Proceedings of the Seventeenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment
(AIIDE 2021)

Sabre: A Narrative Planner Supporting Intention and Deep Theory of Mind

Stephen G. Ware, Cory Siler
Narrative Intelligence Lab
University of Kentucky, Lexington, KY, USA 40506
sgware@cs.uky.edu, cory.siler@uky.edu

Abstract

Sabre is a narrative planner—a centralized, omniscient decision maker that solves a multi-agent storytelling problem. The planner has an author goal it must achieve, but every action taken by an agent must make sense according to that agent's individual intentions and limited, possibly wrong beliefs. This paper describes the implementation of Sabre, which supports a rich action syntax and imposes no arbitrary limit on the depth of theory of mind. We present a search procedure for generating plans that achieve the author goals while ensuring all agent actions are explained, and we report the system's performance on several narrative planning benchmark problems.

Introduction

Virtual environments with interactive stories, like games, training simulations, and tutoring systems, often give the player control of one or more characters while the system controls the others. These virtual characters need to seem like realistic agents with their own goals and limited, possibly wrong beliefs. Story generation techniques fall on a spectrum from *emergent* to *planned* (Riedl and Bulitko 2013). Emergent systems generate a story from the interactions of many realistic agents but struggle with "herding cats" when the narrative is required to contain certain content. Planning systems centralize decision making to ensure they meet the author's constraints but struggle to make agents seem realistic. Narrative planning is an interesting problem because it can be performed by a single agent but needs to generate a solution befitting a multi-agent system. A traditional planner ignores agent realism, while a traditional multi-agent system suffers limitations that don't apply in virtual worlds, where an omniscient and omnipotent author can coordinate the story behind the scenes.

This paper presents Sabre, a forward-chaining state-space narrative planning system that models both the intentions of the author (i.e. the system designer's constraints on the solution) as well as the intentions and beliefs of each virtual character. Sabre finds plans that improve the author's utility but are only composed of actions that can be explained by the intentions and beliefs of the characters who take them. To our knowledge it is the first such system to support full ADL action syntax (Pednault 1987) and an arbitrarily deep theory of mind. We measure Sabre's performance on benchmark problems and compare its solution space to ablated versions to motivate its unique combination of features.

Related Work

Kybartas and Bidarra (2016) survey many approaches to narrative generation, including expert systems, planning, multi-agent simulations, and case-based reasoning. Since then, deep learning techniques have also been applied (Martin et al. 2018). We focus this survey on planning systems.

One branch of research prefers to use existing planning algorithms to generate stories. Examples include HTN planning to generate plots for the television show *Friends* (Cavazza, Charles, and Mead 2002) and fast classical planners to make *The Merchant of Venice* interactive (Porteous, Cavazza, and Charles 2010). Work by Porteous et al. (2010), in particular, highlights the use of PDDL 3 trajectory constraints and planning landmarks to control story pacing.

Another branch integrates computational models of narrative directly into the knowledge representation and search of the planner. Young et al. (2013) survey planning algorithms that model intentionality, conflict, surprise, suspense, and other phenomena. Some focus on generating the events of the story (plot), while others focus on the telling (discourse). Sabre integrates narrative models into the algorithm and generates plot, so we focus our survey here.

Intention Riedl and Young's IPOCL (2010) introduced intentional planning, which defines both author and character goals. A valid plan accomplishes the author goal but can only be composed of actions that contribute to the goals of the characters who take them. Ware et al. (2014) extended their model to allow conflict and failed plans. In CPOCL (Ware and Young 2011) and Glaive (Ware and Young 2014), a character action no longer needs to be part of a successfully executed plan to achieve their goal; as long as such a plan exists, the action is reasonable, even if that plan is never actually executed, perhaps because it failed or conflicted another agent's plan. Sabre uses a similar model, except that it uses utility functions instead of discrete propositional goals.

Theory of Mind Several narrative planners focus on character knowledge, partial observability, and wrong beliefs. Virtual Storyteller (Brinke, Linssen, and Theune 2014), HeadSpace (Thorne and Young 2017), and work by Christensen et al. (2020) use a 1 layer theory of mind, meaning they reason about what is true, and what each character believes is true, but not what x believes y believes, and so on. IMPRACTical (Teutenberg and Porteous 2013) uses a 1 layer model, and past that defers to a shared global set of popular beliefs. Shirvani et al. (2017) demonstrate that planner-generated stories need a multi-level theory of mind to model some forms of deception, cooperation, anticipation, and surprise. Sabre places no arbitrary limit on the depth of theory of mind.

Si and Marsella's Thespian (2014) treats theory of mind as central, while Ryan et al.'s Talk of the Town (2015) models characters that observe, forget, and lie. These and others like them are multi-agent systems; they use a host of agents with true partial observability and leverage little centralized planning to coordinate the story.

Intention + Theory of Mind At least two systems model intentions and theory of mind for centralized planning. Previously we defined a state space with these features (Shirvani, Ware, and Farrell 2017) but focused on validating the knowledge representation and did not provide a planning algorithm or measuring performance. Ostari (Eger and Martens 2017) has these features but models true uncertainty via Dynamic Epistemic Logic. The high cost of modeling all doxastically accessible possible worlds limits the scope of problems it can solve. In Sabre's model, agents can have arbitrarily nested and wrong beliefs, but must always commit to specific beliefs. We found this tradeoff allows us to tell most of the stories we want to tell, and it allows Sabre to solve larger problems.

