Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-20)

Using Domain Compilation to Add Belief to Narrative Planners

Matthew Christensen,[1*] **Jennifer M. Nelson,**[1*] **Rogelio E. Cardona-Rivera**[1,2]
Laboratory for Quantitative Experience Design
[1]School of Computing, [2]Entertainment Arts and Engineering Program
University of Utah, Salt Lake City, UT, USA
matthew.l.christensen@utah.edu, jennifer.m.nelson@utah.edu, rogelio@eae.utah.edu

Abstract

Using domain compilation, we present a narrative planning system that is capable of creating narrative plans that use both character intention and character beliefs. We introduce a model capable of representing character beliefs in PDDL domains. This model allows characters to fail at actions when their beliefs about the world differ from the actual world state. Domains of this type can be compiled into purely intentional domains, and fed as input to intentional planners. The resulting stories feature characters that pursue their own intentions based on their own knowledge of the world, learn from mistakes to update their beliefs, and communicate information to each other. These types of stories are not possible with purely intentional domains.

In stories, characters often have differing beliefs about what is true in the world, and those beliefs play a role in determining what actions those characters take (Young 2017). A villain, for instance, could not steal a valuable artifact unless they knew where that artifact was hidden. Likewise, a hero would not rationally attempt to open a door if they already believed it was locked. Character beliefs thus affect what choices a character makes, and can also contribute to when characters fail. Perhaps the hero initially believed the door was unlocked, but upon trying to open it, it would not budge. These failures occur when a character's beliefs are different from the actual state of the world.

Several scholars have developed computational models that would enable reasoning about belief during procedural narrative generation. Logic-based models (e.g. Wadsley and Ryan 2013, Eger and Martens 2017a) build upon modal logics of belief to characterize the mental states of agents over time. Planning-based models (e.g. Teutenberg and Porteous 2015, Shirvani, Farrell, and Ware 2018) characterize the dynamics of character beliefs relative to the rational deliberation and intentional agent activity.

However, despite broad interest in reasoning about character beliefs during narrative generation, most existing models are either not available as deployed systems or are infeasible to use in practice.

This is problematic. Reasoning about beliefs is critical to meaningfully expand the *expressive range* (Smith and Whitehead 2010) of these procedural narrative generators. Presently, it is needlessly difficult to assess the relative strengths and weaknesses of the diverse, extant models and, as a consequence, forward progress is slow. In this paper, we fill that gap: we present a method to afford reasoning about character beliefs using already-published and efficient narrative systems.

Following Haslum (2012), we developed an automated method to perform *belief compilation* for plan-based procedural narrative: our system[1] performs a systematic remodelling of a belief-augmented intentional narrative planning problem written in the Planning Domain Definition Language (PDDL, McDermott et al. 1998) such that an intentional plan that solves the reformulated problem *without* reified beliefs – e.g. a plan provided by the Glaive narrative planner (Ware and Young 2014) – also solves the narrative planning problem *with* reified beliefs. Succinctly, our compilation transforms a *belief-and-intention narrative planning problem* to an *intention narrative planning problem* that preserves the belief semantics of the original.

Contributions We present a conservative model of narrative character beliefs, and describe its specification as a PDDL belief-and-intention narrative planning problem. We then describe how this belief-and-intention model can be compiled to a narrative planning problem that *only* supports intention while preserving belief semantics. Our procedure can be used in conjunction with an intentional planner to produce narrative plans that codify character reasoning of both beliefs and intentions, and we used Glaive to verify the validity of our approach. Because there are no other belief-supporting narrative planning systems to compare against, we formally characterize our compilation's resulting effect on the narrative problem's complexity. We demonstrate that we can expand the expressive range of procedural narrative generators at a cost of greater problem size: the compiled domain's size is bounded polynomially.Finally, we discuss the broader implications and future plans of our work.

[*]Now at Microsoft.

[1]https://github.com/qed-lab/belief-intention-compilation

1 Related Work

Relative to Genette's (1980) tripartite model of narrative, all prior efforts have targeted modeling character beliefs at the level of the *story*, as opposed to the level of *discourse* or *medium*. Here, we focus on planning-based approaches to modeling character beliefs, and briefly mention a relevant logic-based one.

