness of planning algorithms is a requirement. This is a highly desirable property when producing plans for robot execution, for example, running robots executing tasks on a factory floor. The focus of classical planning approaches on the representation of physical properties of the world and on requirements that any plan produced by a planner must be sound limits the production of plans to drive characters bumbling through a story world. Our work aims to distinguish between soundness on a material level (which classical planning focuses on) and on an epistemic level (where agents performing the actions believe all of the preconditions of those actions obtain at the point where the agent will execute them). As we describe below, HEADSpace-produced plans have steps having epistemically satisfied preconditions but not necessarily materially satisfied ones. Plans consisting of failed actions still have soundness properties relative to character beliefs, because characters must believe all preconditions of an action obtain before attempting it.

The initial work addressing disparities of knowledge between agents in a planning context was done by Pollack in her work on the Spirit plan inference system (Pollack 1986). This work in turn motivated Geib’s (Geib 1994) approach to formalizing intention in a plan generation system. As part of the resulting ItPlanS planner, Geib and Webber (Geib and Webber 1993) draw the distinction between an action’s preconditions and other conditions that are necessary for an action’s execution (but are not established by the planner should they not hold). Geib also considers the importance of reasoning about action failure in the context of plan generation. Cavazza and his collaborators (Cavazza, Charles, and Mead 2003) describe an approach to the generation of story sequences where characters are unaware of some aspects of the world around them, including the harmful consequences of some of their own actions. Shirvani and their collaborators (Shirvani, Ware, and Farrell 2017) propose an extension to state space planning models that represents character beliefs as well. Their approach also tracks character beliefs, only with deeper layers of nested beliefs. However, their approach doesn’t leverage these beliefs, as HEADSpace does, to extend the space of narrative plans that can be produced to include ones where characters perform actions that can fail due to their flawed beliefs about the world. Parallelizing earlier work by Haslum (Haslum 2012) on the pre-compilation of intentional narrative planning into more efficient conventional planning representations, recent work by Christensen and their collaborators (Christensen, Nelson, and Cardona-Rivera 2020) developed processes that pre-compile belief-based planning models into similarly efficient representations.

The IMPRACTical planner (Teutenberg and Porteous 2013) produces story plans guided by a heuristic that incorporates individual characters’ intentions. Furthermore, extension to this work (Teutenberg and Porteous 2015) allows for separate belief models for each agent. Using a combination of observation axioms and operator annotations, their system can create disparities between the belief models of the agents and the world state. Plan generation is directed based on actions supported by beliefs of the enacting agent. This enables deceptive social action involving manipulation of the belief state of one agent by actions of another. Subsequently the manipulated agent can be induced to act against its own interests because of the incorrect beliefs it holds.

**Representation**

HEADSpace uses a PDDL-like (McDermott et al. 1998) syntax for representing schematized action types in which actions are characterized in terms of preconditions – conditions that must obtain in the world state in order for the action to execute – and effects – conditions in the world state that change upon the action’s successful execution. For efficiency, we follow the approach of Nebel and Hoffmann (2001) and others and pre-compile schematized operators for a given domain into a set of ground operators representing every valid ground instantiation of a domain’s act-types. We further differentiate the knowledge representation by describing both preconditions and effects related to the physical world and others that obtain in the beliefs of the character performing the action. In HEADSpace, a world frame captures the sets of ground literals that can be used to characterize the world, as well as the set of symbols used to name the characters capable of taking action in the world.

In the HEADSpace knowledge representation, the set O contains all the object constants for a given domain. There is a distinguished type of object symbols called character, character symbols are contained in a set C where C ⊆ O. Characters are distinguished from other objects by their ability to take action.

In HEADSpace, a world frame captures the sets of ground literals that can be used to characterize the world, as well as the set of symbols used to name the characters capable of taking action in the world.