Problem Definition

We use a domain from our previous work (Ware et al. 2019) as a running example. Tom needs to get a potion for his grandmother. He begins in her cottage with one coin. There is a merchant at the market who is selling medicine and a sword for one coin each. Also at the market is a guard with a sword who wants to punish criminals. In a nearby camp, a bandit has a sword and one coin and wants to acquire more valuable items, like coins and medicine. A crossroads connects the cottage, the market, and the bandit's camp. Characters can walk from place to place. Characters with a coin can buy an item from the merchant. Armed characters (i.e. those who have swords) can attack and kill other characters. Armed characters can steal items from unarmed characters.

Characters and Fluents

A Sabre problem defines a finite number of *characters*, special entities which should appear to have beliefs and intentions. Henceforth we use the term "character" rather than "agent" because the narrative planner is the only decision maker, though it creates the appearance that each character is an agent. Tom, the merchant, the guard, and the bandit are the characters in our example domain.

A problem defines some number of state *fluents*, properties whose values can change over time. For some fluent f, let D_f denote the set of possible values f can take on, called the *domain* of f. Sabre supports two kinds of fluents: nominal and numeric. For nominal fluents, D_f is a finite set of possible nominal values. For numeric fluents, $D_f = \mathbb{R}$. We denote numeric fluents as $f_{\mathbb{R}}$.

Sabre supports seven types of logical literals l. The first six are given by this grammar:

$$\begin{array}{ll} l := & f = v \mid f \neq v \mid f_{\mathbb{R}} > n \mid f_{\mathbb{R}} \geq n \mid f_{\mathbb{R}} < n \mid f_{\mathbb{R}} \leq n \\ v := & \text{any value in } D_f \\ n := & v \mid f_{\mathbb{R}} \mid n + n \mid n - n \mid n \cdot n \mid n \div n \end{array}$$

These six kinds of literals have a fluent f on the left, a relation ($=, \neq, >$ etc.) in the middle, and a value on the right. For nominal literals, a value is one of D_f. For numeric literals, a value is a real number, a numeric fluent, or an arithmetic expression. Tom's location is an example of a nominal fluent whose domain is the cottage, market, camp, or crossroads. Tom's wealth is an example of a numeric fluent.

The seventh kind of literal takes the form $believes(c, l)$, which we abbreviate $b(c, l)$, and which means that character c believes literal l is true. Beliefs can be arbitrarily nested, so $b(c_1, b(c_2, f = v))$ means that character c_1 believes character c_2 believes that fluent f has value v. Sabre does not limit how deeply nested literals may be.

Logical Expressions

Sabre uses three kinds of complex logical expressions: *preconditions* that must be checked, *effects* that describe how states are modified, and *utility functions* that define preferences over states.

During pre-processing, preconditions are converted to disjunctive normal form. We use a process similar to Weld's (1994) to compile out first order quantifiers. Universal quantifications are replaced by conjunctions, and existential by disjunctions, like so:

$$\forall v(f = v) \leftrightarrow (f = v_1) \wedge (f = v_2) \wedge ...$$
$$\exists v(f = v) \leftrightarrow (f = v_1) \vee (f = v_2) \vee ...$$

In keeping with classical planners, preconditions and effects must be finite, so quantifiers over numeric fluents are not permitted. We consider $\top$ and $\bot$ to be in disjunctive normal form. $\top$ is a disjunction of one empty clause and always true, while $\bot$ is a disjunction of zero clauses and always false. Negated literals are compiled out like so:

$$\neg(f = v) \leftrightarrow (f \neq v)$$
$$\neg(f > v) \leftrightarrow (f \leq v) \text{ etc.}$$

Complex doxastic expressions about what a character believes are compiled out using these equivalencies:

$$\neg b(c, x) \leftrightarrow b(c, \neg x)$$
$$b(c, x \wedge y) \leftrightarrow b(c, x) \wedge b(c, y)$$
$$b(c, x \vee y) \leftrightarrow b(c, x) \vee b(c, y)$$

These axioms, especially the first (not believing x is equivalent to believing $\neg x$), reflects Sabre's constraint that characters cannot be uncertain in their beliefs.

Preconditions may not be contradictions. For example, the precondition $b(c, f = v) \wedge b(c, f \neq v)$ is not allowed (because characters must commit to beliefs).

Effects describe how the state after an event differs from the state before. Effects can be conditional, meaning they may not apply, depending on the state before the event. Like UCPOP (Penberthy and Weld 1992) and Fast Downward (Helmert 2006), Sabre represents all effects as having a condition, even if that condition is simply $\top$.

A single effect e can be described by this grammar:

$$\begin{aligned} e &:= \quad p \rightarrow g \\ g &:= \quad f = v \mid f_{\mathbb{R}} = n \mid b(c, g) \end{aligned}$$

All effects have a condition p in disjunctive normal form. The effect $p \rightarrow (f = v)$ means that, when p holds in the state before, fluent f has value v in the state after. Numeric fluents can be assigned numeric values following the grammar for n given above (i.e. n is a number, numeric fluent, or arithmetic expression). For example, $\top \rightarrow (f_{\mathbb{R}} = f_{\mathbb{R}} + 1)$ means that $f_{\mathbb{R}}$ has a value one higher in the state after.

