**JAAMAS manuscript No.**
(will be inserted by the editor)

# Automated Narrative Planning Model Extension

**Julie Porteous · João F. Ferreira**
**Alan Lindsay · Marc Cavazza**

Received: date / Accepted: date

**Abstract** Interactive Narrative (IN) is an emerging application of automated planning, in which a planning domain is used to generate a consistent chain of narrative actions that constitute a plot structure. The task of creating narrative planning domains has been identified as a bottleneck which is hampering further development of the field. This stems from the difficulties faced by humans authoring such planning domains due to the need to provide the range of alternative content, such as actions, which are required to support the important properties of diversity and robustness. Narrative planning domains must be capable of generating diverse sets of narratives to ensure system replayability, and they must also be able to respond robustly in the face of narrative execution failure due to user interaction.

In this paper, we introduce a novel approach to the development of narrative planning domains based on the automatic expansion of a baseline planning domain through application of principled operations applied to both operators and predicates. We overview two such operations in this paper. The first of these, ANTON for ANTONymic operators, is based on the generation of contrary operators that can be invoked in the face of action failure, and whose structure is derived from a model of state transitions triggered by the original

Julie Porteous
School of Computing Technologies, RMIT University, Melbourne, Australia.
E-mail: julie.porteous@rmit.edu.au

João F. Ferreira
INESC-ID & Instituto Superior Técnico, Universidade de Lisboa, Portugal.
E-mail: joao@joaoff.com

Alan Lindsay
Heriott Watt University,
E-mail: alan.lindsay@hw.ac.uk

Marc Cavazza
University of Greenwich, London, UK.
E-mail: m.cavazza@greenwich.ac.uk

operator. Since the intention is for additional operators to be incorporated to the baseline, human-authored, domain model, the generated contents should be human-readable. This is achieved by using combined linguistic resources to access antonyms of predicates occurring inside operators, and parsing them from and into hyphenated units. The second operation, part of the same approach, referred to as THYPE, generates variants of operators by exploring type hierarchies for the main concepts associated with individual operators; the resulting concepts being fully integrated into a new operator's structure.

Our evaluation procedures are directly derived from the target properties of narrative planning domains, which are diversity and robustness, the former being measured through plot diversity and the latter, plot continuation following planned action execution failure. We used published narrative domains as datasets for these evaluations. Results demonstrated strong generative ability, and even more significant plan completion following action failure. Moreover, our evaluation demonstrates the synergic nature of ANTON and THYPE when applied simultaneously. Future work will focus on improving the integration of ANTON and THYPE operations through better balance between linguistic and conceptual hierarchies.

**Keywords** Virtual Agents · Narrative Planning · Domain Modeling

## 1 Introduction

Interactive Narrative (IN) has emerged as a novel application for Artificial Intelligence (AI) techniques, whereby virtual character behaviours can be generated, controlled and turned into media content. One AI technique of particular interest is the use of AI Planning to generate plans corresponding to character behaviours which are then presented to human audiences using, for example, 2D or 3D graphics or text (Riedl and Young, 2010; Porteous et al., 2010; Wang et al., 2018; Yao et al., 2019). Planning is a powerful technology for such applications as it has a good representational fit, helps ensure causality (important for appropriate virtual character behaviour) and offers flexible narrative generation possibilities (Young, 1999).

While most previous work on Planning has emphasised the search for optimal solutions in terms of plan length or resources, these objectives do not apply to IN. When using planning for IN, well-formedness remains essential to guarantee consistency and causality, yet optimality is replaced by other quality criteria which relate to the shape of the planning trajectory itself (Porteous et al., 2011). Another important aspect is the ability of narrative planners to produce a range of potential solutions to ensure diversity, or to compensate for plan failure following user interaction. Some researchers have proposed dedicated planning architectures providing specific support for narratively relevant phenomena, such as intentional planners (Riedl and Young, 2004). However, it has also been demonstrated that state-of-the-art mainstream planners can support IN, with the focus for the development of narrative generation system being on the formalisation of narrative planning domains.

