A Collection of Benchmark Problems for the Sabre Narrative Planner

Technical Report

November 18, 2023

Stephen G. Ware and Rachelyn Farrell
Narrative Intelligence Lab
Department of Computer Science
University of Kentucky

Abstract

We have compiled a collection of story planning problems from several authors to test the Sabre narrative planner. This report identifies the origins of these problems, describes the modifications we made translating them into Sabre's syntax, and gives recommended settings when searching for solutions.

Contents

Introduction

Storytelling algorithms have existed since at least Meehan's seminal 1977 paper on *Tale-Spin* [9], which generated short stories similar to Aesop's Fables. These algorithms can be classified according to several features [6]. For this report, we focus on story planning algorithms. Planning algorithms perform a search to determine whether a story meeting certain requirements is possible. Story *planning* algorithms can be contrasted with *reactive* algorithms (like ABL [8] and other descendants of the Oz Project [7]), which decide on the next story event based on the current state and perform little or no lookahead when making their decisions. Reactive storytelling systems scale well but produce emergent stories that may be hard to control. Planning systems are expensive but can guarantee constraints on a story's content and structure.

Young [24] was perhaps the first to suggest that AI planning algorithms would be suitable for interactive storytelling in computer systems. A planning algorithm is a symbolic, logical system that takes as input (1) a description of the initial state, (2) a goal, and (3) a set of possible actions which have preconditions that must hold before they occur and effects which modify the world state. A planning algorithm, or *planner*, searches for a plan—a sequence of actions executable in the initial state that achieves the goal [23]. Young observed that planners provide a formal, generative model of action and can reason about narratively important properties like causality.

Planners for Narrative vs. Narrative Planners

To oversimplify the research landscape, there are two broad approaches to planning stories. The first uses off-the-shelf planners for storytelling purposes. Their advantage is that advances in planning research can immediately be leveraged for storytelling. Their disadvantage is that all narrative reasoning must be encoded into the preconditions and effects of actions, or into other features provided by a planner that was not designed for storytelling. This approach is often used by Marc Cavazza and his academic descendants [1, 10].

The second approach modifies planning algorithms to reason directly about narrative phenomena. In this approach, the preconditions and effects of actions are limited to the most basic constraints; other features of the planner decide when they should occur based on explicit narrative reasoning. Their advantage is a richer set of storytelling features in the algorithm; their disadvantage is that it becomes difficult to integrate advances in planning research into these systems. This approach is often used by R. Michael Young and his academic descendants [25].

About Sabre

The Sabre narrative planner [19] arose from the second school of thought. Though none of its individual features is unique, it is the first algorithm with this particular set of features:

- *Intentionality:* A character can only take an action if they believe it will further their goals.
- *Conflict:* Characters can thwart one another's plans, and stories can contain failed or partially-executed character plans.
- *Arbitrary Theory of Mind:* It reasons about what is actually true, what character x believes, what x believes character y believes, what x believes y believes z believes, and so on, arbitrarily.
- *No Uncertainty:* Beliefs can be wrong, but characters always commit to their beliefs. They can believe p when $\neg p$ is the case, but they cannot believe $p \vee q$. This limitation increases the size of the problems Sabre can solve at the cost of a feature some might find desirable.

This set of features was chosen based on the needs of the interactive narratives we are developing at the Narrative Intelligence Lab at the University of Kentucky. We make no claim that they are the most desirable in general.

Sabre begins at the initial state of a storytelling problem and performs a forward heuristic search through the space of states until it finds a state where the author's goal for the story is achieved. Every action taken on the way to that goal must be *explained* for each of the characters[1] who takes the action (called the *consenting characters*). An action is *explained for a character* when that character can imagine a plan that starts with the action and which, according to that character's beliefs, will lead to a better state for that character and contains only explained actions. Sabre should be cited as follows:

> Stephen G. Ware and Cory Siler. Sabre: A narrative planner supporting intention and deep theory of mind. In *Proceedings of the 17th AAAI conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 99–106, 2021

[1] Sabre uses the term *character*, instead of *agent*, because they are not independent decision-makers. The planner is the only agent. It acts like a puppetmaster, omniscient but attempting to make each character seem like it has its own goals and limited, possibly wrong beliefs.

About this Collection

Sabre provides several search methods and heuristics. To test them, we have written several problems. We also want to compare Sabre's performance on problems written by others, so we have collected what problems we can from the narrative planning literature. Many of these problems were created for other planning systems, so we have had to translate them into Sabre's syntax, a process which has certainly introduced some bias.

In an attempt to partially mitigate that bias, the next section details the process we followed when translating problems for testing in Sabre. The third section gives details about each problem and explains their origins.

It is our hope that this collection of problems will serve as a repository of benchmark problems for those interested in narrative planning. However, we also want to offer a word of caution learned from the AI planning community. Performance on this suite of problems should not be considered an authoritative measure of whether a planner is state-of-the-art. A planner should not be judged uninteresting because it fails to perform well on these tests. In other words, we want these benchmarks to encourage comparison across planners without limiting the richness of narrative planning research.

This report can be cited as follows:

> Stephen G. Ware and Rachelyn Farrell. A collection of benchmark problems for the Sabre narrative planner. Technical report, Narrative Intelligence Lab, University of Kentucky, November 2023

Methodology

While some of the problems in this collection were initially created for Sabre, many were originally made for other systems with different features[2]. We want to show that Sabre can generalize enough to solve problems established by other systems, but we recognize that translating these problems has changed them. This section explains the guidelines we followed when translating problems.

Spirit Over Structure

Our translations have attempted to maintain the spirit of a problem as we understood it rather than its exact formal structure. The goal was to write the

[2]Most problems we translated were originally written in PDDL, the Planning Domain Definition Language [4], or something similar to it.

problem as if the original author had designed it for Sabre from the beginning. For example, many early planning systems use only Boolean predicates, whereas Sabre supports multi-valued fluents. We could have "directly" translated these problems by using only Boolean fluents in Sabre, but this makes the problem artificially harder. Consider representing a character's location when there are $c = \{c_1, c_2, ...\}$ characters and $p = \{p_1, p_2, ...\}$ places. A Boolean fluent like $location(c_1, p_1)$ means "character c is at place p," and we need $|c| \cdot |p|$ of these fluents, exactly one of which is true in any state. Sabre can use a single fluent $location(c)$ which can have any place as its value. When possible, we used multi-valued fluents like this, even though it results in a less exact translation of the problem.

Fixing Bugs

Some problems had mistakes which we have fixed. In some cases, we contacted the original authors (or we were the original authors), so we are confident in these changes. Some were judgment calls based on our understanding of the spirit of the problem. For example, one of the key events in the *Hospital* problem is a stressed doctor misdiagnosing a patient. The effects of the `mistake-assess` action never appear as preconditions of any other action or in the problem goal; we believe this was a mistake in the original problem, and we have modified it to allow a mistaken diagnosis to lead to a patient's death. Similar bugs have been fixed in other problems and are noted in the details for that problem in this report.

Goals as Utility

Most planners use propositions to define the goals of a planning problem. Sabre uses utility functions—numeric expressions to be maximized. Sabre defines an author utility, which the solution plan itself should maximize, and one character utility function per character, which those characters each try to maximize. Some narrative planners do not represent character goals at all. Some do, but allow characters to adopt and drop their goals during a plan. Sabre's utility functions cannot be changed during a plan, though their values can be conditional. For all of these reasons, translating the goals of problems into Sabre utility functions required some significant interpretation. Consider the *Western* problem, where a character can be bitten by a snake, causing everyone who loves that character to adopt the goal of healing the snakebite. When it is healed, the goal is dropped. We represent this in Sabre using numeric fluents like $relationship(c_1, c_2)$. When c_1 loves c_2 and c_2 is healthy, the value of this fluent is 1. If c_1 does not love c_2, or if c_2 is sick, the value is 0. Each character's relationships contribute to their utility

function. When c_1 loves c_2 and c_2 is bitten by a snake, c_1's utility decreases by 1, and any plan which restores c_2's health raises c_1's utility by 1.