Effects can modify character beliefs directly. $p \rightarrow b(c, f = v)$ means that, when p holds in the state before, character c believes fluent f has value v in the state after. Belief effects can also be arbitrarily nested, e.g. $p \rightarrow b(c_1, b(c_2, f = v))$ and so on. Having defined a single effect, we define an effect expression as a conjunction of effects:

$$(p_1 \rightarrow g_1) \wedge (p_2 \rightarrow g_2) \wedge ...$$

In keeping with classical planning, effects must be deterministic, so disjunctions and expressions equivalent to disjunctions (like existential quantifications) are not permitted in effect expressions. Effects may not be contradictions.

Utility functions are compiled into a normal form similar to effects. They are conditional, but exactly one condition must hold in any state. A utility function is an ordered sequence of m conditional numeric expressions $p \rightarrow n$:

$$\langle p_1 \rightarrow n_1, p_2 \rightarrow n_2, ..., \top \rightarrow n_m \rangle$$

Here, $m \geq 1$, $p_{1...m-1}$ are conditions in disjunctive normal form, and $n_{1...m}$ are numeric values following the grammar for n above (a number, numeric fluent, or arithmetic expression). The last condition p_m must be $\top$ to ensure that one case will always hold. Utility functions are similar to if/elseif/else statements in a programming language. To evaluate a utility function in a state, we consider each conditional expression $p_i \rightarrow n_i$ in order until we find one where p_i holds, then we evaluate n_i. In our example domain, the honest merchant (M) wants money. Her utility function is:

$$\langle criminal(M) = \top \rightarrow 0, \top \rightarrow wealth(M) \rangle$$

If she is a criminal, her utility is 0, else her utility is her number of coins. She prefers plans that increase her wealth, but will not commit crimes to do so.

Initial State and Goal

The initial state defines an assignment of a value to every fluent; i.e. $\forall f : f = v$ where $v \in D_f$. It also defines any wrong beliefs that characters initially hold; for example, $(f = v) \wedge b(c, f = u)$.

We use Shirvani et. al's (2017) variant of the *closed world assumption* to assume any beliefs not explicitly defined in the initial state. When a character's belief about a fluent is not specified, we assume they believe its actual value. So if $f = v$ and there is no explicit statement otherwise, we assume $b(c, f = v)$. Similarly, characters assume other characters have the same beliefs they do. If the initial state explicitly defines $b(c_1, f = v)$ but has no explicit statement for what c_1 believes c_2 believes, we assume $b(c_1, b(c_2, f = v))$. These assumptions are only used for the initial state.

A problem defines an *author utility function*, which expresses preferences for the states the planner should attempt to reach. Recall the planner is the only decision maker, but each character should appear realistic. A Sabre problem also defines a utility function for every character, and characters should act to improve their utility, but the planner chooses when and how they act to maximize the author's utility. In other words, the puppetmaster tells the best story it can without making its puppets act out of character.

Let $u(c_{author}, s)$ denote the author's utility in state s. Let $u(c, s)$ denote the utility of character c in state s. In our example domain, the author's utility is as follows (where T is Tom, C the cottage, S the medicine, and loc is "location"):

$$\langle loc(T) = C \wedge loc(S) = T \rightarrow 2, alive(T) = \bot \rightarrow 1, \top \rightarrow 0 \rangle$$

The system prefers stories where Tom succeeds in bringing home the medicine, and will accept stories where he dies.

Events: Actions and Triggers

Events change the world state. Domain authors can create two kinds of events: *actions*, which the planner can choose to take, and *triggers*, which must occur when they can.

An action a defines $\text{PRE}(a)$ and $\text{EFF}(a)$, which are its precondition and effect expressions respectively. An action also defines $\text{CON}(a)$, a set of 0 to many consenting characters. Consenting characters are the ones responsible for taking the action. For an action to make sense in a narrative plan, every consenting character needs a reason to take the action.

Actions also define when characters observe them occurring. Formally, an action a defines a function $\text{OBS}(a, c)$. For any character c, $\text{OBS}(a, c)$ returns a precondition expression p such that, when p holds, c observes action a occur. When a character observes an action, they update their beliefs based on the action's effects. When a character does not observe an action, their beliefs remain the same (unless explicitly modified by the effect). Consider this example action, where character c_1 attacks c_2 at location x:

Action a: $attack(c_1, c_2, x)$

$$\begin{aligned} \text{PRE}(a) &= \quad alive(c_1) = \top \wedge loc(c_1) = x \wedge loc(c_2) = x \\ \text{EFF}(a) &= \quad armed(c_1) \rightarrow alive(c_2) = \bot \\ \text{CON}(a) &= \quad \{c_1\} \\ \text{OBS}(a, c) &= \quad loc(c) = x \end{aligned}$$

Suppose c_1 is the bandit, c_2 is Tom, and x is the crossroads. This action can only occur if the bandit is alive and the bandit and Tom are both at the crossroads. After it occurs, if the bandit was armed then Tom is dead, or else the action has no effect. Only the bandit is a consenting character, because only she needs a reason to take the action (Tom is a victim, and the action can occur whether or not he wants it to happen). The observation function means that, for any

character c, when $loc(c) = x$ holds, c observes the action; i.e. any characters also at the crossroads will see the action happen. They will know Tom is dead, whereas those not at the crossroads will not know.

Triggers are events that must happen when their preconditions are met. A trigger t defines $\text{PRE}(t)$ and $\text{EFF}(t)$ as above. Any time the world is in a state where $\text{PRE}(t)$ holds, $\text{EFF}(t)$ must immediately be applied to change the world state. Actions advance time, but triggers do not. In other words, after time is advanced by taking an action, any number of triggers may then apply to update the state, but they happen instantly. If multiple triggers can apply in a state, Sabre chooses arbitrarily. To ensure determinism, triggers should not be defined such that their outcome depends on order of execution.