Teutenberg and Porteous (2015) developed IMPRACTical, a decentralized narrative planning system in which the dynamics of individual characters emerge from correspondingly individual planners. Each character's planner tracks its own set of beliefs and intentions, but their model of belief is strict: in order for a character to plan an action, the character must believe it is possible to carry out and the action must *in reality* be possible to carry out. In other words, characters cannot make mistakes and update their beliefs on the basis of those mistakes. In contrast, our compilation affords authoring planning domains in which characters may attempt actions on the basis of mistaken beliefs. Further, our compilation affords specifying what happens to the mistaken character's beliefs when their action fails. In that sense, our expressive power matches that of the HEADSPACE planner (Thorne and Young 2017). The drawback to HEADSPACE is its reasoning algorithm, which has a large branching factor in order to accommodate the necessary belief information. This is consistent with the belief-intentional planner developed by Shirvani, Farrell, and Ware (SFW, 2018), which searches through a much larger search space. The branching factor is large due to SFW's representation of beliefs via modal logic (like the approaches described earlier), codified using Kripkean possible-worlds semantics (Shirvani, Ware, and Farrell 2017) that afford arbitrary levels of nested beliefs (like those necessary to represent recursive theory of mind).

None of the cited planning systems is publicly available for use, unlike our compilation method. Our belief model represents a single planning "director" agent (unlike IMPRACTical), is as expressive as HEADSPACE, and less expressive than SFW. Our compilation is based in part on the methods within Haslum's (2012) compilation of narrative planning problems built for the Intentional Partial Order Causal Link (IPOCL) planner 2010 to classical planning problems solvable by off-the-shelf systems.

The Haslum Compilation offered significant speedup of narrative planning, thanks to fast heuristic search planning systems, like Fast Forward (Hoffmann 2001). It drew influence from Palacios and Geffner's (2006) classical compilation of conformant planning problems, in which the planner represents an agent who is uncertain about the world. This uncertainty admits the possibility of the planner having an imperfect representation of the world state. Unlike their work, in which the planner has imperfect knowledge, our planner always has perfect knowledge of the world state (i.e. it knows *ground truth*); the planner, however, *also* tracks belief states for the agents it is directing.

Finally, it is worth mentioning that Eger and Martens (2017a) modeled character beliefs via Dynamic Epistemic Logic (DEL, Van Ditmarsch, van Der Hoek, and Kooi 2007), which has affordances to describe how beliefs change over time. In parallel, Eger and Martens (2017b) developed *Ostari*, a (publicly-available) macro system that compiles their domain-specific language for specifying character beliefs to a (cumbersome-to-use) DEL-compliant representation. In effect, they did for logic-based approaches what in this paper we do for planning-based ones. While our compilations are not directly comparable, they are similar in spirit.

2 Narrative Planning Background

As stated, we rely on the formulation of **narratives as plans**. This formulation casts story telling and sensemaking as distinguished kinds of planning processes. This is due to planning's (a) affordances for representing and reasoning about causal, temporal, and hierarchical structure (Young et al. 2013) and (b) correspondence to cognitive theories that posit sensemaking activity as a *search* for meaning (Cardona-Rivera and Young 2019). Our representation is built atop *classical* planning, a problem solving-model wherein agent actions are fully observable and deterministic. For us, a plan represents Genette's (1980) *story level*.

Classical Planning A classical STRIPS-model (Fikes and Nilsson 1971) planning problem is a tuple $\mathcal{P}{=}\langle \mathcal{L}, \mathcal{I}, A, \mathcal{G}\rangle$: $\mathcal{L}$ is a set of ground atoms, $\mathcal{I}{\subseteq}\mathcal{L}$ is the initial state, $\mathcal{G}{\subseteq}\mathcal{L}$ is the set of goal conditions, and A is a set of actions. Each action a is a triple $\langle \textsc{pre}(a), \textsc{add}(a), \textsc{del}(a)\rangle$ of precondition, add, and delete lists respectively, all subsets of $\mathcal{L}$. A state is a conjunction of ground atoms. An action a is applicable in a state s iff $\textsc{pre}(a){\subseteq}s$; applying said action in the state results in a new state $s'{=}(s{\setminus}\textsc{del}(a)){\cup}\textsc{add}(a)$. A solution to $\mathcal{P}$ is a plan $\pi{=}[a_1, ..., a_m]$, a sequence of actions $a_i{\in}A$ that transforms the initial state $\mathcal{I}$ to a state s_m that satisfies the goal; i.e. $\mathcal{G}{\subseteq}s_m$. We assume problems are codified via PDDL.

Intentional Planning Classical planners are built to solve problems with efficient plans. While useful in most task environments, narrative-theoretic success depends less on agent optimality and more on agent *believability* (Riedl and Young 2010). Agents are narratively believable to the degree they seem *intentional* (Bates 1994). We build atop methods established by intention-supporting *narrative planners*.