One Problem, Few Versions

The benchmark problems used in the AI planning community typically distinguish between a *domain* and a *problem*. A domain defines the actions while a problem defines a set of objects, an initial state, and a goal. There are typically many problems per domain. There seems to be less of a need for this distinction in storytelling systems. Many storytelling domains have only one associated problem, or a few very similar problems. There is less variation in their initial states and goals. Sabre defines everything—actions, objects, initial state, and goals—in the same unit, called a problem. This does not mean a Sabre problem can tell only a single story. Complex utility functions and disjunctive goals can give rise to many stories. Consider the *Save Gramma* problem, where Tom represents the player character in an interactive narrative game. The author's utility is 0 in the initial state, 1 when Tom is dead, and 2 when Tom achieves his goal of returning home with medicine for his grandmother. There are many ways for Tom to die and for Tom to complete his goal, so a wide variety of stories can still be told when solving this problem.

We can represent several versions of a Sabre problem (what other planning researchers might call many problems in the same domain) by setting a higher goal for the author utility. In *Save Gramma*, a goal of reaching author utility 1 means any plan that ends in either Tom's death or the successful completion of Tom's quest can be accepted as a solution. This corresponds to the benchmark problem `gramma_any`. By setting a goal of author utility 2, only solutions where Tom completes his quest are accepted. This corresponds to the benchmark problem `gramma_win`.

When a problem's goal has multiple parts, we typically made each of these parts contribute independently to the author's utility. This allows the same Sabre problem to be used in many benchmark tasks simply by settings a higher goal for the author's utility. Consider *Aladdin*. The original problem has a goal with two parts: Jasmine is married to Jafar and the Genie is dead. In our version of the problem, the author gains 1 point of utility for each of these. The benchmark problem `aladdin_any` can be solved by any story that marries Jasmine to Jafar or kills the genie, whereas both goals must be achieved in `aladdin_both`.

Some problems, like *Hospital*, *Basketball*, and *Lovers*, were designed as domains with multiple problems in mind. In those cases, we have chosen a single problem to implement, usually the smallest and simplest. This reduces the richness and variety of the original problems, but our objective with this suite of benchmarks

is to test many types of stories rather than many stories of the same type.

Belief

Sabre tracks what is actually true, what each agent believes, what they believe others believe, and so on arbitrarily. Sabre provides three ways that beliefs are updated. First, each action has an *observation function* which defines when a character notices an action occur. When a character observes an action, they update their beliefs (and their beliefs about the beliefs of others who observed it, etc.). Sabre also provides *triggers*, which are like actions except they must occur when they can. Triggers are frequently used for belief updates. For example, one commonly used trigger can be read as "When character c_1 is at the same location as character c_2, but c_1 believes c_2 is somewhere else, c_1 now realizes that c_2 is at their location." Finally, Sabre actions can update character beliefs directly. The `report` action in *Save Gramma*, where one character tells the guard they have seen the bandit, and the `tell` action in *Lovers*, where one character tells another what item they want, are two actions with examples of direct belief updates.

When translating problems that did not originally model beliefs, we tried to use semantics informed by the stories being modeled. Most problems have some concept of a location, so generally characters observe any action that happens at their location. Most problems include triggers that allow characters to observe the people, places, and things, at their location.

Some problems originally designed for planners that do no model belief occasionally used predicates that clearly represented beliefs. For example, the `knows-location` predicate in the original version of *Raiders of the Lost Ask* has been removed and replaced with Sabre's model of belief about a location.

Any wrong beliefs that characters have in the initial state must be specified; otherwise Sabre assumes their beliefs reflect the real world. For most problems that did not originally model belief we assumed no wrong beliefs, unless they were obvious from the semantics, such as the `knows-location` predicate above.

Intention

Intentionality is the tendency of intelligent agents to act in service of their own goals, or in Sabre's case, to raise their own utilities. Each Sabre action has a list of characters, called the *consenting characters*, who must have a reason to take the action. A Sabre plan can only contain an action if it is explained for all its consenting characters. An action is explained for a character if that action is the first in a sequence of actions which that character believes they can take, will result in a higher utility for them, and is composed only of actions which that

character believes are also explained for all their consenting characters. Note that intention is linked to belief—characters can take actions they think will help, but wrong beliefs may cause their actions to go awry. In *Raiders of the Lost Ark*, the Nazis believe opening the Ark of the Covenant will grant them immortality, so they open it; in reality, it kills them.

An action with no consenting characters is called an author-only action and represents a *deux ex machina*—an action that the author can choose for the sole reason that it advances the plot. Characters cannot expect them to happen.

When translating problems that did not originally model intention, we used the semantics of the verbs the actions modeled. Typically any characters mentioned in an action were made consenting characters, with some obvious exceptions. For example, in *Basketball*, the thief is the only consenting character for `steal`; clearly this action is reasonable even when the victim does not want it.

Guided by Exemplar Stories

Problems generally had known solutions provided by the original author. While translating problems, we ensured these example solutions, or something very similar in spirit, could be produced by the Sabre version of the problem. These exemplar solutions are listed in the section for each problem, but this does not imply that they are the only solutions that exist, or even the best solutions.

Occasionally, ensuring these solutions were possible required us to violate the guidelines. For example, in *Western* we assume that characters only observe actions that occur at their location. This means the town sheriff (who starts at the saloon) will not notice Hank's robbery of the general store. To ensure the exemplar solution was still possible, we modified the `take` action so the sheriff always observes it no matter his location.

Occasionally it was impossible to ensure exemplar solutions were possible due to Sabre's minimality constraint. A plan is minimal if no actions can be left out without lowering the utility it achieves or causing other actions to become unexplained. Sabre requires plans for characters and the author to be minimal[3], but some exemplar stories were not minimal. The original *Space* story is about a misunderstanding between a starship captain and an alien. They battle to a stalemate and flee when a volcano starts to erupt. The battle sequence can be left out without changing the outcome, so even though that sequence of actions exists for Sabre and is explained, it is not considered a solution.

[3] A minimal plan does not need to be the shortest possible plan, but it cannot contain any steps that could be left out. In other words, if some subsequence of a plan is also explained and achieves the same utility, that subsequence is the solution, not the longer plan.

Just as we designed problems to produce ideal solutions, we sometimes modified them to avoid producing problematic ones, even if they were valid solutions to the original problem. In the original *Western*, after being bitten by a snake, Timmy could tie up his father, drag him to the jailhouse, then walk to the general store to steal the antivenom himself. His path via the jailhouse, and his dragging a prisoner around with him, are an unnecessarily long yet still valid and minimal path to the general store. It also happens to be a convenient way to achieve the author's goal. We have modified the problem to rule out this strange solution.

Recommended Search Settings

There are three important limits that can be imposed on Sabre's search. The *author temporal limit* is the maximum number of actions in the author's plan—that is, the actual actions that will be executed to raise the author's utility. The *character temporal limit* is the maximum number of actions in a plan a character imagines when justifying an action. Consider this solution to the *Save Gramma* problem:

Tom walks to the crossroads.
The bandit walks to the crossroads.
The bandit kills Tom.

It can be found with an author temporal limit of 3, since the plan contains only 3 actions, but it cannot be found unless the character temporal limit is higher. This is because Tom needs to imagine a longer plan to justify why he would want to walk to the crossroads:

Tom walks to the crossroads.
Tom walks to the market.
Tom buys the medicine from the merchant.
Tom walks to the crossroads.
Tom walks to the cottage.

Only the first action in Tom's plan actually happens in the story before he is interrupted by the bandit. Still, he has to have a reason to walk to the crossroads before he can act, and the planner cannot find this reason unless the character temporal limit is set to 5 or higher.