Triggers are similar to the axioms and derived predicates of PDDL planners (Thiébaux, Hoffmann, and Nebel 2005), but not identical. Triggers modify state fluents directly, whereas PDDL planners have basic predicates which can be directly modified by effects and special derived predicates whose values are deduced from basic predicates.

Triggers are commonly used for sensing and belief updates based on character observations. Consider this example, which means "When characters c_1 and c_2 are in the same place, c_1 sees that c_2 is there."

Trigger t: $see(c_1, c_2, x)$
$\text{PRE}(t) = \quad loc(c_1) = x \wedge loc(c_2) = x \wedge b(c_1, loc(c_2) \neq x)$
$\text{EFF}(t) = \quad \top \rightarrow b(c_1, loc(c_2) = x)$

Suppose the bandit is at the crossroads and believes Tom is in the cabin when Tom is actually at the market. If the bandit walks to the market, she should notice Tom is there and update her wrong belief about Tom's location.

Triggers represent rules of the world common to all, so they do not define consenting characters or observation functions (i.e. if a character believes a trigger can happen, it does, regardless of whether they want it to).

Action Results Action effects have a finite number of conjuncts as specified by the domain author, but because actions can be observed and because we impose no limit on the depth of theory of mind, actions can cause infinitely many changes to the world state. In the above example, when Tom is killed, the bandit knows it, and knows that she knows it, and knows that any bystanders know it, and knows that they know that she knows it, and so on. Shirvani et al. (2017) showed these changes can be expressed in a finite graph, and at any rate only a finite number of these changes actually need to be calculated.

Let $\text{RES}(a)$ be the results of action a, a possibly infinite conjunction composed of the action's explicitly authored effects and any effects implied by observations. Its conjuncts are in the same format as above, $p \rightarrow g$, where p is a condition and g is either an assignment of a value to a fluent or a belief. $\text{RES}(a)$ is defined by these two rules:

1. $\text{RES}(a) \models \text{EFF(A)}$
2. $\forall c : (\text{RES}(a) \models p \rightarrow g) \Rightarrow$
   $(\text{RES}(a) \models (\text{OBS}(a, c) \wedge b(c, p)) \rightarrow b(c, g))$

The first rule states the effects of a are also results of a. The second defines results implied by observations. If $\text{RES}(a)$ includes the effect $p \rightarrow g$, and character c observes action a occur, and c believes the condition p holds, then c will now believe g. If the guard is at the crossroads when the bandit attacks Tom, and the guard believes the bandit is armed, then the guard now believes Tom is dead even though this was not explicitly authored as an effect of the *attack* action.

Sabre's belief and observation model is a key strength. Authors can explicitly specify beliefs in preconditions and modify them in effects using the $believes$ predicate (for example, to create the lie action used in some domains (Farrell and Ware 2020)), but Sabre automates many common belief updates via observed actions and triggers.

State Space

Sabre starts at the initial state and searches forward through the space of possible future states for a solution. To save memory, Sabre does not store a state as an array of values for each fluent; rather, it stores the history of events that led to that state, and when it needs to know the value of a fluent, it looks back through the history for the last time the fluent was modified. Let s be a state, and let $\sigma(e, s)$ denote the state after event e (recall an event is an action or a trigger). $\sigma(e, s)$ is only defined when $s \models \text{PRE}(e)$, i.e. when the event's precondition holds in s. For any literal l, we can determine if l holds in $\sigma(e, s)$ using this procedure:

if e has result $p \rightarrow g$, and p holds in s, and $g \models l$ **then**
  return $\top$ ▷ Case 1
else if e has result $p \rightarrow g$, and p holds in s, and $g \wedge l \rightarrow \bot$ **then**
  return $\bot$ ▷ Case 2
else if $\text{PRE}(e) \models l$ **then return** $\top$ ▷ Case 3
else if $\text{PRE}(e) \wedge l \rightarrow \bot$ **then return** $\bot$ ▷ Case 4
else if l holds in s **then return** $\top$ ▷ Case 5
else return $\bot$ ▷ Case 6

There are six cases. Case 1 states that l holds if e has a result (whose condition holds) which makes l true. Case 2 states l does not hold if e has a result which contradicts l. Consider the precondition $f_{\mathbb{R}} > 1$. The effect $f_{\mathbb{R}} = 0$ is not an exact negation of it, but $f_{\mathbb{R}} > 1 \wedge f_{\mathbb{R}} = 0$ is a contradiction, so this effect would make that precondition $\bot$.

Characters can have wrong beliefs, so it is possible for them to observe an action happen which they did not believe was possible. We call this a *surprise action*. When a character is surprised, they first update their beliefs so the action's precondition holds and then update their beliefs based on its effect. If the guard believes the bandit is dead, but then observes the bandit attack Tom, this would be a surprise. The guard updates his beliefs to realize the bandit is alive and then considers the results of the attack. Cases 3 expresses this—that l holds if it is implied by e's precondition. Case 4 states that l does not hold if it contradicts e's precondition. Since cases 1 and 2 are checked first, 3 and 4 only apply when l was not changed by a result of the event. If none of these cases apply, event e has no bearing on l, so we check l in the previous state s (cases 5 and 6).