Describing narrative planning domain models is a challenging knowledge elicitation and modelling task. This is reflected in the development of tools to assist in the modelling of domains such as the tools proposed by Vaquero et al. (2013) and by Dolejsi et al. (2018), as well as approaches to the automated creation of models (Cresswell and Gregory, 2011; Wu et al., 2007; Lindsay et al., 2017). These general challenges of domain modelling are further compounded for IN applications due to two issues which we refer to as *robustness* and *diversity*. Domain models for use in IN systems need to be *robust*, meaning that they must cover world state variations that result from user interactions making dynamic changes to the narrative world and also that they must be sufficient to enable appropriate recovery from such interactions, for example by plan repair or replanning. Domain models for use in IN systems also need to be *diverse*, meaning that they must allow for the generation of diverse sets of narrative plans, in order to ensure novelty, creativity and replayability.

The issues of robustness and diversity have created a paradoxical situation for entertainment applications: on the one hand, domain model content is provided by "authors" who may be less familiar with planning technologies; and on the other hand, the increased complexity of modelling narrative planning domains that ensure robustness and diversity requires planning expertise. Previous work on addressing this situation has included the development of authoring aids based on planning domain consistency checking and simulation (Pizzi and Cavazza, 2008) or visual authoring of meta-planning elements, which addressed plan diversity (Porteous et al., 2011), although not at the level of domain elicitation.

One specific difficulty, often associated with robustness and diversity of planning domains, is the need to handcraft more than is required by a planning formalisation of narrative actions and types of narrative objects. In other words, trying to anticipate the consequences of plan failure and the remedial actions or objects needed, or describing several potential alternatives instead of one. This observation has led us to explore the potential to assist the modelling process through automated extension of partially developed models, an approach that remains largely unexplored. Domain model extension has much to offer for the development of robust and diverse models, as the authoring process by which baseline narratives are translated into planning models faces the difficulty of encoding, beyond the definition of a baseline narrative, those actions and types of objects which can underpin narrative variants. In other words, what has been described as the paradox of interactive narrative, namely the need to reconcile narrative consistency with action substitution, also applies to its authoring component, a situation which has hitherto not received sufficient attention.

Thus the objective of our approach is to operationalise narrative action and object substitution during narrative plan authoring. To that effect, we have defined a number of high-level operations that can be applied to the default domain model forming part of a "baseline plan narrative", whose outcome is to produce alternative, human-readable domain model content. Importantly, these operations are inspired by the above mentioned requirements of robustness and diversity. These high-level operations detailed in this paper are:

*Generating Alternative Antonymic Actions*: to address this we introduce an operation inspired from negation that aims to generate contrary operators which can be used in narrative variants where a default narrative action cannot be executed. This transformation operates both on the operator's post-conditions structure and the labelling of constituent predicates. As the contrary operators are generated through analysis of antonym relations, this operation is referred to as ANTON.

*Generating Alternative Types of Narrative Objects*: we introduce an operation based on the generalisation of default narrative objects in the domain model. The aim is to generate alternative types of objects with which to specialise operators. These domain model extensions are generated via analysis of hypernym and hyponym relations and referred to as THYPE.

We introduced the operations ANTON and THYPE in earlier work, Porteous et al. (2015) and Porteous et al. (2020) respectively. In this paper we draw these operations together and discuss in the context of a unified framework.