The third limit is the *epistemic limit*, which constrains how deeply Sabre will search into a character's theory of mind. It is important to note that Sabre always has an infinitely deep theory of mind, and beliefs are always modeled the same way regardless of this limit. The epistemic limit only affects which states will be

explored by the search. In other words, setting this limit to 0 does not remove or alter Sabre's theory of mind; it only limits which solutions can be found. It is also important to note that Sabre can still *reason* deeper than this limit without necessarily *searching* deeper than this limit. Tom's plan to buy the medicine from the merchant requires reasoning 2 layers deep, because Tom has to anticipate that the merchant will sell him the medicine. Sabre has to reason about what Tom believes the merchant believes (2 layers). However, the solution can still be found with an epistemic limit of 1. The state that represents the merchant's beliefs after buying the medicine is generated even though it is never visited. Not visited means Sabre never considers building longer plans starting in that state, but because the merchant's utility is instantly improved in that state, Sabre does not need a longer plan to recognize it as good for the merchant. An epistemic limit of 1 is sufficient to find this explanation, even though it reasons about a belief at layer 2.

For each problem, we list recommended settings for the author temporal limit, character temporal limit, and epistemic limit. These limits allow all of the exemplar solutions described for that problem. For some problems, we recommend a higher epistemic limit than what is strictly required to produced the example solutions because those problems allow interesting behavior that can only happen at higher epistemic limits. For example, in Save Gramma, characters will never report the bandit's location to the guard unless they can imagine the guard chasing after the bandit, so while the exemplar solutions can be found with an epistemic limit of 1, we recommend 2 for that problem to allow actions like `report` to be explored.

Problems

This section explains the origin of each benchmark problem, mentions any important details about how we translated it, and recommends search settings for using the problem as a benchmark in Sabre.

Bribery

This is the first of four problems detailed in the appendix of Mark O. Riedl's dissertation. His dissertation is one of the seminal works on narrative planning that established the model of intention used in several future planners.

> Mark O. Riedl. *Narrative planning: Balancing plot and character.* PhD thesis, North Carolina State University, 2004

The problem is small and describes how a villain gains control over the president using a bribe. It has three characters: the villain, the president, and a hero who

the villain can manipulate into robbing a bank to obtain the money needed for the bribe. The two solutions to this problem were originally meant to highlight a feature of Riedl's Fabulist system [13] that allowed characters to act contrary to their personalities if given sufficient motivation. In this problem, the usually lawful hero can rob a bank if they are first coerced by the villain.

Riedl's planner models intention but not belief. It also has a feature that allows one character to delegate its goals to another, which was used in the `coerce` action. There is no easy way to translate this notion of goal delegation into Sabre's utility functions. To maintain the minimality and simplicity of the original problem, we have modified the `coerce` action to cause the coerced character to want an item to be at a specific location. In the example below, `coerce(Villain, Hero, Money)` makes the hero want the villain to have the money.

Solution 1: The villain robs the bank themselves.

```
steal(Villain, Money, Bank)
bribe(Villain, President, Money)
```

Solution 2: The villain coerces the hero into robbing the bank.

```
threaten(Villain, Hero)
coerce(Villain, Hero, Money)
steal(Hero, Money, Bank)
give(Hero, Villain, Money)
bribe(Villain, President, Money)
```

Version	Goal Utility
`bribery`	1

Setting	Value
Author Temporal Limit	5
Character Temporal Limit	5
Epistemic Limit	2

Deer Hunter

This is the second of four problems detailed in the appendix of Riedl's dissertation.

> Mark O. Riedl. *Narrative planning: Balancing plot and character.* PhD thesis, North Carolina State University, 2004

The problem illustrates how actions can be reused for multiple goals. The protagonist, Bubba, owns a gun and some ammunition. He can become greedy and use his gun to rob a bank. He can also become hungry and use his gun to hunt a deer.

Riedl's planner models intention but not belief. In the original problem, author-only actions caused Bubba to adopt the goal of having money or not being hungry. This allowed the planner to choose either or both goals for the story. We have preserved that structure by changing the `decide_to_get_money` action to increase Bubba's greed (allowing him to raise his utility by stealing money) and the `decide_to_eat` action to make Bubba hungry (lowering his utility, and allowing him to raise it again by eating). The author gains 1 point of utility when Bubba gets money and 1 point when the deer is eaten.

Solution 1: Bubba robs a bank.
(Shortest known plan for author utility 1)

```
decide_to_get_money(Bubba)
pickup(Bubba, Rifle, House)
pickup(Bubba, Ammo, House)
load(Bubba, Rifle, Ammo)
go(Bubba, House, Bank)
steal(Bubba, Clerk, Rifle, Bank)
```

Solution 2: Bubba hunts a deer.
(Alternative solution for author utility 1)

```
decide_to_eat(Bubba)
pickup(Bubba, Rifle, House)
pickup(Bubba, Ammo, House)
load(Bubba, Rifle, Ammo)
go(Bubba, House, Forest)
shoot(Bubba, Bambi, Rifle, Forest)
eat(Bubba, Bambi, Forest)
```

Solution 3: Bubba does both.
(Shortest known plan for author utility 2)

```
decide_to_get_money(Bubba)
decide_to_eat(Bubba)
pickup(Bubba, Rifle, House)
pickup(Bubba, Ammo, House)
load(Bubba, Rifle, Ammo)
go(Bubba, House, Bank)
steal(Bubba, Clerk, Rifle, Bank)
go(Bubba, Bank, Forest)
shoot(Bubba, Bambi, Rifle, Forest)
eat(Bubba, Bambi, Forest)
```

Version	Goal Utility
deerhunter_any	1
deerhunter_both	2

Setting	Value
Author Temporal Limit	10
Character Temporal Limit	6
Epistemic Limit	1

Secret Agent

This is the third of four problems detailed in the appendix of Riedl's dissertation.

> Mark O. Riedl. *Narrative planning: Balancing plot and character.* PhD thesis, North Carolina State University, 2004

The problem requires a secret agent to enter the compound of an evil mastermind and assassinate him. To enter, he must be unarmed, but to assassinate the mastermind he needs a gun, which he can find in the compound.

Riedl's planner models intention but not belief. It also has a feature that allows the planner to decide some parts of the initial state that the author intentionally left unspecified. In this example, the location of the gun was unspecified, and the planner could place it at any location. We have preserved this feature by adding the `find` action, which allows the agent to find an item at their current location as long as the item's location has not yet been set. Note the `find` action has the secret agent as a consenting character. It cannot be an author-only action; if so the agent would not believe it is possible to find a gun after entering the compound and would never act.

Solution 1: The secret agent assassinates the mastermind.

```
move(SecretAgent, Headquarters, Dropbox)
pickup(SecretAgent, Papers, Dropbox)
move(SecretAgent, Dropbox, Courtyard)
move(SecretAgent, Courtyard, Lobby)
find(SecretAgent, Gun, Lobby)
pickup(SecretAgent, Gun, Lobby)
move(SecretAgent, Lobby, Office)
kill(SecretAgent, Mastermind, Gun, Office)
```

Version	Goal Utility
secretagent	1

Setting	Value
Author Temporal Limit	8
Character Temporal Limit	8
Epistemic Limit	1

Aladdin

This is the fourth and largest of four problems detailed in the appendix of Riedl's dissertation. It was also detailed in a journal article about Riedl's IPOCL planner, so we suggest using that citation when referring to this problem.

> Mark O. Riedl and R. Michael Young. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39(1):217–268, 2010

The problem is an alternative version of the tale of *Aladdin*, which was added to a later edition of *One Thousand and One Nights* and later adapted into a 1992 Disney film. Many elements have been changed. Jafar is king, not adviser to the king. A dragon has been added. The end of the story has been changed. These modifications were likely due to the fact that Riedl's planner did not support conflict, the ability for an agent to partially execute a plan which then fails. Support for conflict was added in a later algorithm by Ware et al. [22] which enabled new solutions to this problem.

Riedl's and Young's planner models intention but not belief. It also has a feature that allows one character to delegate its goals to another, which was used in the original `order` and `command` actions. There is no easy way to translate this notion of goal delegation into Sabre's utility functions. To maintain their spirit, we have replaced them with the `command_kill`, `command_love`, and `command_bring` actions, which model a requester assigning a task to a worker. The tasks require the worker to kill a character, bring an item, or cause another character to love the requester respectively. Characters gain utility by completing tasks.