An intentional narrative planning problem modifies the set of actions A of classical STRIPS-style problems; an action $a{\in}A$ can now specify one or more *consenting* agents (i.e. story characters). Transitively, said action a is only applicable if all of its preconditions are met (as described) *and* a leads toward achieving a goal of each of its consenting agents, who intend that the goal be satisfied (Riedl and Young 2010; Ware 2012; Ware and Young 2014).

An agent's goals are represented using the modal predicate *intends*. For instance, a villain may intend to have an artifact, and that intention is represented in PDDL-style as `(intends villain (has villain artifact))`.

It is important to make a distinction between a character's goals, and goals in a classical planning sense. A character's goals are not necessarily the same as the goal state of a planning problem. An intentional narrative problem's goal

state are the author's goals for the *outcome* of the story. For clarity, we refer to the narrative planning system – i.e. the planning agent responsible for orchestrating the behavior of story characters – as the *story director* (agent). Story directors will search for a sequence of events that results in a state that satisfies the outcome, while ensuring every character action is a step toward that character's goals.

3 Our Model of Belief

The term *belief* as it relates to narrative and planning has a specific meaning. In the BDI model, a character's belief state is their understanding of the world state (Rao and Georgeff 1995). It is *rational* that a character acts per their beliefs; a character can be *rationally critiqued* (Bratman 1987) if they behave in a way unsupported by what they believe.

In general, beliefs are not limited to simple facts of the world: a character can have beliefs about another character's intentions or beliefs. This can quickly lead to very complex situations, as can be seen in the work by Shirvani, Ware, and Farrell (2017), where characters can have beliefs about others' beliefs to arbitrary depth. To make the problem more tractable, we restrict our model to only allow for beliefs of simple statements about the world (non-modal atoms). This corresponds to a *theory-of-mind* of 1 layer, which we posit is sufficient to express interesting character dynamics, though humans can reason with more layers. Later, we demonstrate that our belief model is expressive enough to afford generating stories that could not be created using a purely intentional narrative planner.

Of particular interest to us are story situations in which a character attempts to take an action they believe is possible, but in reality is impossible. Such a situation can only arise when an agent has at least one belief about the world state that isn't true and that belief plays a role in what they want to do. In classical and intentional narrative planning, if an action is impossible (i.e., when its preconditions aren't met), it is deemed non-applicable, and thus cannot be planned with. However, it should be possible for a character to plan to execute an action they cannot actually complete if they *believe* they can. Such an attempt results in failure. Normatively (Bratman 1987), characters should not act unless they believe their action will succeed. Our model of belief encodes this by requiring characters to believe all preconditions of an action before they attempt to take it.

Lastly, any principled model of character belief has to account for how those beliefs *change* (Alchourrón, Gärdenfors, and Makinson 1985; Da Costa and French 1989). In task environments, failing any action has consequences that can affect both the state of the world and the beliefs of the character. There are a number of *micro-theories* (Lenat and Guha 1990) that could be axiomatically applied to update character beliefs when an action succeeds or fails; see Smullyan (2012) for a discussion about different types of possible *human reasoner models* with associated axioms about belief. We are motivated by affording the novel procedural narrative capacity of generating stories in which a character fails *for the purpose* of having them learn something important about the world. In fact, in one of our example domains – *Hubris*, Figure **??** – a failure opens up a new opportunity to achieve character/author goals. To be conservative, and as a matter of retaining authoring flexibility, we do not commit to a *particular* belief-update microtheory. Instead, we require that domain authors explicitly identify all the individual belief changes that result from action failure (like Thorne and Young 2017).

4 Belief Model Formulation

In this section, we present our STRIPS-like belief formalism that codifies §3's belief model. We then present its corresponding PDDL-encoding; the encoding is the input to our compilation, which is conceptually illustrated in Figure 1.

Formal Definition

A **belief-intention narrative planning problem** is a tuple $\mathcal{P}{=}\langle \mathcal{L}, \mathcal{B}, \mathcal{N}, \mathcal{I}, A, \mathcal{G}, \mathcal{C}\rangle$; $\mathcal{L}$ and $\mathcal{G}$ are as in §2, $\mathcal{B}$ and $\mathcal{N}$ are sets of modal atoms (explained below), $\mathcal{I}{\subseteq}\mathcal{L}{\times}\mathcal{B}{\times}\mathcal{N}$, $\mathcal{C}$ is a set of constants (predicate logic 0-ary functions) representing story *characters*, and A is a set of *intentional* actions.

Belief-Intention planning problems can syntactically specify *character beliefs* and *character intentions*. Character beliefs are *modal atoms* of the form $B(c,\phi){\in}\mathcal{B}$, denoting that character c believes non-modal atom $\phi{\in}\mathcal{L}$ or its negation to be the case. Similarly, character intentions are modal atoms of the form $I(c,\phi){\in}\mathcal{N}$, denoting that character c intends *to* make ϕ true, without commitment (i.e. a plan) to how they will effect it.