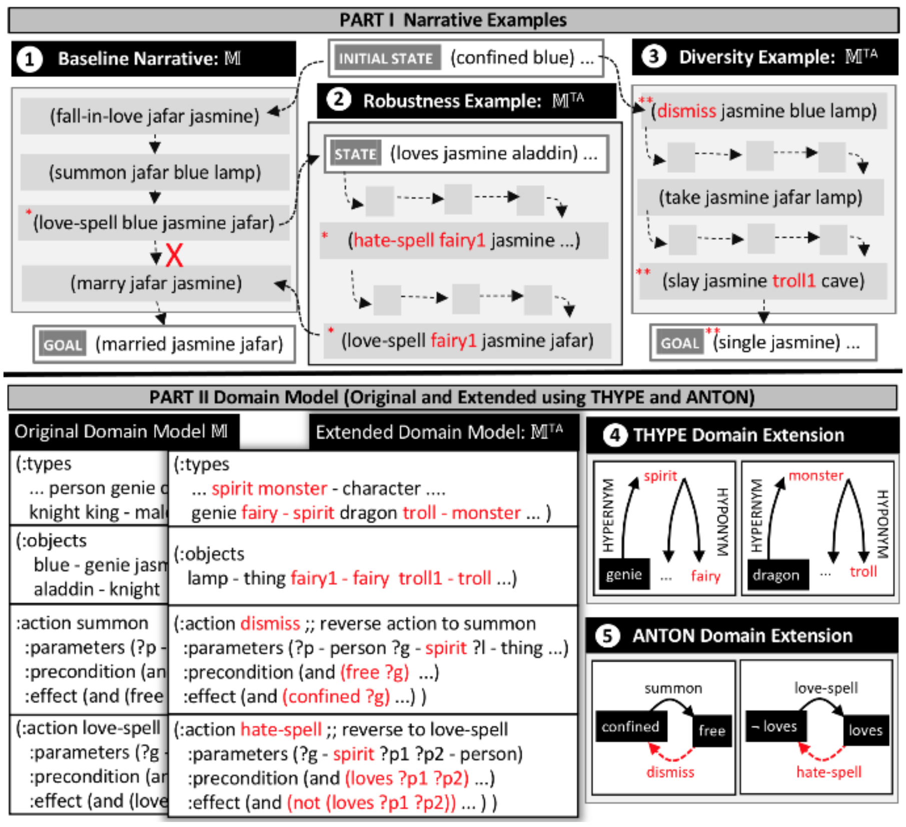
Both operations are influenced by linguistic and ontological approaches and can be seen as the application of antonymy, synonymy, hypernym and hyponymy relations to domain model content via the semantic labels that best describe them, although the transformations affect the larger domain model itself e.g. from the action label but affecting the entire operator structure.

An additional objective of our approach is for newly generated domain model extensions to be "human-readable", as the authoring and debugging process of narrative domain models is still meant to be supervised by a human author, as well as supporting exchange of planning domains between research groups. The above operations all involve generating new predicates, new named operators and types of narrative objects, taking as a starting point content labels from the default baseline domain model, and this process is heavily based on the use of linguistic resources.

The remainder of the paper is organised as follows. We begin in the next section with a motivating example which will be used as illustration throughout. In Section 3 we discuss closely related work and follow this in Section 4 with an overview of the planning framework and related background. Sections 5 and 6 present detailed overviews of ANTON and THYPE respectively. In Section 7 we define a modular and extensible framework which allows for the combination of multiple off-line generated domain model extensions. Section 8 presents the results of evaluation of ANTON and THYPE domain model extensions on a series of benchmark narrative domains. We conclude and discuss directions for future work in Section 9 at the end of the paper.

## 2 Motivating Example

Central to our approach to domain model extension is a rather traditional framework, albeit rarely presented as such, in which a narrative domain is de-


PART I Narrative Examples
1 Baseline Narrative: 𝕄
INITIAL STATE (confined blue) ...
3 Diversity Example: 𝕄TA
2 Robustness Example: 𝕄TA
(fall-in-love jafar jasmine)
(summon jafar blue lamp)
*(love-spell blue jasmine jafar)
X
(marry jafar jasmine)
GOAL (married jasmine jafar)
STATE (loves jasmine aladdin) ...
* (hate-spell fairy1 jasmine ...)
* (love-spell fairy1 jasmine jafar)
**(dismiss jasmine blue lamp)
(take jasmine jafar lamp)
** (slay jasmine troll1 cave)
GOAL **(single jasmine) ...
PART II Domain Model (Original and Extended using THYPE and ANTON)
Original Domain Model 𝕄
Extended Domain Model: 𝕄TA
4 THYPE Domain Extension
HYPERNYM
HYPONYM
spirit
genie
...
fairy
monster
dragon
troll
5 ANTON Domain Extension
summon
confined
free
dismiss
love-spell
¬ loves
loves
hate-spell

```
(:types
 ... person genie c
knight king - mal

(:objects
 blue - genie jasm
 aladdin - knight

:action summon
 :parameters (?p -
 :precondition (an
 :effect (and (free

(:action love-spell
 :parameters (?g -
 :precondition (ar
 :effect (and (love
```

```
(:types
 ... spirit monster - character ....
 genie fairy - spirit dragon troll - monster ... )

(:objects
 lamp - thing fairy1 - fairy  troll1 - troll ...)

(:action dismiss ;; reverse action to summon
 :parameters (?p - person ?g - spirit ?l - thing ...)
 :precondition (and (free ?g)  ...)
 :effect (and (confined ?g) ...) )

(:action hate-spell ;; reverse to love-spell
 :parameters (?g - spirit ?p1 ?p2 - person)
 :precondition (and (loves ?p1 ?p2) ...)
 :effect (and (not (loves ?p1 ?p2)) ... ) )
```