The original problem goal has two parts: Jasmine should be married to Jafar and the genie should be dead. The author gains 1 point of utility for each, allowing two versions of this problem.

Solution 1: Aladdin is frightened by the genie and slays it.
(Shortest known plan for author utility 1)

```
fall_in_love(Aladdin, Jasmine, Castle)
travel(Aladdin, Castle, Mountain)
slay(Aladdin, Dragon, Mountain)
pillage(Aladdin, Dragon, Lamp, Mountain)
summon(Aladdin, Genie, Lamp, Mountain)
appear_threatening(Genie, Aladdin, Mountain)
slay(Aladdin, Genie, Mountain)
```

Solution 2: Jafar uses a love spell to marry Jasmine.
(Alternative solution for author utility 1)

```
fall_in_love(Jafar, Jasmine, Castle)
travel(Jafar, Castle, Mountain)
slay(Jafar, Dragon, Mountain)
pillage(Jafar, Dragon, Lamp, Mountain)
travel(Jafar, Mountain, Castle)
summon(Jafar, Genie, Lamp, Castle)
command_love(Jafar, Genie, Jasmine, Castle)
love_spell(Genie, Jasmine, Jafar)
marry(Jafar, Jasmine, Castle)
```

Solution 3: Jafar marries Jasmine and Aladdin slays the genie.
(Shortest known plan for author utility 2)

```
fall_in_love(Jafar, Jasmine, Castle)
travel(Jafar, Castle, Mountain)
slay(Jafar, Dragon, Mountain)
pillage(Jafar, Dragon, Lamp, Mountain)
travel(Jafar, Mountain, Castle)
summon(Jafar, Genie, Lamp, Castle)
command_love(Jafar, Genie, Jasmine, Castle)
love_spell(Genie, Jasmine, Jafar)
marry(Jafar, Jasmine, Castle)
appear_threatening(Genie, Aladdin, Castle)
slay(Aladdin, Genie, Castle)
```

Solution 4: Story used by Riedl and Young to evaluate IPOCL [14].
(Alternative solution for author utility 2)

```
fall_in_love(Jafar, Jasmine, Castle)
command_bring(Jafar, Aladdin, Lamp, Castle)
travel(Aladdin, Castle, Mountain)
slay(Aladdin, Dragon, Mountain)
pillage(Aladdin, Dragon, Lamp, Mountain)
travel(Aladdin, Mountain, Castle)
give(Aladdin, Jafar, Lamp, Castle)
summon(Jafar, Genie, Lamp, Castle)
command_love(Jafar, Genie, Jasmine, Castle)
love_spell(Genie, Jasmine, Jafar)
marry(Jafar, Jasmine, Castle)
appear_threatening(Genie, Aladdin, Castle)
slay(Aladdin, Genie, Castle)
```

Version	Goal Utility
aladdin_any	1
aladdin_both	2

Setting	Value
Author Temporal Limit	13
Character Temporal Limit	10
Epistemic Limit	2

Hospital

This problem was first used by a storytelling system called *NetworkING*, which focuses on the relationships between characters and how they change as a driving force for narrative generation.

> Julie Porteous, Fred Charles, and Marc Cavazza. NetworkING: using character relationships for interactive narrative generation. In *Proceedings of the international conference on Autonomous Agents and Multi-Agent Systems*, pages 595–602, 2013

The problem describes a medical drama where hospital staff navigate fraught relationships as they treat patients. A version of the problem was provided by Julie Porteous on her website.

> https://porteousjulie.bitbucket.io/

The version on her website is simpler than the one described in the paper. Actions like `spread-malicious-gossip` and `show-appreciation-treatment-advice` are missing, with this simpler version focusing on a stressed doctor who can make a mistake when treating too many patients.

Porteous et al. use an off-the-shelf planner which does not reason about intention or belief. We have added these concepts based on our understanding of medical dramas. Because Sabre models belief, we were able to combine the `assess` and `mistake-asses` action into a single action. The doctor can diagnose a patient with the wrong illness if they are overworked. The effects of our `assess` action are that the doctor and patient believe the patient has a certain illness, but these beliefs can be wrong.

We have also fixed what we suppose to be two bugs. In the original, the effects of the `mistake-assess` action were never used in the preconditions of other actions or in the goal. Also, the preconditions of the `die` and `recover` action were not constrained by the patient's diagnosis. This meant patients who got the correct treatment could still die and those who got the wrong treatment could still recover. This may have been intended, but we modified the problem so that patients who get the correct treatment always recover and those who get the wrong treatment always die. In doing this, we removed the `die` and `recover` actions and added a conditional effect to `treat`.

The original problem was specified as a PDDL domain with 10 associated problems. We used the first problem in our translation, which has a single doctor and 3 patients. Because they are not needed in that problem, we removed the `move`, `get-work-help`, and `lose-consciousness` actions from our version. Each of the original 10 problems had different goals. In our version, the author gains 1 point of utility if some patient has been healed and 1 point if some patient has died.

Solution 1: Dr. Hathaway treats a single patient.
(Shortest known plan for author utility 1)

```
admit(Hathaway, Jones, PatientRoomA)
walk(Jones, Admissions, PatientRoomA)
assess(Hathaway, Jones, SymptomA, PatientRoomA)
treat(Hathaway, Jones, TreatmentA, PatientRoomA)
```

Solution 2: Dr. Hathaway admits too many patients and misdiagnoses Jones. (Alternative solution for author utility 1)

```
admit(Hathaway, Jones, PatientRoomA)
admit(Hathaway, Ross, PatientRoomB)
admit(Hathaway, Young, PatientRoomC)
walk(Jones, Admissions, PatientRoomA)
assess(Hathaway, Jones, SymptomB, PatientRoomA)
treat(Hathaway, Jones, TreatmentB, PatientRoomA)
```

Solution 3: Dr. Hathaway makes one incorrect and one correct diagnosis. (Shortest known plan for author utility 2)

```
admit(Hathaway, Jones, PatientRoomA)
admit(Hathaway, Ross, PatientRoomB)
admit(Hathaway, Young, PatientRoomC)
walk(Jones, Admissions, PatientRoomA)
assess(Hathaway, Jones, SymptomB, PatientRoomA)
treat(Hathaway, Jones, TreatmentB, PatientRoomA)
walk(Ross, Admissions, PatientRoomA)
walk(Ross, PatientRoomA, PatientRoomB)
walk(Hathaway, PatientRoomA, PatientRoomB)
assess(Hathaway, Ross, SymptomA, PatientRoomB)
treat(Hathaway, Ross, TreatmentA, PatientRoomB)
```

Version	Goal Utility
hospital_any	1
hospital_both	2

Setting	Value
Author Temporal Limit	11
Character Temporal Limit	5
Epistemic Limit	3

Basketball

This problem was introduced by Kartal et al. in an experiment that used Monte Carlo Tree Search and a model of believable actions for planning-based story generation.

> Bilal Kartal, John Koenig, and Stephen J Guy. User-driven narrative variation in large story domains using Monte Carlo Tree Search. In *Proceedings of the international conference on Autonomous Agents and Multiagent Systems*, pages 69–76, 2014

It describes a crime drama where characters can commit theft and murder while investigators search for clues and arrest criminals. Characters get angry when their items are stolen but can let off steam by playing basketball. The version of the problem we translated was provided by Julie Porteous on her website.

`https://porteousjulie.bitbucket.io/`

The version on her website differs slightly from the one described in the paper. The `earthquake` action is missing, and some additional concepts are present, like vehicles. This PDDL domain was provided with 10 different problems whose objects and initial states are similar but whose goals specify a variety of outcomes, like what crimes should be committed, who should be angry, who should be arrested, etc. We used a small set of objects: three civilians, one detective, three locations, and two items. We did not include `earthquake`, since this was meant to be an unbelievable action, and we did not include vehicles. Though we implemented the `shareClues` action, there is only one inspector, so this action will never occur.

The original problem was created for a traditional planner, so it did not include character goals. We gave each of the citizens different utility functions: Alice starts angry and wants to be calm. Bob is calm and wants everyone to be calm. Charlie starts angry and wants Alice to be dead. The author gains 1 point of utility if fewer than two citizens are angry and 1 point if someone has been arrested.