Thus, a **belief-intention narrative state** is a triple $s{=}\langle W, B, I\rangle$; $W{\subseteq}\mathcal{L}$ represents the ground truth state of the world, $B{\subseteq}\mathcal{B}$ represents the beliefs held by characters in $\mathcal{C}$, and $I{\subseteq}\mathcal{N}$ represents the intentions held by characters in $\mathcal{C}$.

A **belief-intentional action** $a{\in}A$ is represented as:

$$a = \langle \textsc{pre}(a), \textsc{add}_s(a), \textsc{del}_s(a), \textsc{add}_f(a), \textsc{del}_f(a), \textsc{cha}(a)\rangle$$

Here, $\textsc{pre}(a)$ is as in §2 and $\textsc{cha}(a)$ is the (possibly empty) set of characters from $\mathcal{C}$ that must consent to the action's execution. These actions have *two* sets of add and delete lists: the *success* lists $\textsc{add}_s(a)$ and $\textsc{del}_s(a)$, and the *failure* lists $\textsc{add}_f(a)$ and $\textsc{del}_f(a)$. These lists dictate how a narrative state s evolves relative to action a's *success* or *failure*, per the action being *possible* or *not*.

Because add/delete lists represent state-changes, and because belief-intention narrative states are triples, belief-intentional action add/delete lists are triples as well. In other words, the add (delete) list in a belief-intentional action is a triple $\langle W_{\textsc{add}(\textsc{del})}, B_{\textsc{add}(\textsc{del})}, I_{\textsc{add}(\textsc{del})}\rangle$, that represent changes to be made to the world state, character belief states, and character intentions respectively. To obtain a new state s', we apply the respective add and delete lists to state s as in classical planning; e.g., $W'{=}(W{\setminus}W_{\textsc{del}}){\cup}W_{\textsc{add}}$.

Action success and failure are defined relative to *character beliefs*, whereas action possibility is defined relative to the *ground truth state of the world*. From the *story director's* perspective, an intentional action a is applicable in a belief-intention state s in two cases.

If a has no consenting characters ($\textsc{cha}(a){=}\varnothing$), then a is applicable in s iff $\textsc{pre}(a){\subseteq}W{\in}s$. These applicable actions

cannot fail, are taken by the planning agent, and are referred to as "acts of fate" (Riedl and Young 2010). In this case, applying a in s means applying $\text{DEL}_s(a)$ and $\text{ADD}_s(a)$ to s.

In contrast, if a has at least one consenting character, then a is applicable in s iff $\forall c{\in}\text{CHA}(a)\forall l{\in}\text{PRE}(a), B(c,l){\in}B{\in}s$. These applicable actions *can* fail: the stated condition does not require the preconditions of a to be satisfied by the world's ground truth state W. In this case, applying a in s means one of two things. If a is *possible* – i.e. $\text{PRE}(a){\subseteq}W{\in}s$ – then the action *succeeds* and it means applying $\text{DEL}_s(a)$ and $\text{ADD}_s(a)$ to s as before. However, if a is *not possible* – i.e. $\text{PRE}(a){\not\subseteq}W{\in}s$ – then the action *fails* and it means applying $\text{DEL}_f(a)$ and $\text{ADD}_f(a)$ to s.

PDDL Encoding

We encode belief-intention narrative planning problems by expanding the base PDDL representation with custom constructs. Like standard PDDL, we codify a belief-intention planning problem instance $\mathcal{P}{=}\langle\mathcal{L},\mathcal{B},\mathcal{N},\mathcal{I},A,\mathcal{G},\mathcal{C}\rangle$ via two factored representations: the *domain description* and the *problem description*.

Domain Description The domain description codifies the set of predicates P from $\mathcal{L}$, and the belief-intentional actions from A as a set of *template operators* O; thus, an action $a{\in}A$ is an *instance* of a template operator $o{\in}O$, represented as:

$$\langle\text{Name}(o),\text{Param}(o),\text{Precon}(o),\text{Eff}(o),\text{Fail}(o),\text{Agents}(o)\rangle$$