Recall that a trigger must be applied when its precondition holds, and triggers happen instantly. When the world state is s and there exists a trigger t such that $s \models \text{PRE}(t)$, the world must immediately transition to $\sigma(t, s)$. For an event e and state s, we define $\alpha(e, s)$ to be the state of the world after

taking event e and then applying any relevant triggers. α is defined recursively as:

function $\alpha(e, s)$
  if $\sigma(e, s)$ is undefined (i.e. $s \not\models \text{PRE}(e)$) **then**
    return undefined
  else if $\exists$ trigger t such that $\sigma(e, s) \models \text{PRE}(t)$ **then**
    return $\alpha(t, \sigma(e, s))$
  else return $\sigma(e, s)$

As shorthand, let $\alpha(\{a_1, a_2, ..., a_n\}, s)$ represent the state after a sequence of n actions taken in that order from state s.

Explanations and Solutions

Having defined Sabre's representation of the problem and its search space, we can now define a solution. Informally, a solution is:

- A sequence of actions which can be executed and leads to a state where the author's utility is improved.
- Every action taken by a character can be explained; i.e. each character believes the action can lead to increasing their utility.

To state this formally, we need a way to refer to what a character believes the state to be. Let c be some character, and let s be any state. We define $\beta(c, s)$ to be the state that c believes to be the case when the actual state is s. This equivalence defines $\beta(c, s)$:

$$s \models b(c, l) \leftrightarrow \beta(c, s) \models l$$

We say an action a_1 is *explained* in state s when it is explained for all of $\text{CON}(a_1)$, its consenting characters. An action a_1 is *explained for* character c in state s when:

1. There exists a sequence of $n \geq 1$ actions $\{a_1, a_2, ..., a_n\}$ that starts with a_1.
2. $\alpha(\{a_1, a_2, ..., a_n\}, \beta(c, s))$ is defined.
3. $u(c, \alpha(\{a_1, a_2, ..., a_n\}, \beta(c, s))) > u(c, \beta(c, s))$.
4. All actions a_i where $i > 1$ are explained in the state before they occur; i.e. a_2 is explained in $\alpha(a_1, \beta(c, s))$, a_3 is explained in $\alpha(\{a_1, a_2\}, \beta(c, s))$, and so on.
5. No strict subsequence of $\{a_1, a_2, ..., a_n\}$ also meets these 5 criteria and achieves the same or higher utility.

Requirement #1 states that there exists a plan which starts with action a_1. #2 states that character c believes the plan can be executed. In other words, the plan can be executed from $\beta(c, s)$, even if it cannot be executed from s, perhaps because c has wrong beliefs. #3 states that c believes the plan will lead to a state where their utility is higher (even if it actually won't). #4 states that the rest of the actions (after a_1) must also be explained. For example, if the plan relies on actions by other characters, those actions must make sense for those characters. #5 states that the sequence should not include redundant or unnecessary actions.

Suppose Tom is at the cottage, has one coin, and wants medicine. The merchant is at the market, has medicine, and wants coins. Tom knows where the merchant is and that the merchant has medicine, but the merchant does not know Tom has money. When characters are in the same place, they see what things each other have via a trigger. Would Tom walking to the market be explained?

To be *explained*, it must be *explained for* all of its consenting characters, which in this case is just Tom. Tom can imagine a plan to walk to the market, buy the medicine, and return to the cottage (#1). This plan is possible based on his current beliefs (#2), and he thinks it will lead to a state where his utility is higher (#3). Both Tom and the merchant are consenting characters for the *buy* action. Tom anticipates the merchant will let him *buy*, which is reasonable given his beliefs about her (#4). Note that, in the initial state, the merchant *cannot* form a plan to walk to the cottage and have Tom buy medicine, because the merchant does not think Tom has any money. However, once Tom arrives at the market, the merchant will see he has money via a trigger that updates the merchant's beliefs. Now the merchant will consent to *buy*. Requirement #5 means Tom cannot plan to walk to the market, walk home, walk back to the market, buy the medicine, and walk home. This plan is redundant; the first two actions could be removed and it would still achieve the same utility.

Finally, we define a solution. Let s_0 be the initial state. A solution to a Sabre problem is a sequence of n actions $\{a_1, a_2, ..., a_n\}$ such that:

1. $\alpha(\{a_1, a_2, ..., a_n\}, s_0)$ is defined.
2. $u(c_{author}, \alpha(\{a_1, a_2, ..., a_n\}, s_0)) > u(c_{author}, s_0)$.
3. All actions in $\{a_1, a_2, ..., a_n\}$ are explained in the state immediately before they occur.
4. No strict subsequence of $\{a_1, ..., a_n\}$ meets these criteria.

In short, the plan is possible, leads to a state where the author's utility is higher, has only actions that make sense for characters, and is non-redundant. We validated this model of believable actions with human subjects (Shirvani, Farrell, and Ware 2018), but we also acknowledge believable character behavior is more than simply seeking higher utilities. Parallel work is using Sabre to expand our definition to include emotion and personality (Shirvani and Ware 2020).

Pre-Processing and Simplification

Sabre does several pre-processing steps before planning begins. Recall that actions can have an infinite number of results. Fortunately, there are a finite number of preconditions that ever need to be checked—those that appear in event preconditions, action observation functions, effect conditions, and utility conditions. As long as Sabre calculates all *relevant* results (i.e. all results that would affect this finite list of preconditions) it can be sound. During pre-processing, Sabre does this, adding explicit effects to actions so that all relevant results are accounted for.