The original problem also did not model belief, but the `knows-clue` predicate and `findClues` action have obvious belief semantics. The investigator gains 1 point of utility for every location they investigate and every criminal they arrest.

Solution 1: Bob calms Alice by playing basketball with her.
(Shortest known plan for author utility 1)

```
travel(Bob, HomeB, BasketballCourt)
travel(Alice, Downtown, BasketballCourt)
play_basketball(Bob, Alice, BasketballCourt)
```

Solution 2: Sherlock arrests Charlie for murder.
(Alternative solution for author utility 1)

```
kill(Charlie, Alice, Bat, Downtown)
find_clues(Sherlock, Murder, Bat, Downtown)
suspect_of_crime(Sherlock, Charlie, Murder, Bat, Downtown)
arrest(Sherlock, Charlie, Downtown, Murder)
```

Solution 3: Alice is calmed by basketball but arrested for theft.
(Shortest known plan for author utility 2)

```
travel(Bob, HomeB, BasketballCourt)
travel(Alice, Downtown, BasketballCourt)
travel(Sherlock, Downtown, BasketballCourt)
steal(Alice, Bob, Basketball, BasketballCourt)
play_basketball(Alice, Bob, BasketballCourt)
suspect_of_crime(Sherlock, Alice, Theft, Basketball,
BasketballCourt)
arrest(Sherlock, Alice, BasketballCourt, Theft)
```

Version	Goal Utility
basketball_any	1
basketball_both	2

Setting	Value
Author Temporal Limit	7
Character Temporal Limit	5
Epistemic Limit	2

Western

This is the first of three narrative planning problems used by Ware et al. to study perceptions of conflict in automatically generated stories.

> Stephen G. Ware, R. Michael Young, Brent Harrison, and David L. Roberts. A computational model of narrative conflict at the fabula level. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games*, 6(3):271–288, 2014

The problem is detailed in the appendix of Ware's dissertation.

> Stephen G. Ware. *A plan-based model of conflict for narrative reasoning and generation.* PhD thesis, North Carolina State University, 2014

It tells the story of rancher Hank in the American old West whose son Timmy is bitten by a snake. Hank robs the town general store to get some antivenom but is then arrested by Will, the town sheriff.

This planning domain and its one associated problem were created for Ware's and Young's Glaive narrative planner [20] which was designed to allow conflict. The story cannot be told unless characters form plans that are thwarted before they are complete. For example, Hank's plan to steal the antivenom is thwarted by the sheriff.

Glaive supports intention but not belief. We used a default model of belief where characters observe the actions that happen at their location, with one exception. The sheriff needs to observe all instances of the `take` action or else he will only be aware of crimes committed at his location. We also fixed a bug in the initial state of this problem where no sheriff had been specified.

In the original problem, the `travel` action specified where the traveler was leaving from and where they were going to. The `force-travel` action moved both the traveler and a helpless prisoner. Since moving alone and moving with a prisoner both required one action, and since the travel actions specified their origin, it was possible for characters to take meandering routes to their destinations. For example, once bitten by a snake, Timmy could tie up his father, move both of them to the jailhouse, and them travel by himself to the general store to steal the antivenom. This path is minimal in the sense that no actions can be left out, but it is strange. We have modified the `travel` and `force-travel` actions so they no longer include their origin. We have also changed the `force-travel` action to move only the prisoner. This avoids unnecessarily long paths that are a convenient but strange way to achieve the author's goal.

Like IPOCL [14], Glaive allows characters to adopt and drop goals. We translated these concepts into utility functions by using relationships. Every character has a numeric relationship, including a relationship with themselves. When someone a character loves is healthy, the relationship has a value of 2. When someone the character loves is sick, the relationship has a value of 1. Otherwise, relationships have a value of 0. All characters love themselves. Thus, when a character is bitten by a snake, their utility goes down, as does the utility of everyone who loves them. That utility can be restored by healing the sick loved one, or permanently lost if the loved one dies. The author's utility is 1 if Timmy is dead and Hank is tied up in the jailhouse, otherwise the author utility is 0.

Solution 1: Story used in the study by Ware et al. [22].

```
snakebite(Timmy, Ranch)
travel(Hank, GeneralStore)
tie_up(Hank, Carl, GeneralStore)
take(Hank, Antivenom, Carl, GeneralStore)
travel(Will, GeneralStore)
tie_up(Will, Hank, GeneralStore)
force_travel(Will, Hank, Jailhouse)
die(Timmy, Snakebite, Ranch)
```

Version	Goal Utility
`western`	1

Setting	Value
Author Temporal Limit	8
Character Temporal Limit	5
Epistemic Limit	1

Fantasy

This is the second of three narrative planning problems used by Ware et al. to study perceptions of conflict in automatically generated stories.

> Stephen G. Ware, R. Michael Young, Brent Harrison, and David L. Roberts. A computational model of narrative conflict at the fabula level. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games*, 6(3):271–288, 2014

The problem is detailed in the appendix of Ware's dissertation.

> Stephen G. Ware. *A plan-based model of conflict for narrative reasoning and generation.* PhD thesis, North Carolina State University, 2014

It tells a Medieval story of princess Talia and two potential suitors: the poor Rory whom she loves and the rich Vince whom she does not love. There is also a nearby dragon who has treasure that can be stolen but is on the lookout to increase her hoard.

This planning domain and its one associated problem were created for Ware's and Young's Glaive narrative planner [20]. Glaive supports intention but not belief. We used a default model of belief where characters observe the actions that happen at their location.

The author's utility is based on Talia's happiness. She wants to be married to someone she loves and she wants to be rich—the wealthier the better. Each character has a different utility function to add some variety to the possible stories. Talia wants to be happily married and wealthy; she does not care about hunger. Rory wants to be happily married, fed, and rich. Vince wants to be happily married and fed; he does not care about money. The dragon wants to be wealthy and fed; she does not care about marriage.

Solution 1: Talia gets rich.
(Shortest known plan for author utility 1)

```
travel(Talia, Village, Cave)
pickup(Talia, Treasure, Cave)
```

Solution 2: Talia marries for love.
(Alternative solution for author utility 1)

```
propose(Rory, Talia, Village)
accept(Talia, Rory, Village)
marry(Rory, Talia, Village)
```

Solution 3: Talia marries for money.
(Alternative solution for author utility 1)

```
propose(Vince, Talia, Village)
accept(Talia, Vince, Village)
marry(Vince, Talia, Village)
```

Solution 4: Talia finds love and wealth.
(Shortest known plan for author utility 2)

```
propose(Rory, Talia, Village)
accept(Talia, Rory, Village)
marry(Rory, Talia, Village)
travel(Talia, Village, Cave)
pickup(Talia, Treasure, Cave)
```

Solution 5: Story used in the study by Ware et al. [22].
(Alternative plan for author utility 2)

```
propose(Rory, Talia, Village)
accept(Talia, Rory, Village)
travel(Rory, Village, Cave)
steal(Rory, Gargax, Treasure, Cave)
travel(Rory, Cave, Village)
marry(Rory, Talia, Village)
```

Solution 6: Talia has it all.
(Shortest known plan for author utility 3)

```
propose(Rory, Talia, Village)
accept(Talia, Rory, Village)
marry(Rory, Talia, Village)
get_hungry(Gargax)
travel(Gargax, Cave, Village)
eat(Gargax, Vince, Village)
take(Talia, Money, Vince, Village)
travel(Talia, Village, Cave)
pickup(Talia, Treasure, Cave)
```

Version	Goal Utility
`fantasy_any`	1
`fantasy_two`	2
`fantasy_all`	3

Setting	Value
Author Temporal Limit	9
Character Temporal Limit	3
Epistemic Limit	2

Space

This is the third of three narrative planning problems used by Ware et al. to study perceptions of conflict in automatically generated stories.

> Stephen G. Ware, R. Michael Young, Brent Harrison, and David L. Roberts. A computational model of narrative conflict at the fabula level. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games*, 6(3):271–288, 2014

The problem is detailed in the appendix of Ware's dissertation.