The *Name* syntactically distinguishes it from other operators that might have the same arity. *Param*eters are the terms from $\mathcal{L}$ relevant for the action in question. Both the *Precon*dition and *Eff*ect are unground logic formulae that codify a belief-intentional action a's preconditions $\text{PRE}(a)$ and *success* add/delete lists $\text{ADD}_s(a)$/ $\text{DEL}_s(a)$, respectively. *Fail* is an unground logic formula that codifies a's *failure* add/delete lists $\text{ADD}_f(a)$/ $\text{DEL}_f(a)$. Finally, *Agents* codifies a's consenting agents $\text{CHA}(a)$. Listing 1 illustrates an example operator drawn from our *rooms* domain.

```
(:action enter  :parameters (?who ?r1 ?r2)
  :precondition (and (not (locked ?r2))
                     (in ?who ?r1))
  :effect       (and (not (in ?who ?r1))
                     (in ?who ?r2)
                     (believes ?who (not (in ?who ?r2)))
                     (believes ?who (in ?who ?r2)))
  :fail         (believes ?who (locked ?r2))
  :agents       (?who))
```

Listing 1: An example belief-intentional action operator in our PDDL-style encoding. Here, the consenting agent would have to believe that they are in room `?r1` and that room `?r2` is not locked for this action to be applicable in a state.

Listing 1's `enter` represents a character attempting to leave one room and enter another. It contains two departures from standard PDDL: the `believes` predicate and the `:fail` heading. In keeping with PDDL-convention, we signal the use of these special-purpose planning constructs via *domain requirements*, identified in the domain description with a `:requirements` heading.

We introduce the `:belief` requirement, which affords using `:fail` within operator definitions. It also *implicitly* defines – i.e. it is not added to P – and allows the use/-compilation of the belief modal predicate `(believes ?c ?l)`; `?l` is some non-modal predicate or its negation and `?c` is a character defined within the problem description. The `:intentionality` requirement achieves the same effect for the intention modal predicate `(intends ?c ?l)`. Together these codify $\mathcal{B}$ and $\mathcal{N}$. These requirements demand being able to distinguish constants as characters. Thus, the `:belief` and `:intentionality` requirements entail a *derived* `:types` (standard-PDDL) requirement that introduces a *typing system* T (Wickler 2011) to the domain description. In sum, the **domain description** is a tuple $D{=}\langle R,T,P,O\rangle$, grouping a set of *R*equirements, *T*ype system, set of (unground) *P*redicates, and set of *O*perators.

Problem Description The problem description codifies the problem instance's initial state $\mathcal{I}$ and goal state $\mathcal{G}$ directly, and also codifies the set of constants $\mathbf{C}$ (logical 0-ary functions) from $\mathcal{L}$. The constants include the (typed) character terms C. The belief state for each character c is encoded in the state of the world via all statements of the form `(believes c ?l)` that are either true in the initial state or made true through some action. Thus, the **problem description** is a tuple $P{=}\langle D,\mathcal{I},\mathcal{G},\mathbf{C}\rangle$, grouping a *D*omain description, initial state, goal coditions, and constant terms.

The Closed World Assumption of Beliefs

In our implementation, we allowed for actions to have negative preconditions. [2] Under the Closed World Assumption (CWA), preconditions of the form $\neg p$ for some formula p in state s are satisfied iff $p \notin s$. However, in belief planning, we consider negative preconditions to be satisfied in s *with respect to the belief of a consenting agent* c iff there is a literal of the form `(believes c (not p))`$\in s$.

This is different from `(not (believes c p))`. The former explicitly means that c believes that p is false. The latter, conjoined with `(not (believes c (not p)))` implies that c is *agnostic* regarding p. Thus, while the CWA still applies for the story director, it does not strictly apply for the story's characters. For instance, in the *rooms* domain (Figure **??**), the main character does not have any belief about the key.

5 Belief-Intention Compilation

Our compilation systematically transforms a PDDL belief-intention narrative domain $D{=}\langle R,T,P,O\rangle$ and problem $P{=}\langle D,\mathcal{I},\mathcal{G},\mathbf{C}\rangle$ into an alternate PDDL domain D' and problem P', which does away with the belief modality and preserves the original domain and problem's semantics. D' and P' represent a compiled intentional domain and problem, respectively. From there, the compiled domain and problem can be supplied to an intentional planner to obtain a plan (Figure 1); we use Glaive (Ware and Young 2014), but any system that can create plans from intentional domains could be used (e.g., Haslum's intention compilation followed by a classical planner). Our compilation leaves T and $\mathbf{C}$ untouched, and updates

[2] enabled by the encoding mentioned in section whatever

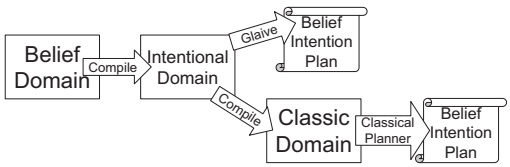

Figure 1: An overview of the belief planning process.