The astute reader may also notice a challenge that arises when dealing with triggers and beliefs. Triggers are not only checked in the current state s, but also in every character's beliefs. Consider an example trigger, where x and y are shorthand for any two literals:

Example trigger t:
$\text{PRE}(t) = \quad x \wedge \neg y$
$\text{EFF}(t) = \quad \top \rightarrow y$

Algorithm 1 The Sabre algorithm

```
1:  Let 𝒜 be the set of all actions defined in the domain.
2:  SABRE(c_author, s_0, ∅, s_0)
3:  function SABRE(c, r, π, s)
4:      Input: character c, start state r, plan π, current state s
5:      if u(c, s) > u(c, r) and π is non-redundant then
6:          return π
7:      Choose an action a ∈ 𝒜 such that s ⊨ PRE(a).
8:      for all c' ∈ CON(a) such that c' ≠ c do
9:          Let state b = α(a, β(c', s)).
10:         if b is undefined then return failure.
11:         else if SABRE(c', b, ∅, b) fails then return failure.
12:     return SABRE(c, r, π ∪ a, α(a, s))
```

And imagine c is a character and the state is:

$$x \wedge y \wedge b(c, x) \wedge b(c, \neg y)$$

The trigger does not apply in this state, but it does apply in $\beta(c, s)$. The state should immediately transition to:

$$x \wedge y \wedge b(c, x) \wedge b(c, y)$$

The above procedure defining α fails to capture this case. Even if it did, since Sabre does not limit how far theory of mind can be nested, it is difficult to know how many levels of beliefs need to be checked to make sure all triggers have been applied. We previously proposed a solution to this problem (Shirvani, Ware, and Farrell 2017), but it requires graph isomorphism checks. Sabre works differently.

During pre-processing, for every character c, if there exist a trigger t with the effect $p \rightarrow g$, and $b(c, g)$ is in the set of all possible preconditions, a new trigger t' is generated with $\text{PRE}(t') = b(c, \text{PRE}(t))$ and $\text{EFF}(t') = b(c, \text{EFF}(t))$. This ensures triggers also account for all relevant effects. To use the earlier example:

Original trigger t:			New trigger t':	
$\text{PRE}(t) =$	$x \wedge \neg y$	$\Rightarrow$	$\text{PRE}(t') =$	$b(c, x) \wedge b(c, \neg y)$
$\text{EFF}(t) =$	$\top \rightarrow y$		$\text{EFF}(t') =$	$\top \rightarrow b(c, y)$

The preconditions of these new triggers may add to the set of all possible precondition literals, so the process is repeated until no new triggers are needed. It is possible to construct triggers that would cause this process to run infinitely, in which case we would need to revert to our previous graph-based solution, but in practice we have never encountered such a domain.

Sabre also uses methods adapted from other planners (Hoffmann 2003; Helmert 2006) to simplify the problem by detecting propositions which must always be true or false. This sometimes allows the removal of fluents, actions, and triggers from the domain to reduce the time and memory required during search.

Search

Sabre's search procedure is given in Algorithm 1. It takes four inputs: the character c for whom we want to find a plan, the state r where the search began, a plan π, and the current state s. The plan π is the sequence of actions that can be executed from r to reach s. SABRE can find a plan for any character from any state. The initial call to SABRE (line 2) uses the special author character c_{author}, the initial state of the problem s_0, and the empty plan $\emptyset$.

SABRE starts by checking if the current plan is a solution (line 5). It is a solution if character c's utility is higher in s than it was in r and the plan does not contain unnecessary actions. If the current plan is not a solution, we nondeterministically choose an action a to add to the plan. Before exploring further, Sabre checks whether that action can be explained for all the consenting characters other than c (lines 8 to 11). When c is c_{author} Sabre needs to find an explanation for all consenting characters. When c is a character, they must be able to anticipate the cooperation of other characters who are part of their plan. Tom must expect the merchant will consent to the *buy* action to include it in his plan.

There are two cases where a cannot be explained for a consenting character c'. The first is when $\alpha(a, \beta(c', s))$ is undefined (line 10), meaning c' does not think a is possible. The second is when c' cannot imagine a plan starting with a that improves their utility (line 11), so they have no reason to consent to it. Tom cannot expect the merchant to simply give him the medicine, because it wouldn't improve the merchant's utility. If the action can be explained for all consenting characters besides c, we add a to the plan, advance the current state to $\alpha(a, s)$, and recursively call SABRE.

If the first call to SABRE on line 2 returns a plan, it is a solution to the problem; i.e. it can be executed from the initial state, leads to a state where the author's utility is higher, and all actions can be explained for all characters. SABRE may not terminate if no solution exists, so in practice we impose a maximum depth on the search.

Evaluation

Sabre has a unique set of features; it is a centralized planner that reasons about intentions and beliefs with no limit on theory of mind but without uncertainty. These features are ideal for the interactive narratives we generate, but they make it difficult to compare to other systems, so we use benchmark problems to convey the scope of what Sabre can solve and an ablation study to motivate its combination of features.

Intentional planners like IPOCL (Riedl and Young 2010), Glaive (Ware and Young 2014), and the original IMPRACTical (Teutenberg and Porteous 2013) do not reason about beliefs, so either Sabre must reason about belief when it is not required, or the problems are unsolvable by the intentional planners. HeadSpace (Thorne and Young 2017) and a later version of IMPRACTical (Teutenberg and Porteous 2015) limit theory of mind. Multi-agent systems like Thespian (Si and Marsella 2014) and Talk of the Town (Ryan et al. 2015) reason about theory of mind but have limited centralized planning, so they would need to run many times until they happen to achieve the author's goal. Ostari (Eger and Martens 2017) is centralized with intentions and beliefs but allows true uncertainty, which dramatically increases the size of its search space. As a test, the smallest example from the *Lovers* domain (described below) was implemented and tested in Ostari, but it quickly ran out of memory before finding any solutions. None of these makes for a fair empirical comparison.