> Stephen G. Ware. *A plan-based model of conflict for narrative reasoning and generation.* PhD thesis, North Carolina State University, 2014

It tells the story of starship captain Zoe who wants to befriend the reptilian guardian of an alien planet which features a volcano that might erupt at any time. Zoe does not know that the guardian will get angry if she walks on the planet's surface, and conflict may ensue.

This planning domain and its one associated problem were created for Ware's and Young's Glaive narrative planner [20]. Glaive supports intention but not belief. We used a default model of belief where characters observe the actions that happen at their location, with one exception. When someone sets foot on a location for which someone is the guardian, the guardian observes the action and gets angry. Otherwise, the exemplar stories that involve conflict between Zoe and the lizard beast are not possible.

We fixed a bug in this problem that prevented characters from traveling between locations on the planet.

A dead character always has a utility of 0. Otherwise, characters get 1 point of utility for being healthy and 1 point for being in a safe location. They also gain 1 point for each friend they have and lose 1 point for each enemy. Zoe can make an enemy of the lizard beast by visiting the planet, but the two can become friends via the `make_peace` action. The author goal in the original problem could be achieved without taking any character actions, so we have made the author's utility a little more complex to encourage a variety of stories. The author gets

1 point for making the surface of the planet uninhabitable, 1 point for each dead character, and 3 points if Zoe is friends with the lizard beast. Note it is not possible to achieve author utility 6 because Zoe's friendship toward the lizard beast ends if Zoe dies.

The original solution to this problem used by Ware et al. in their study is not a solution in our translated problem because Sabre requires that solutions be minimal (that is, no actions can be left out without decreasing author utility). The original story involved an interesting but unnecessary fight between Zoe and the lizard beast. That plan is still a valid sequence of explained actions in our problem, but because the fight can be left out, Sabre does not consider it a solution.

Solution 1: The volcano erupts.
(Shortest known plan for author utility 1)

```
begin_erupt(Surface)
erupt(Surface)
```

Solution 2: Zoe dies from the volcano.
(Shortest known plan for author utility 2)

```
teleport_from_ship(Zoe, Ship, Surface)
begin_erupt(Surface)
erupt(Surface)
```

Solution 3: Zoe and the lizard beast make peace.
(Shortest known plan for author utility 3)

```
teleport_from_ship(Zoe, Ship, Surface)
walk(Zoe, Surface, Cave)
make_peace(Zoe, Lizard, Cave)
```

Solution 4: Zoe and the lizard beast make peace, then the volcano erupts.
(Shortest known plan for author utility 4)

```
teleport_from_ship(Zoe, Ship, Surface)
walk(Zoe, Surface, Cave)
make_peace(Zoe, Lizard, Cave)
begin_erupt(Surface)
erupt(Surface)
```

Solution 5: Zoe and the lizard beast make peace, then the beast dies.
(Shortest known plan for author utility 5)

```
teleport_from_ship(Zoe, Ship, Surface)
walk(Lizard, Cave, Surface)
make_peace(Zoe, Lizard, Surface)
begin_erupt(Surface)
teleport_to_ship(Zoe, Surface, Ship)
erupt(Surface)
```

Plan 6: Story used in the study by Ware et al. [22].

```
teleport_from_ship(Zoe, Ship, Surface)
walk(Lizard, Cave, Surface)
attack(Lizard, Zoe, Surface)
stun(Zoe, Lizard, Surface)
begin_erupt(Surface)
teleport_to_ship(Zoe, Surface, Ship)
break_free(Lizard, Surface)
walk(Lizard, Surface, Cave)
erupt(Surface)
```

Version	Goal Utility
space_any	1
space_two	2
space_three	3
space_four	4
space_all	5

Setting	Value
Author Temporal Limit	9
Character Temporal Limit	3
Epistemic Limit	1

Raiders of the Lost Ark

This problem was used as an example of a story that cannot be told without conflict in Ware and Young's paper on the Glaive narrative planner.

> Stephen G. Ware and R. Michael Young. Glaive: A state-space narrative planner supporting intentionality and conflict. In *Proceedings of the 10th AAAI conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 80–86, 2014

A slightly expanded version of the problem is detailed in the appendix of Ware's dissertation.

Stephen G. Ware. *A plan-based model of conflict for narrative reasoning and generation.* PhD thesis, North Carolina State University, 2014

It describes the plot of the 1981 film by the same name. Indiana Jones is the only one who knows where to find the Ark of the Covenant. He travels to Tanis to excavate it, but before he can return home he is intercepted by the Nazis. They believe the Ark will grant them power and open it, but instead its angelic guardians kill the Nazis, leaving Jones free to retake it and return home.

This planning domain and its one associated problem were created for the Glaive narrative planner [20] which supports intention but not belief. The original problem used a `knows-location` predicate to ensure only Jones could dig up the Ark. We have replaced that with Sabre's model of belief. Because the Nazis could not have wrong beliefs about the outcome of opening the Ark, their goal in the original problem is that the Ark be open. In our version, their goal is to be immortal, and they have a wrong belief about the danger of opening the Ark, which is more in line with the film the problem is based on.

The author's utility is a straightforward translation of the original film's ending and the original problem's goal: 1 point if both the US Army has the Ark and the Nazis are dead, otherwise 0. Jones and the US Army have the same goal, that the US Army should have the Ark. The problem cannot be solved without reasoning about conflict—the ability of characters to form and begin executing plans and then be thwarted by others. Jones is the only one who can excavate the ark. He does so hoping to bring it to the Army, but before he can finish his plan the Nazis steal it.

Solution 1: Seemingly unnecessary travel.
(Shortest known solution)

```
travel(Jones, USA, Tanis)
dig(Jones, Ark, Tanis)
take(Nazis, Ark, Jones, Tanis)
travel(Nazis, Tanis, USA)
open(Nazis, Ark, USA)
take(USArmy, Ark, Nazis, USA)
```

The shortest solution deviates from the plot of the original film by having the Nazis travel to the US before opening the Ark. Their plan is unnecessarily long, but still valid. At first, it may seem to violate Sabre's minimality constraint, because they could omit the travel action and still complete their plan, but they cannot omit the travel action and still open the Ark *in the US*. This situation highlights a problem with putting a location parameter in the signature of each action—`open(Nazis,`

`Ark, USA)` and `open(Nazis, Ark, Tanis)` are different actions, though we tend to think of them as the same.

Solution 2: Plot of the original film.
(Alternative solution)

```
travel(Jones, USA, Tanis)
dig(Jones, Ark, Tanis)
take(Nazis, Ark, Jones, Tanis)
open(Nazis, Ark, Tanis)
take(Jones, Ark, Nazis, Tanis)
travel(Jones, Tanis, USA)
give(Jones, Ark, USArmy, USA)
```

Version	Goal Utility
`raiders`	1

Setting	Value
Author Temporal Limit	7
Character Temporal Limit	4
Epistemic Limit	1

Treasure Island

This problem was used as a minimal example of a story demonstrating intention, conflict, wrong beliefs, and communication in the paper that introduced Sabre's model of belief.

> Alireza Shirvani, Stephen G. Ware, and Rachelyn Farrell. A possible worlds model of belief for state-space narrative planning. In *Proceedings of the 13th AAAI conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 101–107, 2017

It is a simplified version of Robert Louis Stevenson's story by the same name. Jim Hawkins inherits a treasure map detailing the location of buried pirate treasure, but he has no ship to reach the island. Long John Silver has a ship but does not know the location of the treasure. First, Hawkins spreads a rumor that the treasure is buried on Treasure Island. Silver sails them both to the island. Hawkins digs up the treasure. Both characters hope to end up with the treasure, but Hawkins succeeds in the end.