Fig. 1: Aladdin Narrative Examples (Part I): ① A baseline narrative generated using original domain model 𝕄; ② For the same initial state and goal, alternate actions and types of objects increase robustness, enabling recovery when user changes cause *love-spell** to go wrong (Jasmine ends up loving Aladdin rather than Jafar) and alternate actions and types are required for narrative continuation (the new type, *fairy*, casts spells so jasmine falls out of love with Aladdin and in love with Jafar); ③ Diverse narrative example, with alternate actions and types** leading to a very different ending (Jasmine puts the genie back in the lamp, slays the troll and lives happily, singly ever after).
Domain Model (Part II): shows original and THYPE and ANTON extended model; ④ alternate types of objects generated via Hypernym and Hyponym relations (e.g. *fairy* and *troll*); ⑤ alternate actions generated from antonym relations (e.g. *dismiss* and *hate-spell*). See text for details.

veloped around a "baseline narrative" often inspired from a well-known literary or filmic work. Thus, as motivation and to illustrate the sorts of domain model extensions that can be generated, we present an example using a baseline narrative domain model based on the well-known Aladdin folk tale. This domain model will also be used as a running example throughout the paper. This domain was selected due to its familiarity, previous use in narrative research (Riedl and Young, 2010; Teutenberg and Porteous, 2013) and suitable scope for experimentation. It features a range of different types of virtual characters and objects such as knights, princesses, dragons, genies, magic lamps and so on. We have created a baseline domain model for this domain with which it is possible to generate a range of narrative plans that see virtual characters travelling around, slaying dragons, falling in love, freeing genies and so on. As illustration, an initial narrative generated for this domain might proceed as shown in Figure 1 Part I ①, with narrative actions that include having King Jafar fall in love with Princess Jasmine, subsequently freeing Blue the genie from the magic lamp, ordering Blue to cast a love spell for Jasmine to fall in love with him, slaying a dragon and marrying to conclude the narrative variant. The figure illustrates domain model extensions that address both robustness and diversity as follows:

*Robustness:* In interactive narrative systems user interaction can change the state of the planning world and hence necessitate re-planning or repair (as in Young's "gun" example (Young, 1999)). Thus, domain models used in such systems must be robust: capable of responding to dynamic changes to the story world and continuing a narrative through to the intended ending regardless of user interaction. One way in which robustness can be promoted in a domain model is through the inclusion of alternative actions and types of narrative objects which go beyond those required for a baseline narrative, and which enable the generation of alternative courses of action from points of narrative execution "failure" onwards. As illustration, Figure 1 shows user interaction changing the state of the story world so that Princess Jasmine loves Aladdin and not Jafar (highlighted ** in ① and ②). The effect of this user interaction is to render the original goal impossible to reach. Our insight is that capturing alternatives for narrative-world remediation would make the model more robust and enable narrative continuation.

The extensions we explored are:

- ANTON generated actions capturing reverse state transitions. In the figure this is illustrated with actions such as *hate-spell* (see ⑤ in Figure 1). This action captures the state transition from love to hate (the reverse of *love-spell* with an appropriate label for the content). The resultant action allows generation of a narrative with a spell cast so that Jasmine hates Aladdin, allowing for the continuation of the narrative to the original goal as shown.
- THYPE generated alternative types of virtual characters and narrative objects that can be used to instantiate narrative actions. In this exam-

ple, Figure 1 shows the generation of the type *fairy* which allows for regeneration of a narrative which has the fairy casting a *love-spell* which makes Jasmine fall back in love with Jafar and allowing for continuation through to the original goal (see ② and ④ in Figure 1).