Domain	Char.	Fluents	Actions	Triggers	Time	Visited	Generated
Raiders (1)	3	21	39	66	1.4 s	905	17,815
Space (1)	2	16	32	0	6 ms	18	192
Treasure (1)	3	4	34	0	1 ms	22	288
Hubris (1)	2	29	14	0	47 ms	58	831
BearBirdJr (1)	2	13	20	0	14.0 m	290,711	34,084,608
Lovers (9)	3	111.3	312.0	370.0	40.3 m	126,983	5,198,414
Grandma (2)	4	61.0	836.0	896.0	6.2 h	598,577	105,178,466

Sabre	Intention		Belief	
✓	✓	✗	✓	✗
3	0	0	0	110
2	1	1	1	0
2	0	0	2	2
1	0	0	1	2
6	0	0	0	110
10.0	0.0	0.0	3.2	50.7
1.0	0.0	0.0	1.0	13.0

Table 1: Performance on benchmark problems (left) and ablation study solution counts for those problems (right)

Since we cannot make a direct comparison, we demonstrate the scale of problems Sabre can solve with a suite of benchmark narrative domains from the literature by several authors. *Raiders* was introduced with Glaive (Ware and Young 2014) as a problem that requires failed plans and conflict; *Space* is an additional Glaive domain. Since Glaive domains were authored for a planner that did not support belief, we added common sense beliefs and triggers (e.g. characters observe actions that occur in their location, etc.). *Raiders* had a pseudo-belief predicate ("knows location of") which we replaced with true beliefs. *Treasure* (Shirvani, Farrell, and Ware 2018) and *Hubris* (Christensen, Nelson, and Cardona-Rivera 2020) require intentions and wrong beliefs. *BearBirdJr* simplifies Sack's (1992) Micro-TaleSpin version of Meehan's (1977) story generator. *Lovers* was introduced for belief and intention recognition tasks (Farrell and Ware 2020); this domain is parameterizable, so we randomly generated 10 instances known to be solvable. 9 were successfully solved by Sabre. *Grandma* is from a recent planning-based narrative game (Ware et al. 2019) with intention and belief and has two instances: one where the player wins and one where the player dies. We will publish these domains and problems with our modifications in a technical report.

Table 1 shows the number of characters, fluents, actions, and triggers for the problems in each domain after grounding and simplification. The table shows the time required to find the first solution for each problem in the domain and the number of nodes visited and generated by the search. When a domain has several problems (and thus may have a different numbers of triggers, solutions, etc.), we give averages. We used a deterministic A* version of Algorithm 1 (Pohl 1970), using Bonet and Geffner's h^+ heuristic (2001) on a Dell Precision 7920 x64 with a 2.1 GHz processor.

Since we cannot compare directly to other planners, we motivate Sabre's features via ablation. For our test problems, we generated all plans at or below a fixed length and counted which are solutions according to full Sabre, Sabre with only intention, and Sabre with only belief. Only intention means characters try to improve their utility, but beliefs are not tracked and all characters are omniscient; it approximates intentional planners like IPOCL, Glaive, and early IMPRACTical. Only belief means beliefs are tracked and characters only take actions they think possible, even if the actions can't lead to improving their utility; it approximates belief-only planners like HeadSpace. The fixed length for a problem is the min depth at which any Sabre solutions exist.

The right of Table 1 show how many solutions Sabre found vs. the ablated planners. ✓ counts solutions which were also valid Sabre plans, and ✗ counts those which were valid according to the ablated model but not Sabre. As emphasized above, it is unfair to compare these ablated planners on problems they were not designed to solve. These numbers do not prove Sabre better than previous planners; we report them only to motivate reasoning about intention and belief together. We want to show that one could not, for example, simply add an extra check on solutions generated by an intentional planner to get a system that effectively reasons about belief as well. We previously demonstrated that audiences significantly prefer stories with both intention and belief and notice flaws when one or the other is lacking (Shirvani, Farrell, and Ware 2018).

Implementation

The planner and relevant documentation can be found here:

http://cs.uky.edu/~sgware/projects/sabre

Limitations, Future Work, and Conclusion

This paper describes Sabre, the first narrative planner to support ADL, numeric fluents, intention, and deep theory of mind. Though Sabre enables automated story generation, the general problem of author burden remains; domain authors must still define fluents, actions, triggers, and utility functions, and debugging narrative planning domains is hard. Supporting theory of mind enables new stories (Shirvani, Ware, and Farrell 2017), but adds significant cost above an intention-only planner. Sabre does not support uncertainty, which may be necessary for some stories, like murder mysteries (Eger 2020).

The most obvious direction for future work is to explore improvements to the search process via pruning and better heuristics that account for the beliefs and intentions, an approach which dramatically improved performance for other narrative planners like Glaive and IMPRACTical.

Acknowledgments

This work was supported by the National Science Foundation, Grant No. IIS-1911053. All opinions are our own. We thank Markus Eger for implementing a sample problem in Ostari to evaluate the suitability of comparing it to Sabre. We thank Rachelyn Farrell for her help in converting domains to Sabre's format.

References

Bonet, B.; and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1): 5–33.