**Definition (World Frame).** A world frame is a tuple \(W = (GL, C)\) where GL is a set of positive ground literals and C is a set of constants, each denoting a unique character. C contains one distinguished character name E, which designates the environment.

A belief state characterizes the ground literals that a character believes to be true and false, as well as those whose truth values that are unknown to the character.

**Definition (Belief State).** Given a world frame \(W = (GL, C)\), a belief state for some character \(c \in C\) is a tuple \(BS_c = (B^+_c, B^-_c, U_c)\) such that \(B^+_c, B^-_c\) and \(U_c\) together form a partition of GL, where \(B^+_c\) designates all the ground literals that \(c\) believes to be true, \(B^-_c\) includes all the ground literals that \(c\) believes to be false and \(U_c\) designates all the ground literals that \(c\) does not believe to be true and does not believe to be false.

A world state assigns truth values to every ground literal in a world frame, and also provides belief state specifications for every character in a world frame.

**Definition (World State).** Given a world frame \(W = (GL, C)\), a world state is a tuple \(w = (GL, C, BS_1, BS_2, ... , BS_n)\) where each \(BS_i\) is the belief state for character \(i\).
\(<T_w, F_w, BS_{c_1}, \ldots BS_{c_n}\) where \(T_w\) and \(F_w\) together form a partition of \(GL\), where \(T_w\) designates all the ground literals that are true at \(w\), \(F_w\) includes all the ground literals that are false at \(w\) and each \(BS_{c_i}\) designates the belief state for character \(c_i\) at \(w\), where \(1 \leq i \leq |C|\).

**Definition** (Epistemic Goal Specification). Given a world frame \(W = (GL, C)\), an epistemic goal specification for some character \(c \in C\) is a tuple \(EG_c = \langle B^+_c, B^-_c, U_c \rangle\) such that \(B^+_c, B^-_c\) and \(U_c\) contain only elements from \(GL\) and have no common elements, where \(B^+_c\) designates all the ground literals that \(c\) should believe to be true, \(B^-_c\) includes all the ground literals that \(c\) should believe to be false and \(U_c\) designates all the ground literals that \(c\) should not believe to be true and should not believe to be false.

**Definition** (Master Goal Specification). Given a world frame \(W = (GL, C)\), a master goal specification is a tuple \(MGS = \langle T_w, F_w, EG_{c_1}, \ldots EG_{c_n}\rangle\) where each element of the tuple is a set that contains only elements from \(GL\), \(T_w \cap F_w = \emptyset\), where \(T_w\) designates all the ground literals that must be true at some goal state, \(F_w\) includes all the ground literals that must be false at some goal state and each \(EG_{c_i}\) designates the epistemic goal specification that must hold for character \(c_i\) at the goal state, where \(1 \leq i \leq |C|\).

A ground operator is a complete specification of an action in terms of the character performing the action, the conditions that must be true or false in the world in order for the action to execute, what the character performing the action must believe about the world in order for her to take the action, and how the action, once successfully executed, changes the world and the beliefs of the performing character.

**Definition** (Ground Operator). A ground operator \(GOP\) is a tuple \(GOP = (c, \text{PRE-}T, \text{PRE-}F, \text{PRE-}B^+, \text{PRE-}B^-, \text{PRE-}U, \text{EFF-}T, \text{EFF-}F, \text{EFF-}B^+, \text{EFF-}B^-, \text{EFF-}U)\) such that

- \(\text{PRE-}T, \text{PRE-}F, \text{PRE-}B^+, \text{PRE-}B^-, \text{PRE-}U, \text{EFF-}T, \text{EFF-}F, \text{EFF-}B^+, \text{EFF-}B^-, \text{EFF-}U \subseteq GL\)
- \(\text{PRE-}T \cap \text{PRE-}F = \emptyset, \text{PRE-}B^+ \cap \text{PRE-}B^- = \emptyset, \text{PRE-}U = \emptyset, \text{EFF-}T \cap \text{EFF-}F = \emptyset, \text{EFF-}B^+ \cap \text{EFF-}B^- = \emptyset, \text{EFF-}U = \emptyset\)
- \(c \in C\).