The problem was designed before Sabre was implemented, and once tested we found it had two bugs which demonstrate the complexity of belief reasoning and which we have fixed. In the original, Silver begins the story believing the treasure's

location is unknown: `believes(Silver, location(Treasure))=?`. This uses Sabre's unknown constant `?`, which behaves more like *null* than true uncertainty, so it might be more accurate to say that Silver believes the treasure is nowhere. Missing from the original problem, but added in our translation, is the additional fact that Hawkins believes Silver believes the treasure is nowhere. If this is not specified, then Sabre will assume Hawkins believes that Silver believes the same as he—that is, Hawkins will believe Silver knows the location of the treasure. If Silver already knows the location of the treasure, there is no need for the `rumor` action.

The second bug in the original was that Hawkins was the only character who observed `rumor`. At first this does not seem like a bug, because the effects of `rumor` directly modify Silver's beliefs so that he thinks the treasure is buried on the island. However, if Silver does not also observe the action, Silver will not believe that Hawkins believes the treasure is on the island. Silver can only expect Hawkins to sail with him to the island if Silver believes Hawkins believes there is treasure to be had there. In our translation, Silver also observes `rumor`.

Utility functions are simple. Characters get 1 point for having the treasure and otherwise have a utility of 0. The author gets 1 point for Hawkins having the treasure, otherwise 0.

This problem is small and simple, perhaps trivially so. We added it to this collection because it demonstrates many of Sabre's features, including intention, conflict, belief, and belief updates. It can be a valuable debugging tool. It is also interesting because it requires an epistemic limit of 3 or higher to solve.

Solution 1: The plot of the original story.
(Shortest known solution)

```
rumor()
sail()
dig()
take(Hawkins, Treasure)
```

Version	Goal Utility
`treasure`	1

Setting	Value
Author Temporal Limit	4
Character Temporal Limit	4
Epistemic Limit	3

Save Gramma

This problem was originally based on a narrative planning domain for the game *The Best Laid Plans* [21], a prototype interactive narrative driven by the Glaive narrative planner [20]. Ware details the full original domain and two associated problems in his dissertation [16]. Solutions in the original are long due to a large map of mostly empty locations. For future experiments, Ware et al. distilled the most interesting parts of this problem into a smaller problem called *Save Gramma* which was the basis for a game used to validate that Sabre's model of character behavior was believable.

> Stephen G. Ware, Edward T. Garcia, Mira Fisher, Alireza Shirvani, and Rachelyn Farrell. Multi-agent narrative experience management as story graph pruning. *IEEE Transactions on Games*, 15(3):378–387, 2022

This version of the problem, originally written for Sabre, is meant to be as small as possible while still offering a wide variety of possible stories. The player takes the role of Tom, who is given a coin by his sick grandmother. Tom can walk to the market to buy some medicine. Tom wins if he can return home with the medicine. He loses if he dies along the way. The problem also includes a bandit who wants to steal items and a town guard who wants to punish criminals.

There are several ways to win and lose. Tom can be killed by the bandit during a robbery. He can buy the medicine and return home. He can use his coin to buy a sword and resort to robbery to get the medicine, though he then risks being killed by the guard. Tom can also steal the coin from the bandit's camp to buy both a sword and the medicine. Tom can report the bandit's location to the guard, either to get the guard's help or to get the guard to leave his post so Tom can rob the merchant with no witnesses.

Tom gets 1 point of utility for being home with the potion; otherwise his utility is 0. The merchant gets 2 points for every coin she has, as long as she is not a criminal. The merchant also gets 1 point for being at the market, her preferred location. The guard gets 2 points of utility when the bandit is dead, as long as the guard is not a criminal. The guard also gets 2 points of utility if Tom is a criminal, Tom is dead, and the guard is not a criminal. The guard gets 1 point for being at the market, his preferred location. The bandit gets 2 points for each coin she has (her coin can remain in the chest at her camp). The bandit also gets 2 points for having the medicine. The bandit gets 1 point for being at the camp, her preferred location.

The author gets 1 point when Tom is dead but 2 points if Tom is home with the medicine. Thus, the author will accept any story that ends, but prefers for Tom to win.

Solution 1: Tom dies in a robbery.
(Shortest known plan for author utility 1)

```
walk(Tom, Cottage, Crossroads)
walk(Bandit, Camp, Crossroads)
attack(Bandit, Tom, Crossroads)
```

Solution 2: Tom buys the medicine.
(Shortest known plan for author utility 2)

```
walk(Tom, Cottage, Crossroads)
walk(Tom, Crossroads, Market)
buy(Tom, Medicine, TomCoin, Market)
walk(Tom, Market, Crossroads)
walk(Tom, Crossroads, Cottage)
```

Solution 3: Tom turns to crime and pays the price.
(Alternative solution for author utility 1)

```
walk(Tom, Cottage, Crossroads)
walk(Tom, Crossroads, Market)
buy(Tom, MerchantSword, TomCoin, Market)
rob(Tom, Medicine, Merchant, Market)
attack(Guard, Tom, Market)
```

Solution 4: Sometimes crime does pay.
(Alternative solution for author utility 2)

```
walk(Tom, Cottage, Crossroads)
walk(Tom, Crossroads, Market)
buy(Tom, MerchantSword, TomCoin, Market)
rob(Tom, Medicine, Merchant, Market)
walk(Tom, Market, Crossroads)
walk(Tom, Crossroads, Cottage)
```

This problem has many other plans made of explained actions which Sabre does not consider solutions because they are not minimal (that is, some actions could be removed without lowering the author utility). These plans are still interesting because this problem was designed to be interactive, with a player controlling Tom and the planner constantly rewriting the story in response. Here is one example of an interesting but non-minimal plan where Tom reports the bandit's location to the guard.

Plan 5: Tom gets help from the guard.

```
walk(Tom, Cottage, Crossroads)
walk(Bandit, Camp, Crossroads)
rob(Bandit, TomCoin, Tom, Crossroads)
walk(Tom, Crossroads, Market)
report(Tom, Crossroads, Market)
walk(Guard, Market, Crossroads)
attack(Guard, Bandit, Crossroads)
walk(Tom, Market, Crossroads)
loot(Tom, TomCoin, Bandit, Crossroads)
walk(Tom, Crossroads, Market)
buy(Tom, Medicine, TomCoin, Market)
walk(Tom, Market, Crossroads)
walk(Tom, Crossroads, Cottage)
```

Though all four exemplar solutions can be found with an epistemic limit of 1, we recommend setting the epistemic limit to 2 for this problem because some interesting actions (like `report`) are only possible when one character anticipates the actions of another.

Version	Goal Utility
`gramma_any`	1
`gramma_win`	2

Setting	Value
Author Temporal Limit	6
Character Temporal Limit	5
Epistemic Limit	2

Jailbreak

This problem was used in a series of experiments by Farrell, Ware, and Baker to investigate which events people remember and expect during an interactive story.

> Rachelyn Farrell, Stephen G. Ware, and Lewis J. Baker. Manipulating narrative salience in interactive stories using Indexter's Pairwise Event Salience Hypothesis. *IEEE Transactions on Games*, 12(1):74–85, 2020

The authors later implemented the story in Sabre. It is about two prison inmates, Ernest and Roy. They steal a pack of cigarettes and become the target of the prison bully who threatens to kill them. Ernest plans escape by stealing some civilian clothes to flee down the highway. Roy plans revenge on the bully by stealing a

knife to stab him in the gym. In the original experiment, both make preparations for their plans and meet at the air vents. The vents lead both to the highway (for escape) and the gym (for revenge). The guards come running, and one inmate must give himself up so the other can succeed. The experiment altered the characters, timing, and locations of events to measure which events players remembered and which ending they chose.

Both Ernest and Roy gain 2 points of utility for stealing the cigarettes. They each take a -1 penalty if they have been threatened by the bully. They can avoid that penalty by escaping or killing the bully. They also gain a -3 penalty for being dead. The bully gains 1 point for being in the gym, his preferred location. He takes a -1 penalty if someone he has threatened is alive. He can remove that penalty by killing the people he has threatened. The author gains 1 point for each dead inmate. The author also gains 3 points when the cigarettes have been stolen and the threat to each inmate has been neutralized.