Brinke, H. t.; Linssen, J.; and Theune, M. 2014. Hide and Sneak: story generation with characters that perceive and assume. In *Proceedings of AIIDE*, 174–180.

Cavazza, M.; Charles, F.; and Mead, S. J. 2002. Character-based interactive storytelling. *IEEE Intelligent Systems special issue on AI in Interactive Entertainment* 17(4): 17–24.

Christensen, M.; Nelson, J.; and Cardona-Rivera, R. E. 2020. Using domain compilation to add belief to narrative planners. In *Proceedings of AIIDE*, volume 16, 38–44.

Eger, M. 2020. Murder mysteries: the white whale of narrative generation? In *Proceedings of AIIDE*, 210–216.

Eger, M.; and Martens, C. 2017. Character beliefs in story generation. In *Proc. of INT Workshop at AIIDE*, 184–190.

Farrell, R.; and Ware, S. G. 2020. Narrative planning for belief and intention recognition. In *Proc. of AIIDE*, 52–58.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26: 191–246.

Hoffmann, J. 2003. The Metric-FF planning system: translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research* 20: 291–341.

Kybartas, Q.; and Bidarra, R. 2016. A survey on story generation techniques for authoring computational narratives. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games* 9(3): 239–253.

Martin, L. J.; Ammanabrolu, P.; Wang, X.; Hancock, W.; Singh, S.; Harrison, B.; and Riedl, M. O. 2018. Event representations for automated story generation with deep neural nets. In *Proceedings of AAAI*, 868–875.

Meehan, J. R. 1977. TALE-SPIN, an interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, 91–98.

Pednault, E. P. D. 1987. Formulating multiagent, dynamic-world problems in the classical planning framework. In *Reasoning About Actions & Plans*, 47–82.

Penberthy, J. S.; and Weld, D. S. 1992. UCPOP: a sound, complete, partial order planner for ADL. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, volume 92, 103–114.

Pohl, I. 1970. First results on the effect of error in heuristic search. *Machine Intelligence* 5: 219–236.

Porteous, J.; Cavazza, M.; and Charles, F. 2010. Applying planning to interactive storytelling: Narrative control using state constraints. *ACM Transactions on Intelligent Systems and Technology* 1(2): 1–21.

Riedl, M. O.; and Bulitko, V. 2013. Interactive narrative: an intelligent systems approach. *AI Magazine* 34(1): 67–77.

Riedl, M. O.; and Young, R. M. 2010. Narrative planning: balancing plot and character. *JAIR* 39(1): 217–268.

Ryan, J. O.; Summerville, A.; Mateas, M.; and Wardrip-Fruin, N. 2015. Toward characters who observe, tell, misremember, and lie. In *Proceedings of EXAG Workshop at AIIDE*, 56–62.

Sack, W. 1992. Micro-TaleSpin: a story generator. http://lispm.de/source/misc/micro-talespin.lisp. Accessed: 2021-08-06.

Shirvani, A.; Farrell, R.; and Ware, S. G. 2018. Combining intentionality and belief: revisiting believable character plans. In *Proceedings of AIIDE*, 222–228.

Shirvani, A.; and Ware, S. G. 2020. A formalization of emotional planning for strong-story systems. In *Proceedings of AIIDE*, 116–122.

Shirvani, A.; Ware, S. G.; and Farrell, R. 2017. A possible worlds model of belief for state-space narrative planning. In *Proceedings of AIIDE*, 101–107.

Si, M.; and Marsella, S. C. 2014. Encoding Theory of Mind in character design for pedagogical interactive narrative. *Advances in Human-Computer Interaction* .

Teutenberg, J.; and Porteous, J. 2013. Efficient intent-based narrative generation using multiple planning agents. In *Proceedings of the 2013 international conference on Autonomous Agents and Multiagent Systems*, 603–610.

Teutenberg, J.; and Porteous, J. 2015. Incorporating global and local knowledge in intentional narrative planning. In *Proceedings of the 2015 international conference on Autonomous Agents and Multiagent Systems*, 1539–1546.

Thiébaux, S.; Hoffmann, J.; and Nebel, B. 2005. In defense of PDDL axioms. *Artificial Intelligence* 168(1-2): 38–69.

Thorne, B. R.; and Young, R. M. 2017. Generating stories that include failed actions by modeling false character beliefs. In *Proceedings of INT Workshop at AIIDE*, 244–251.

Ware, S. G.; Garcia, E. T.; Shirvani, A.; and Farrell, R. 2019. Multi-agent narrative experience management as story graph pruning. In *Proceedings of AIIDE*, 87–93.

Ware, S. G.; and Young, R. M. 2011. CPOCL: a narrative planner supporting conflict. In *Proc. of AIIDE*, 97–102.

Ware, S. G.; and Young, R. M. 2014. Glaive: a state-space narrative planner supporting intentionality and conflict. In *Proceedings of AIIDE*, 80–86.

Ware, S. G.; Young, R. M.; Harrison, B.; and Roberts, D. L. 2014. A computational model of narrative conflict at the fabula level. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games* 6(3): 271–288.

Weld, D. S. 1994. An introduction to least commitment planning. *AI magazine* 15(4): 27–61.

Young, R. M.; Ware, S. G.; Cassell, B. A.; and Robertson, J. 2013. Plans and planning in narrative generation: a review of plan-based approaches to the generation of story, discourse and interactivity in narratives. *Sprache und Datenverarbeitung, Special Issue on Formal and Computational Models of Narrative* 37(1-2): 41–64.