Informally, \(c\) designates the character initiating (or performing) the ground operator, \(\text{PRE-}T\) and \(\text{PRE-}F\) indicate the conditions in the world that must be true or false in order for the operator to execute, \(\text{PRE-}B^+\) and \(\text{PRE-}B^-\) and \(\text{PRE-}U\) indicate the conditions that \(c\) must believe to be true, false or unknown in the world in order for \(c\) to consider the operator executable; \(\text{EFF-}T, \text{EFF-}F\) indicate the conditions in the world that become true or false upon the action’s successful execution, and \(\text{EFF-}B^+\), \(\text{EFF-}B^-\), \(\text{EFF-}U\) indicate the conditions that \(c\) comes to believe are true, false or unknown in the world state resulting from the operator’s successful execution.

- \(\text{PRE-}T\) indicates the conditions in the world that must be true in order for the operator to execute.
- \(\text{PRE-}F\) indicates the conditions in the world that must be false in order for the operator to execute.
- \(\text{PRE-}B^+\) indicates the conditions that \(c\) must believe to be true in the world in order for \(c\) to consider the operator executable.
- \(\text{PRE-}B^-\) indicates the conditions that \(c\) must believe to be false in the world in order for \(c\) to consider the operator executable.
- \(\text{PRE-}U\) indicates the conditions that \(c\) must believe to be true or false in the world in order for \(c\) to consider the operator executable.
- \(\text{EFF-}T\) indicates the conditions that become true in the world state resulting from the operator’s successful execution.
- \(\text{EFF-}F\) indicates the conditions that become false in the world state resulting from the operator’s successful execution.
- \(\text{EFF-}B^+\) indicates the conditions that \(c\) believes become true in the world state resulting from the operator’s successful execution.
- \(\text{EFF-}B^-\) indicates the conditions that \(c\) believes become false in the world state resulting from the operator’s successful execution.
- \(\text{EFF-}U\) indicates the conditions that \(c\) neither believes are true nor are false in the world state resulting from the operator’s successful execution.

We call the preconditions and effects that refer to literals that are true or false in the physical world material and those that specify beliefs of the character epistemic. In Headspace, beliefs are always held by a particular agent, and only about ground literals and their truth values. There are no nested beliefs, no existential or universal quantification over beliefs and no implications defined over beliefs.

**Constructing Story Plans from Planning Problem Specifications**

Typical planning representations include a set of schematized action operators characterizing the classes of actions that can occur in a domain. In our approach, we take a set of such operators and a set of object constants and generate a world frame and a set of ground operators from them. This pre-processing is comparable to typical grounding processes used by forward-state planning algorithms (e.g., those of Nebel and Hoffmann (Nebel and Hoffmann 2001)).

A planning problem, then, is a tuple \(\mathcal{P} = \langle W, w_0, MGS, GO \rangle\) including a world frame \(W\) describing all possible ground literals and characters in a domain, an initial world state \(w_0\) characterizing the truth values of all literals and the beliefs of all characters, a goal specification \(MGS\) giving a partial description of a goal world and a set of ground operators \(GO\) available for characters to execute in the domain.

An action, represented by a ground operator, is executable in some state \(w\) just when all its material preconditions obtain in \(w\). We say that a ground operator is unexecutable in state \(w\) just when it is not executable in \(w\). An action is apparently executable in some state \(w\) for a character \(c\) just
Algorithm 1: HEADSPACE algorithm. For Planning Problem $\mathcal{PP} = (WF, w_0, G, GO)$ and plan heuristic ranking function $H$, call $\text{HEADSPACE}(WF, H, (\langle \bot, w_0 \rangle), G, GO)$.