R to remove the `:belief` requirement and add the `:intentionality`, `:negative-preconditions`, and `:disjunctive-preconditions` requirements. Remaining domain and problem elements are modified as follows:

Initial and Goal State Compilation $\forall i \in \mathcal{I}$:

- If i is of the form `(believes c (m x1 x2 ...))` for character c & non-modal atomic formula m, create a new flattened literal `(believes_m c x1 x2 ...)`; add to $\mathcal{I}'$.
- If i's form is `(believes c (not (m x1 x2 ...)))` for character c & non-modal grounded atomic formula m, create a new flattened grounded atomic formula `(believes_not_m c x1 x2 ...)` and add it to $\mathcal{I}'$.
- Otherwise, add i to $\mathcal{I}'$.

The first two conditions flatten belief formula into non-modal versions, where the character who has the belief is appended as the first parameter. For example, `(believes villain (not (at artifact cave)))` compiles to `(believes_not_at villain artifact cave)`. In effect, each character belief is treated as part of the world state. We repeat the process with the beliefs in $\mathcal{G}$ to obtain $\mathcal{G}'$.

Predicate Compilation $\forall p \in P$ of the form `(p ?x1 ?x2 ...)`, create:

- a new atomic formula p_b of the form: `(believes_p ?c ?x1 ?x2 ...)`, and
- a new atomic formula $p_{\neg b}$ of the form: `(believes_not_p ?c ?x1 ?x2 ...)`

where `?c` is of type `character`. Add all predicates to P'.

Operator Compilation $\forall o \in O$:

- if Agents$(o) = \varnothing$, create a new operator $o' = o$. If o' contains `believes` predicates in the preconditions or effects, flatten them into non-modal versions as in the State Compilation. Add o' to O'.
- Otherwise, create *two* new operators to eventually add to O': o'_s – the success variant – and o'_f – the failure variant. These new operators are defined on the basis of o with belief predicates flattened as in the State Compilation:
  - Agents(o'_s)=Agents(o'_f)=Agents(o),
  - Param(o'_s)=Param(o'_f)=Param(o),
  - Eff(o'_s)=Eff(o) and Eff(o'_f)=Fail(o),

  The preconditions of the success and failure variants are defined relative to the consenting agents. Every consenting agent must believe the operator's preconditions are met; these beliefs are flattened into a set Precon$_c^B$ for every consenting agent c as follows:
  - $\forall c \in$Agents(o):
    - Create a new logical sentence Precon$_c^B$ as a copy of Precon(o) in negation normal form.[3]
    - $\forall p \in$Precon$_c^B$, replace it with a new atomic formula p_c^B of the form `(believes_p ?c ?x1 ?x2 ...)` (or `(believes_not_p ...)` if negated). If p is already a `believes` formula, flatten it. Then,

$$\text{Precon}(o'_s) = \text{Precon}(o) \wedge \left[\bigwedge_{c \in \text{Agents}(o)} \text{Precon}_c^B \right]$$

$$\text{Precon}(o'_f) = \neg\text{Precon}(o) \wedge \left[\bigwedge_{c \in \text{Agents}(o)} \text{Precon}_c^B \right]$$

In other words, for o'_s to be applicable, all of o's preconditions must be satisfied, plus every consenting character must believe those preconditions are met (in a flattened version). Contrarily, for o'_f to be applicable, every consenting character must believe those preconditions are met (in a flattened version), but all of o's preconditions must remain *unsatisfied*; alternatively, the negation of o's preconditions must be satisfied. The sentence $\neg$Precon(o) may reverberate as disjunctive and negated preconditions, hence their inclusion as requirements in PDDL.

6 Evaluation

Domain Size To quantify the performance of our method – and given that no belief planners exist that might serve as good runtime baselines – we analyzed the theoretical size of the compiled domain relative to the uncompiled domain. In general, a larger search space will exponentially increase the time it takes to plan (Bylander 1991). It is thus worthwhile to quantify how big compiled domains can get. For every predicate in a belief problem, the compilation adds two more predicates representing the *believes* and *believes_not* variants. Thus, the compilation multiplies the number of predicates in the domain by a factor of 3. Actions in the domain with consenting agents are split into their success and failure versions. This effectively doubles the number of actions in the domain: $|A'|=2|A|$ and $|\mathcal{L}'|=3|\mathcal{L}|$.

Sample Domains We also explored the authorial affordances of our system in a more-qualitative and reflexive manner, which was the motivating reason for pursuing more-expressive narrative planners via belief modeling in the first place. We developed 3 domains to illustrate the space of potential stories that can be generated by our model of belief (and its corresponding compilation). We also computed plans for the compiled problems using Glaive (Figure 2).