*Diversity:* This relates to the ability of a narrative domain model to allow the generation of diverse sets of narratives to allow for system replayability. Domain models can be made more diverse with the addition of both ANTON and THYPE extensions as the addition of new narrative actions, and new types of virtual characters and narrative objects all open up a wider range of narrative possibilities. An important advantage of automating the process of domain model extension is the requirement for generated narratives to be consistent: domain model consistency is something which is enhanced via system generated extensions as it removes the possibility of human error. In the example shown in Figure 1 Part I ③, in order for an Interactive Narrative system to produce multiple story variants, the domain model requires sufficient content to allow for generation of story lines that deviate from the baseline. As illustration, consider the "baseline narrative" shown in the figure where the king Jafar falls in love with Princess Jasmine and frees Blue the genie from the magic lamp, gets Blue to cast a love spell so that Jasmine falls in love with him so they can marry at the end of the narrative. A very different narrative leading to a different narrative goal is shown in Part I ③ of the figure. In this narrative variant, Princess Jasmine puts the genie back in the lamp, takes the magic lamp, travels to the cave, slays the `troll` and lives happily, single, ever after. This variant uses ANTON generated narrative action *dismiss*, and an instance of the THYPE generated alternative type of virtual character *troll*, called *troll1*[1].

These automatic suggestions simplify the authoring process while retaining the combinatorial properties of plan-based generation (as opposed to branching narratives). The underlying idea is to use natural language labels of planning domain elements to generate related concepts using lexical relations between these labels and other terms in online lexical resources. Hence our approach identifies plausible candidate types of virtual characters and narrative objects based on information about semantic relations such as *hypernym* (more general) and *hyponym* (more specific) from linguistic resources (as shown for `fairy` and `troll` in Figure 1). This is predicated on the fact that linguistic hierarchies such as ConceptNet (Liu and Singh, 2004; Speer et al., 2017), can be used as partial ontologies and for common sense reasoning.

## 3 Related Work

Automated domain model extension and creation is a topic of current interest in the AI planning community, in part due to increasing awareness that building domain models is challenging, time consuming and an obstacle to the

[1] New instances of THYPE generated types are named using as described in section 6.

further fielded application of planning technology (Zhuo and Kambhampati, 2013). Recent work in this area has been aimed at learning planner action models from correctly observed plan traces (Amir and Chang, 2008; Cresswell and Gregory, 2011; Zhuo et al., 2010). However, this work has limited application for narrative domains which do not share the same consistency and alignment with real-world domains as do more traditional planning domains such as "logistics" or "rovers".

Also, a growing body of work has considered the implications of incomplete models. Cashmore et al. (2015) and Young et al. (2017) extend planning models during execution in order to incorporate initially unknown information. In the approach of Cashmore et al. (2015), an ontology captures the objects, with their types and attributes, and underpins the construction of the planning model. When a new object is detected by the system, the object is added to the ontology. The object can be used in replanning, which can lead to shorter plans. Moreover, Cashmore et al. (2015) assume that the types of objects are known a priori, whereas a more general framework is developed by Young et al. (2017, 2016), where the surrounding objects are used to build a context and a general image labeller is used to identify objects. *Model-lite* planning (Yoon and Kambhampati, 2007; Kambhampati, 2007) is also motivated by the impracticality of always completely defining a planning model upfront. These works do consider incompleteness within the planning framework itself, but differ with respect to focus (e.g. such things as reducing planning length) and nature of the target extensions which are less concerned with issues such as diversity and hence are not as applicable in Interactive Narrative.