1. $\text{HS}((GL, C), H, \text{Plans}, MGS, GO)$
2. Using heuristic ranking function $H$, rank all plans in Plans. Let $P$ be the highest ranked plan in Plans.
3. if $P$ is a solution then
4. Return $P$
5. else
6. Let $w$ be $w_k$, the world state in $k$th (final) tuple in the plan $P$
7. Let $AE = \emptyset$
8. for all $c \in C$ do
9. Let $AE = AE \cup \{c\}$
10. for all $a \in AE$ do
11. if $a$ is executable by $c$ in $w_k$ then
12. Let $w'$ be the world state resulting from $c$ executing action $a$ in world state $w$
13. else
14. Let $w'$ be the world state resulting from attempting action $a$ in world state $w$
15. end if
16. Append $\langle a, w' \rangle$ to the end of $P$
17. Let $\text{Plans} = \text{Plans} \cup P$
18. end for
19. Call $\text{HEADSPACE}(WF, H, \text{Plans}, G, GO)$
20. end if

when $c$’s belief state in $w$ supports all of the action’s epistemic preconditions in $w$. We say that a ground operator is apparently unexecutable for $c$ in state $w$ just when it is not apparently executable for $c$ in $w$.

A plan for some planning problem $\mathcal{PP} = (W, w_0, MGS, GO)$ is an ordered sequence of tuples where, for each tuple $\langle a_i, w_i \rangle$, $a_i$ indicates the $i$th action in the plan (attempted in world state $w_i$) and $w_i$ the state that obtains after $a_i$ was attempted. A solution for some planning problem $\mathcal{PP} = (WF, w_0, MGS, GO)$ is a plan $P$ for $\mathcal{PP}$ where, for every tuple $\langle a_i, w_i \rangle$ in $P$, $a_i$ is apparently executable in $w_{i-1}$, $w_i$ is the world resulting from attempting $a_i$ in $w_{i-1}$, and for a plan of length $k$, $w_k$ supports MGS.

Plan Generation

The $\text{HEADSPACE}$ algorithm, shown in Algorithm 1, uses forward-directed state-space search. Search starts at a given initial state, and the transition from a given state to its successor states is made through the ground operators that are apparently executable by the characters in the given state. Given a world frame $W = (GL, C)$ and a world state $w_i = \{T_{w_i}, F_{w_i}, BS_{c_1}, \ldots BS_{c_m}\}$, the planner generates successor states for $w_i$ as follows. First, the planner generates the set of all apparently executable ground operators at $w_i$, designated $AE_{w_i}$, by taking the union of all actions that appear executable in $w_i$ to each character $c_k, 1 \leq k \leq |C|$. For every executable action in $AE_{w_i}$, the algorithm finds the resulting world state from executing the action at $w_i$. The resultant world state is computed by applying all the effects of the action. Both material effects (i.e. Eff-T.Eff-F) as well as the epistemic effects (i.e. Eff-B.Eff-B-FF.Eff-U), with the latter applied to the belief state of the character performing the action.

For two adjacent tuples $\langle a_i, w_i \rangle$ and $\langle a_{i+1}, w_{i+1} \rangle$ in a plan $P$, when $a_{i+1}$ is an executable action, we say that $a_{i+1}$ was executed by $c_{i+1}$ in $w_i$, resulting in $w_{i+1}$. For two adjacent tuples $\langle a_i, w_i \rangle$ and $\langle a_{i+1}, w_{i+1} \rangle$ in a plan $P$, when $a_{i+1}$ is an unexecutable action, we say that $a_{i+1}$ was attempted by $c_{i+1}$ in $w_i$, resulting in $w_{i+1}$.

When unexecutable actions are attempted by a character, the actions fail. We call the manner in which the planner manages this kind of action failure the planner’s failure policy. In the current work, we define a relatively straightforward failure policy. First, with regard to action occurrence, none of the attempted action’s effects obtain and no material conditions in the world change. In effect, the action does not execute. Second, with respect to failure detection, the character executing the failed action immediately detects that it fails, but no other character detects the failure. Third, with respect to local attribution, the character executing the failed action assumes that the failure was due neither to execution error nor to an error in the definition of the ground operator. Rather, the character assumes that the failure was due to one or more of the action’s epistemic preconditions not holding in the action’s world state.