The *rooms* domain in Figure 3 is built to demonstrate how belief can facilitate the spread of information. The main character, Alice, wants to obtain the star. However, she doesn't know which room the star is in, nor does she know

[3] This is so negative preconditions are treated as `(believes (not ...))` as opposed to `(not (believes ...))`, preserving the open world for character beliefs.

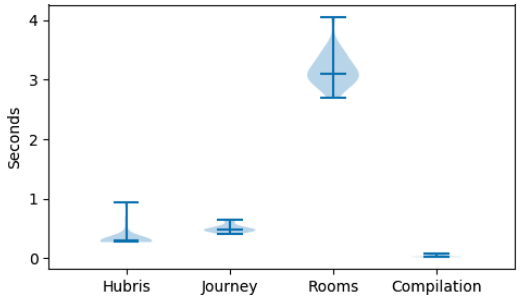

Figure 2: Glaive planning time for each domain (n=25) and compilation time for all domains (n=75). Computed on a 3.1 GHz Dual-Core Intel Core i5 processor.

that the room is locked. There are letters in rooms 2 and 6 that have information about where the key and star are. In an intentional domain, the optimal plan solution would have Alice first enter the room where the key is, then use it to unlock the door and obtain the star. However, in the belief domain, it doesn't make sense for her to take those actions because she wouldn't even know she needs to search for a key yet. First, she must learn that the door is locked by attempting to open it and failing. Then, by traveling to other rooms, she can find letters that tell her where the key is.

The *hubris* domain (Figure 3) was created to demonstrate how failure can add drama to a situation. Here, a villain has obtained the ultimate artifact, and intends to use it to destroy the hero and take over the world. However, they lack the arcane knowledge necessary to wield the artifact. Thus, when they attempt to use it, they fail, alerting the hero in the process. Our hero, who can wield the artifact, takes this opportunity to steal it from the villain and use it against them. In an intentional domain, the villain wouldn't attempt to use the artifact, because they could not do so successfully. It is only the villain's mistaken belief that allows the story to progress to the author's goal.

We created the *journey* domain (Figure 3) to show how belief can add obstacles to a character's journey. Here, we've given Alice the power to turn away a calamity facing her village. However, she doesn't yet know that she has this power. She must take a journey to a far off land before she learns that she had the power the whole time. Like the *rooms* domain, the optimal intentional plan for the story is much shorter, but allowing characters to have incorrect or incomplete beliefs adds interest to the story.

7 Limitations and Future Work

We have not discussed one key feature of intentional narrative planners: *expression parameters* in operators, which allow actions to have parameters that can bind to *literals* as opposed to the default limitation of binding to *terms*. Within intentional planning, this feature supports *delegation*: when a character elicits an intention in some other target character (e.g. by commanding the target, Riedl and Young 2010). Our system can compile said actions, but we do not discuss it here due to space constraints. However, the resulting compiled domains can take *hours*

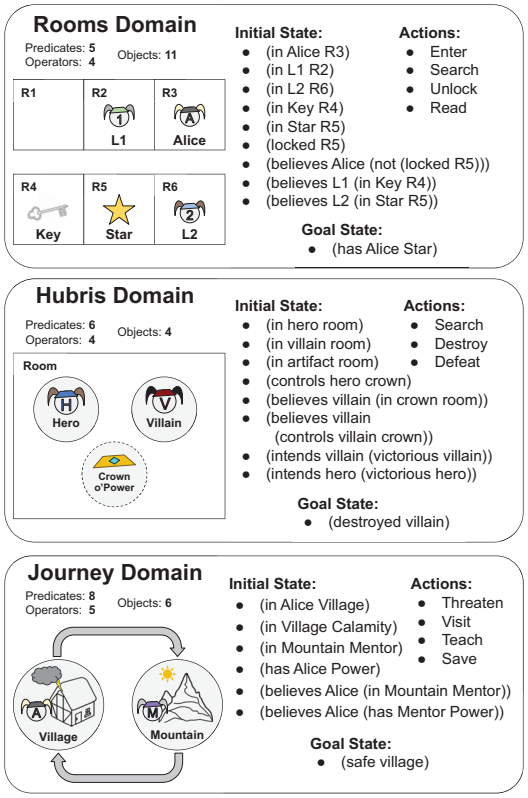

Figure 3: The domains created with our belief model.

to plan through. Regardless, our sample domains evidence that even without expression parameters there are interesting belief-based dynamics to explore.

Our model's theory-of-mind restriction is a limit to we can represent. Affording nested beliefs would make the resulting compilation grow exponentially and we suspect that not only would potential audiences finding challenging to interpret the resultant generated stories but so too would authors find it difficult to engineer the domains in the first place.