Solution 1: The bully kills an inmate.
(Shortest known plan for author utility 1)

```
steal(Ernest, Cigarettes, Cells)
confiscate(Ernest, Cigarettes, Cells, Gym)
recreation(Bully, Gym)
kill(Bully, Ernest, Gym)
```

Solution 2: Ernest escapes.
(Shortest known plan for author utility 3)

```
steal(Ernest, Cigarettes, Cells)
chores(Ernest, Laundry)
steal(Ernest, Clothes, Laundry)
confiscate(Ernest, Cigarettes, Laundry, Highway)
disguise(Ernest, Clothes, Highway)
escape(Ernest, Highway)
```

Solution 3: Roy gets revenge.
(Shortest known plan for author utility 6)

```
steal(Roy, Cigarettes, Cells)
recreation(Bully, Gym)
chores(Roy, Kitchen)
steal(Roy, Knife, Kitchen)
confiscate(Roy, Cigarettes, Kitchen, Gym)
lock_gym(Roy, Gym)
revenge(Roy, Bully, Gym)
```

There were many versions of the story used in Farrell et al.'s experiments that varied in the order and content, but all of them were a mix of the escape and revenge plans with only one inmate succeeding at the end. The actions in these plans are explained but not minimal (that is, actions can be removed without decreasing the author's utility), so Sabre does not consider them solutions.

Version	Goal Utility
`jailbreak_lose`	1
`jailbreak_escape`	3
`jailbreak_revenge`	6

Setting	Value
Author Temporal Limit	7
Character Temporal Limit	6
Epistemic Limit	1

Lovers

This problem was proposed by Farrell and Ware as a randomizable setting for testing belief and intention recognition using an early version of Sabre.

> Rachelyn Farrell and Stephen G. Ware. Narrative planning for belief and intention recognition. In *Proceedings of the 16th AAAI conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 52–58, 2020

The problem is a grid world filled with agents and items. Each character wants one of the items. A character is happy if they have the item they want. Each character also loves one other character. A character gets 1 point of utility for being happy and 1 point if the character they love is happy. The author picks two characters. If both are happy, the author gets 1 point; otherwise the author utility is 0.

The initial locations of agents and items can be randomized. Whom characters love and which two the author wants to be happy can be randomized. Characters can also have random wrong beliefs about where items are or who wants which items.

Characters can move around, pick up items, trade items with others, and tell other characters they want an item. Characters can lie about what item they want.

This problem was designed to feature intention, belief, conflict, communication, and deception but in practice communication is rarely part of an optimal solution because characters can usually correct their wrong beliefs while moving around. Many randomized versions of this problem were used in the original experiments. For our version, we engineered the problem so the shortest solution requires deception.

In the solution, character C1 lies to C2, claiming to want I3 which C2 is holding. C2 will give I3 to C1 because C2 loves C1. That love is unrequited, because C1 trades it to its lover, C3 so both will have the items they want.

```
Solution 1: C1 lies to C2 to get the item C3 wants.
(Shortest known solution)
move(C1, R11, R12)
tell(C1, C2, I3, R12)
give(C2, I3, C1, R12)
move(C1, R12, R22)
trade(C1, I3, C3, I1, R22)
```

Though the solution can be found with an epistemic limit of 1, we recommend setting the epistemic limit to 2 for this problem because many interesting behaviors are only possible when one character anticipates the actions of another.

Version	Goal Utility
`lovers`	1

Setting	Value
Author Temporal Limit	5
Character Temporal Limit	5
Epistemic Limit	2

Acknowledgments

The original research on the Sabre narrative planner, and on these benchmark tests for it, was supported in part by the U.S. National Science Foundation. The opinions and recommendations expressed in this technical report are those of the authors and do not necessarily reflect the views of the National Science Foundation. We would also like to thank the authors of all the original problems that we translated into Sabre syntax for their contributions to the narrative planning research community.

Grant #2145153

Grant #1911053

Bibliography

[1] Marc Cavazza, Fred Charles, and Steven J. Mead. Character-based interactive storytelling. *IEEE Intelligent Systems special issue on AI in Interactive Entertainment*, 17(4):17–24, 2002.

[2] Rachelyn Farrell and Stephen G. Ware. Narrative planning for belief and intention recognition. In *Proceedings of the 16th AAAI conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 52–58, 2020.

[3] Rachelyn Farrell, Stephen G. Ware, and Lewis J. Baker. Manipulating narrative salience in interactive stories using Indexter's Pairwise Event Salience Hypothesis. *IEEE Transactions on Games*, 12(1):74–85, 2020.

[4] Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL: The Planning Domain Definition Language. Technical report, Yale Center for Computational Vision and Control, 1998.

[5] Bilal Kartal, John Koenig, and Stephen J Guy. User-driven narrative variation in large story domains using Monte Carlo Tree Search. In *Proceedings of the international conference on Autonomous Agents and Multiagent Systems*, pages 69–76, 2014.

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

[7] Michael Mateas. An Oz-centric review of interactive drama and believable agents. In *Artificial Intelligence Today: Recent Trends and Developments*, pages 297–328. Springer, 2001.

[8] Michael Mateas and Andrew Stern. A Behavior Language: Joint action and behavioral idioms. In *Life-Like Characters: Tools, Affective Functions, and Applications*, pages 135–161. Springer, 2004.

[9] James R. Meehan. Tale-Spin, an interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 91–98, 1977.

[10] Julie Porteous, Marc Cavazza, and Fred Charles. Applying planning to interactive storytelling: Narrative control using state constraints. *ACM Transactions on Intelligent Systems and Technology*, 1(2):1–21, 2010.

[11] Julie Porteous, Fred Charles, and Marc Cavazza. NetworkING: using character relationships for interactive narrative generation. In *Proceedings of the international conference on Autonomous Agents and Multi-Agent Systems*, pages 595–602, 2013.

[12] Mark O. Riedl. *Narrative planning: Balancing plot and character*. PhD thesis, North Carolina State University, 2004.

[13] Mark O. Riedl and R. Michael Young. Story planning as exploratory creativity: Techniques for expanding the narrative search space. *New Generation Computing*, 24:303–323, 2006.

[14] Mark O. Riedl and R. Michael Young. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39(1):217–268, 2010.

[15] Alireza Shirvani, Stephen G. Ware, and Rachelyn Farrell. A possible worlds model of belief for state-space narrative planning. In *Proceedings of the 13th AAAI conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 101–107, 2017.

[16] Stephen G. Ware. *A plan-based model of conflict for narrative reasoning and generation*. PhD thesis, North Carolina State University, 2014.

[17] Stephen G. Ware and Rachelyn Farrell. A collection of benchmark problems for the Sabre narrative planner. Technical report, Narrative Intelligence Lab, University of Kentucky, November 2023.

[18] Stephen G. Ware, Edward T. Garcia, Mira Fisher, Alireza Shirvani, and Rachelyn Farrell. Multi-agent narrative experience management as story graph pruning. *IEEE Transactions on Games*, 15(3):378–387, 2022.

[19] Stephen G. Ware and Cory Siler. Sabre: A narrative planner supporting intention and deep theory of mind. In *Proceedings of the 17th AAAI conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 99–106, 2021.

[20] Stephen G. Ware and R. Michael Young. Glaive: A state-space narrative planner supporting intentionality and conflict. In *Proceedings of the 10th AAAI conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 80–86, 2014.

[21] Stephen G. Ware and R. Michael Young. Intentionality and conflict in The Best Laid Plans interactive narrative virtual environment. *IEEE Transactions*

on Computational Intelligence and Artificial Intelligence in Games, 8(4):402–411, 2015.

[22] Stephen G. Ware, R. Michael Young, Brent Harrison, and David L. Roberts. A computational model of narrative conflict at the fabula level. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games*, 6(3):271–288, 2014.

[23] Daniel S. Weld. Recent advances in AI planning. *AI Magazine*, 20(2):93, 1999.

[24] R. Michael Young. Notes on the use of plan structures in the creation of interactive plot. In *Proceedings of the AAAI Fall Symposium on Narrative Intelligence*, pages 164–167, 1999.

[25] R. Michael Young, Stephen G. Ware, Bradly A. Cassell, and Justus Robertson. 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, 2013.