Formally, an epistemic update occurs when an action is attempted but fails. The epistemic update creates a new world state where the material state is unchanged, but the belief states are modified: the character that attempted the action is now uncertain about the preconditions of the attempted action holding, i.e. literals in $\text{PRE}^+ \text{PRE}^- \text{PRE}-F$, the attempted action are all added to the $U_c$ belief state of the character. The character does not attribute the cause of failure to the world, but rather, to their own beliefs being inconsistent with the true state of the world. The character then becomes certain about the conditions that can be verified by passive sensing (for example, that the character is still holding the gun), but the character does not need to perform the active sensing actions immediately. The character chooses to construct a plan with their current knowledge on how they can achieve the goal. The steps that form the plan may include the active sensing actions. However, it is also important to note that these steps form a plan to achieve the character’s goals, and the character already has an optimal result of the sensing action along with the rest of the steps of the plan in their head.

Heuristic Implementation

In the algorithm described for $\text{HEADSPACE}$, a heuristic function is used to select the next action to add to a plan during plan construction. We define below a heuristic for determining the next step, which extends the FastForward heuristic by Hoffman and Nebel (Hoffmann and Nebel 2001), taking in to account the context in $\text{HEADSPACE}$ afforded by
multiple characters each holding distinct beliefs about the state of the world.

In the FastForward algorithm, search for next steps to add to a plan is conducted using a relaxed plan graph, a model of possible action sequences that relaxes many of the constraints around plan construction in order to provide an efficient heuristic estimate of the distance to a plan solution. The algorithm for computing the relaxed plan graph for the heuristic calculation is provided in Algorithm 1. There are two major additions to this process that extend the FastForward heuristic calculation. While the algorithm constructs layers similar to the original FastForward algorithm, it stores the world states and character states separately at each time step. This allows for the relaxed plan graph to track not only the changes in the world state and compare it with the authorial goals for the world, but also allows for tracking changes in character beliefs and continuing the plan construction process until the authorial goals for both material conditions in the world and character beliefs are met (as seen in line 6). The second addition is that for any action that can possibly fail, the epistemic update due to attempting to perform the action (and failing) is also added to the relaxed plan graph (line 16). In other words, the heuristic allows for a character to consider the possible effects of failing as well as succeeding while performing an action.

This extended relaxed plan graph construction allows for constructing a graph that extends the world state towards the authorial goals for the world, and the various character beliefs toward the authorial goals for character beliefs simultaneously, and also accounts for the possible effects of actions failing at any point. Once this relaxed plan graph is constructed, we calculate the depth of this graph using the original FastForward relaxed plan graph depth algorithm. This heuristic is then provided to the HEADSPACE algorithm to determine the best action from the possible steps.

**Example**

To demonstrate the range of belief dynamics and the interaction between belief and execution in HEADSPACE, we define a simple story domain we call the Drink Refill domain. Ground operators for the domain are shown in Table 1, although space limitations required that we list only those ground operators from the domain that are used in the particular plan we examine.

The Drink Refill domain example makes use of seven operators. They are HOLD, where a character previously holding nothing holds an object, POUR-DRINK, where a character holding a bottle can pour a drink from it, CHECK-BOTTLE-EMPTY, where a character can take a closer look at the bottle that they are holding in order to determine if the bottle is empty, PLACE-DOWN, where a character places an object down, SERVE-DRINK, where a character serves a filled drink to a customer, OBSERVE-LOCAL+, where a character observes the location of an object that’s in the same location as the character, and OBSERVE-HOLDING+, where a character observes what they are currently holding in their hand.