While we do not commit to a particular microtheory of how to update beliefs, our model requires all consenting agents to *explicitly* believe all preconditions to perform an action. However, as noted earlier, agents are allowed to be ambivalent about facts in the world. A different model of belief might restrict characters not to the actions that they believe are possible, but rather the actions they do not believe are impossible. How to do so is an open question.

Our model and compilation lay a groundwork to pursue each of these questions. We hope to continue building on this work in these directions.

8 Conclusion

Using our model of belief, it is possible to compile belief domains into intentional domains, with a polynomial increase to domain size. Intentional planners can then create stories where characters have different knowledge states and act according to their beliefs. These characters can fail,

learn from their mistakes, and share information with each other. In general, they act more closely to how a reader would expect them to rationally behave. We exemplify this with three domains, each of which rely on belief. With this, narrative planning has one more tool to create more interesting and believable stories.

References

Alchourrón, C. E.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic* 50(2):510–530.

Bates, J. 1994. The role of emotion in believable agents. *Communications of the ACM* 37(7):122–125.

Bratman, M. 1987. *Intention, Plans, and Practical Reason*. Cambridge, MA, USA: Harvard University Press.

Bylander, T. 1991. Complexity results for planning. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, 274–279.

Cardona-Rivera, R. E., and Young, R. M. 2019. Desiderata for a computational model of human online narrative sensemaking. In *Working Notes of the AAAI Spring Symposium on Story-enabled Intelligence*.

Da Costa, N. C. A., and French, S. 1989. On the logic of belief. *Philosophy and Phenomenological Research* 49(3):431–446.

Eger, M., and Martens, C. 2017a. Character beliefs in story generation. In *Proceedings of the 10th Workshop on Intelligent Narrative Technologies at the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Eger, M., and Martens, C. 2017b. Practical specification of belief manipulation in games. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2(3-4):189–208.

Genette, G. 1980. *Narrative Discourse: An Essay in Method*. Cornell University Press.

Haslum, P. 2012. Narrative planning: Compilations to classical planning. *Journal of Artificial Intelligence Research* 44:383–395.

Hoffmann, J. 2001. FF: The fast-forward planning system. *AI magazine* 22(3):57–57.

Lenat, D., and Guha, R. V. 1990. Cyc: A midterm report. *AI Magazine* 11(3):32–32.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. E. 1998. Pddl–the planning domain definition language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, New Haven, CT, USA.

Palacios, H., and Geffner, H. 2006. Compiling uncertainty away: Solving conformant planning problems using a classical planner (sometimes). In *Proceedings of the 21st National Conference on Artificial Intelligence*, 900–905.

Rao, A. S., and Georgeff, M. P. 1995. BDI agents: from theory to practice. In *Proceedings of the International Conference on Multiagent Systems*, 312–319.

Riedl, M. O., and Young, R. M. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research* 39:217–268.

Shirvani, A.; Farrell, R.; and Ware, S. G. 2018. Combining intentionality and belief: Revisiting believable character plans. In *Proceedings of the 14th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Shirvani, A.; Ware, S. G.; and Farrell, R. 2017. 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*.

Smith, G., and Whitehead, J. 2010. Analyzing the expressive range of a level generator. In *Proceedings of the Workshop on Procedural Content Generation in Games at the 5th International Conference on the Foundations of Digital Games*, 1–7.

Smullyan, R. M. 2012. *Forever Undecided*. Knopf.

Teutenberg, J., and Porteous, J. 2015. Incorporating global and local knowledge in intentional narrative planning. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*.

Thorne, B. R., and Young, R. M. 2017. Generating stories that include failed actions by modeling false character beliefs. In *Proceedings of the 10th Workshop on Intelligent Narrative Technologies at the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Van Ditmarsch, H.; van Der Hoek, W.; and Kooi, B. 2007. *Dynamic Epistemic Logic*. Springer Science & Business Media.

Wadsley, T., and Ryan, M. 2013. A belief-desire-intention model for narrative generation. In *Proceedings of the 6th Workshop on Intelligent Narrative Technologies at the 9th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Ware, S. G., and Young, R. M. 2014. Glaive: A state-space narrative planner supporting intentionality and conflict. In *Proceedings of the 10th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Ware, S. G. 2012. The Intentional Fast-Forward Narrative Planner. In *Proceedings of the 5th Intelligent Narrative Technologies Workshop at the 8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 57–62.

Wickler, G. 2011. Using planning domain features to facilitate knowledge engineering.

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.

Young, R. M. 2017. Sketching a generative model of intention management for characters in stories: Adding intention management to a belief-driven story planning algorithm. In *Proceedings of the 10th Workshop on Intelligent Narrative Technologies at the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.