Also related is the work on excuse generation (Göbelbecker et al., 2010) which was later extended to planning task revision (Herzig et al., 2014) which considers how a given problem description can be updated when a plan solution is not possible. We observe that the approaches of Göbelbecker et al. (2010) and Herzig et al. (2014) might also be used to generate extended domain models: creating new content and extending the model on the fly at run-time. In contrast, we present a process for determining how a domain model can be extended during off-line processing, which we see as being complementary to on-line model extension.

We observe that an approach adopted in some Interactive Narrative systems is the use of experience managers to ensure that aesthetic properties of the narrative are maintained in the presence of user interaction. Of relevance to the work we present in this paper are plan-based approaches, via *Narrative Mediation* (Riedl and Young, 2006; Robertson and Young, 2019). The mediation process is one of reasoning about possible user interactions, i.e. alternatives to system planned actions, that would necessitate some repair or replanning in order to guide the narrative to a suitable alternative ending. In the work we present here, our goal is to help develop rich narrative planning models and thus could be used in combination with these mediation approaches to help develop suitable models for the mediation planning process.

In the area of narrative generation there has been increasing work aimed at automated creation of narrative content. A popular approach has been the gathering of story elements via crowdsourcing, an approach which can yield abundant content. For example, the SCHEHERAZADE system of Li et al. (2013) employ this approach to acquire typical story elements which can be assembled as plot graphs and used in a process of story generation. With SCENARIOGEN (Sina et al., 2014) crowdsourcing was used to gather a database of scenarios of everyday activities and likely replacements for use within a serious game context. Orkin and Roy (2012) use a crowdsourcing approach for the hand annotation of logs from user sessions with the *Restaurant Game* for subsequent use in automating character interactions with human participants in a dialogue-based narrative setting. An alternate to crowdsourcing aims to obtain narrative content through mining of weblogs and story corpora: the SAYANYTHING system of Swanson and Gordon (2012) selects narrative content on-the-fly from a corpora of weblogs in response to user text-based interaction; whilst the approach of McIntyre and Lapata (2009) attempts to generate narratives using knowledge mined from story corpora for a particular genre. Our automated mechanism for diversification of narrative planning actions through antonymy differs from these approaches in that it aims to discover alternatives around baseline plots (which themselves could be drawn from existing linear narratives).

Another area of related work to our antonym-based approach is that aimed at the computational creation of content for use in applications requiring narrative generation. Examples of this include the use of conceptual blending in the creation of novel artefacts for use in narrative generation (Li et al., 2012; Harrell et al., 2010) and visual narrative generation (Pérez y Pérez et al., 2012).

We observe that the general issues that surround the creation of planning models have led to the development of a number of tools to assist the process such as the post-design framework of Vaquero et al. (2013) and the PDDL plug-in to VSCode of Dolejsi et al. (2018), along with approaches to support the maintenance of existing models (Bryce et al., 2016) and other approaches that aim to automate the process entirely, such as the LOCM family of algorithms (e.g. (Cresswell et al., 2009) and (Cresswell and Gregory, 2011)), the FRAMER approach of Lindsay et al. (2017), Janghorbani et al. (2019) and Walsh and Littman (2008). In our work we aim to automatically extend existing domain models, thus it could be used in combination with these approaches to extend them with the potential to enhance robustness.

As far as we are aware, the operations ANTON and THYPE which we have introduced in earlier work, Porteous et al. (2015) and Porteous et al. (2020) respectively, and which we draw together in this paper, are the only approaches that have sought to automatically generate extensions to partially developed narrative domain models.

## 4 Planning Framework/Background

In this work we use the deterministic planning language of PDDL (McDermott et al., 1998), which is separated into two parts: the domain model, a definition of the problem domain that defines the world and its behaviours; and an explanation of the specific problem to be solved within that world. A domain model is a tuple, $\mathbb{M} = \langle \mathbb{A}, \mathbb{P} \rangle$, defining the sets of actions, $\mathbb{A}$, and predicates, $\mathbb{P}$. A parameterised action, $\mathcal{A} \in \mathbb{A}$, is represented by a unique symbol (action name), a list of typed variables (parameters), and the action's pre and post-conditions. The pre-condition is a formula that must hold for the application of the action and the post-condition defines the action's positive and negative effects. The pre-conditions use first order predicate logic and we assume these are conjunctions of predicates and their negations. Similarly, we also assume post-conditions are conjunctions of predicates and their negations. Any variables used in the actions (i.e., in the pre- or post-conditions) must be included in the parameters of the action. Examples of parameterised actions are presented in Figure 2. A problem model is a tuple, $\mathbf{P} = \langle O, s_0, g \rangle$, defining the typed objects, $O$, the initial state, $s_0$ and the goal function, $g$. An action is called a *ground action* if all its parameters are replaced with objects from $O$.