As an example, consider a planning problem in this domain that involves a bartender refilling a thirsty customer’s drink glass. An informal sketch of the planning problem’s initial state sets a character, Teddy, to serve as a bartender. The goal for the story is for the drink to be refilled and served to the customer. The plan for the story is shown in Figure 1. In the story plan, Teddy means to hold the bottle that they believe is filled with the drink, pour a refill, and then serve it back to the customer. The plan in Figure 1 shows the actual executed story actions. In world state $w_0$, Teddy believes that Bottle 1 and Bottle 2 are not empty, the drink needs to be refilled, and they are not holding anything. His beliefs at $w_0$ are correct except for the fact that Bottle 1 is actually empty. Teddy first holds Bottle 1, then attempts to pour liquid from it into the glass, thus refilling the drink. However, because the bottle is empty, the action fails. At this point, he realizes that the action failed, and becomes uncertain about just those beliefs that were involved in the failed action’s preconditions.

Thus, in the resulting state, $w_2$, all of the epistemic preconditions for Teddy’s execution of Action 2 (the first POUR-DRINK action) have been asserted as unknown in his belief model. Teddy then passively senses his own location (Action 3) and the location of Bottle 1 (Action 4) and the glass (Action 5). He then passively senses that he’s holding

---

**Algorithm 2: Algorithm for enhanced relaxed plan graph construction for the FastForward Algorithm.**

As input we accept the initial state of the world $w_0$, the initial belief states of the characters $C_0 = c_{l0}, c_{20}, ..., c_{n0}$, the goal conditions for the world $w_G, goals$ for various characters as defined by the author $C_G = c_{1G}, ..., c_{nG}$, a set of ground operators $GO$, call ComputeRelaxedPlanGraph($GO, w_0, w_G, C_0, C_G$).

```
ComputeRelaxedPlanGraph(GO, w_0, w_G, C_0, C_G)
2: Let $l$ be the Layers to be constructed.
3: Let $t = 0$
4: $l.F_w.add(t, w_0)$
5: $l.F_c.add(t, C_0)$
6: while $L.F_c(t) \neq C_G$and$L.F_w(t) \neq G$
7: $t = t+1$
8: for all $o \in GO$
9: if $o$ is apparently executable then
10: $l.A.add(t, o)$
end if
11: end for
12: $l.F_w.add(t, l.F_w(t-1))$
13: $l.F_c.add(t, l.F_c(t-1))$
14: for all $o \in l.A(t)$ do
15: Let $w'$ be the relaxed world state resulting from attempting and failing to execute action $o$ in world state $w$.
16: Let $w''$ be the relaxed world state resulting from executing action $o$ successfully in world state $w$
17: $l.f(t) = l.f(t) \cup w' \cup w''$
end for
18: end for
19: return $l$
```
In this paper, we propose an approach for narrative planning that is capable of producing plot-level structures that are not generated by current narrative planners. Specifically, HEADSPACE can generate plans that allow for characters’ actions to fail, and those failures can prompt belief revision as characters seek new ways to achieve their goals.

As a brief characterization of HEADSPACE’s plan generation process, we provide summary metrics for the Drink Refill domain and specific planning problem described above. The domain consists of six unique operators. The shortest solution plan length for the planning problem consists of seven steps, including failed actions. The search space explored consisted of 19 nodes, with a maximum branching factor of 4 and an average branching factor of 2. Running on an Intel i7-4280K CPU at 3.7GHz with 16GB of RAM, the time taken for the planning algorithm to generate the shortest plan was 49 milliseconds. In this paper, we focus below on an analytical evaluation that characterizes HEADSPACE based on its expressive range (Smith and Whitehead 2010).

While there have been narrative planning approaches that use metrics drawn from conventional planning evaluation (e.g., minimum plan length and plan cost (Teutenberg and Porteous 2015)), we argue that such metrics, used alone or as the over-riding indicator of the strength of an algorithm,
would be inappropriate. Plot lines and the plan-based structures that can be used to characterize them are commonly not optimal or cost efficient, due in part to the limitations of the characters performing the plot’s actions. HEADSPACE features plans that incorporate failed actions, where characters have incorrect beliefs that might lead them to perform actions that are not optimal for their goals. This could lead to narratives which are longer than ones where the least amount of actions lead to the goal state being achieved. Moreover, in scenes such as the example from Avengers: Endgame, metrics such as plan length and minimum cost would not lead to such narratives. Hence, while plan efficiency metrics are important contributors to the design of appropriate story plan generation methods, there is value in narrative planning approaches that generate plans with properties that would not conventionally be considered ideal.

A significant contribution of HEADSPACE is its ability to produces plans that no other planning approach currently generates: plans where actions fail because of incorrect beliefs and characters subsequently correct those incorrect beliefs where possible and carry on in pursuit of their goals. Our work differs from related efforts in several ways.

Teutenberg and Porteous (Teutenberg and Porteous 2015) propose an approach which focuses on characters holding distinct sets of beliefs in order to support deception. In their work, however, their planner is not capable of producing plans containing action failure. Similarly, the Glaive planner (Ware and Young 2014) supports characters holding distinct sets of beliefs about the world, and Glaive does form plans where characters pursue sub-plans that, if fully executed, would not succeed (due to unanticipated conflict with other characters’ subplans). However, Glaive plans also contain no failed actions. Plans produced by VST (Ten Brinke, Linssen, and Theune 2014) contain subplans built using mistaken character beliefs. Like Glaive, however, characters in VST plans always detect their mistaken beliefs prior to attempting actions that would otherwise fail, and so update their plans to avoid action failure. Much like the plans produced by HEADSPACE, the plans produced by the method defined by Christensen, Nelson, and Cardona-Rivera (Christensen, Nelson, and Cardona-Rivera 2020) contain failed actions. However, their approach cannot produce plans where a character’s belief about a proposition changes to ignorance (as happens in HEADSPACE when an action fails). As a result, their method cannot produce plans where information-seeking behavior arises because of action failure and subsequent actions take that plan repair into account.

While these other systems address aspects of character belief and plan/action failure, none of them provide an integrated model producing narratives that involve characters that have incorrect beliefs about the world, plan to take actions that they perceive to be executable, attempt but fail at those actions, and as a result, become ignorant about conditions in the world related to their failed action’s execution. Furthermore, while our approach demonstrates this increased expressivity, it also generates narratives where characters do not have distinct sets of beliefs (e.g., where all characters have complete knowledge of the world) and, as a result, actions do not fail. This is achieved in our approach by assigning all characters full, correct belief in the initial state and restricting the syntax of our operators to have identical epistemic and material preconditions and effects. In this way, a character would never have incorrect beliefs, and would never include in their plan an action whose preconditions were not materially satisfied at execution.

Discussion and Future Work

The HEADSPACE planning algorithm provides an initial definition of a knowledge representation and planning algorithm to generate plots containing actions that fail due to characters’ incorrect beliefs. Analytic evaluation suggests an increased expressive range compared to other story-planning systems. However, our current method for updating characters’ belief states after failed actions have several limitations that we are currently addressing. First, when an action fails, the performing character transfers all preconditions of the failed action into the unknown partition of their belief state. A more informed credit assignment algorithm could do better at picking which preconditions should be called into question upon failure of an action. Second, characters other than the ones performing actions do not become aware of the actions’ success or failure. Third, because preconditions and effects do not make reference to actions, characters currently do not become aware of actions as they execute (or are attempted). Finally, we are looking to leverage prior work by Young (Young 2017) to connect action failure with the intention.

References


Thorne, B.; and Young, R. M. 2017. Generating Stories that Include Failed Actions by Modeling False Character Beliefs. In *Working Notes of the AIIDE Workshop on Intelligent Narrative Technologies*. Salt Lake City, UT.