A solution to the planning problem is a sequence of ground actions that when applied to the initial state lead to a state that satisfies the goal. We say a goal is *reachable* from a state, $s$, if there exists such a sequence of ground actions and otherwise, $s$ is a *deadend*.

A library of language extensions, $\mathbb{E}$, consists of functions, $e : \mathbb{M} \mapsto \mathbb{M}^e$, such that the model, $\mathbb{M}$, is extended through some process into a richer model, denoted $\mathbb{M}^e$. For example, a language extension function, `add-no-op`, would extend a model with a new operator, called `no-op`, with an empty condition and effect. The main contribution of this work are two extension functions ANTON and THYPE which are described in the following sections.

## 5 ANTON: Generating Alternative Antonymic Actions

In this section we overview a domain model extension operation for the generation of alternative narrative actions starting from a baseline domain model. The actions in a baseline narrative domain model can be seen as specifying the properties that different types of domain objects can occupy and the ways in which these can be changed. Thus the contrary is also true: the narrative actions specify those properties that cannot be changed and those transitions that cannot be reversed without compromising the narrative experience (this is, of course, strongly dependent on narrative genre). For example, in the Aladdin story as shown in Figure 1, genies can be trapped in a magic lamp (as opposed to an ordinary one) and can be either trapped or freed (by being summoned from the lamp). Princesses are beautiful and can be single or married. However different variants of the story require contrary transitions and prop-

**ACTIONS**

```
(:action fall-in-love :parameters (?m - male ?p - princess ?l - location)
   :precondition (and (single ?m) (beautiful ?p) (alive ?m) (alive ?p) ...)
   :effect (and (loves ?m ?p)))
(:action summon :parameters (?p - person ?g - genie ?t - thing)
   :precondition (and (confined ?g) (magic ?t) (has ?p ?t) (at ?p ?l))
   :effect (and (at ?g ?l) (controls ?p ?g) (not (confined ?g))))
(:action love-spell :parameters (?g - genie ?p1 ?p2 - person)
   :precondition (and (not (confined ?g)) (not (loves ?p1 ?p2)) (alive ?g) ...)
   :effect (and (loves ?p1 ?p2)))
(:action marry :parameters (?m - male ?p - princess ?l - location)
   :precondition (and (loves ?m ?p) (loves ?p ?m) (single ?m) (single ?p) ...)
   :effect (and (married ?m ?p) (married ?p ?m) (not (single ?m)) (not (single ?p))))
(:action slay :parameters (?k - knight ?m - monster ?l - location)
   :precondition (and (at ?k ?l) (at ?m ?) (alive ?k) (alive ?m))
   :effect (and (not (alive ?m))))
```

**TYPES**

```
agent location thing - object      |  knight king - male
person genie dragon - agent        |
princess male - person             |
```

Fig. 2: Selected actions and types from the Aladdin Domain. Actions specify ways the story world can change e.g. males can fall-in-love with a princess if certain things hold in the story at that point, such as being beautiful. Types form a hierarchy which plays a role in the propagation of conditions e.g. a knight can slay a monster but only genies can be summoned from a lamp.

erties to allow for such things as ugly princesses becoming beautiful, married princesses becoming single and genies being put back into magic lamps.

These contrary properties and transitions are important since they may be required in order to continue the presentation of an ongoing narrative in a dynamic environment, as for the example in Figure 1. Hence our approach to content creation seeks to identify core narrative elements in the input domain model whose negation impacts story progression (section 5.1) and then to use these as candidate transitions for content creation (section 5.2).

### 5.1 State Transition Analysis

Starting with an input domain model a set of state transition rules are constructed which represent the partial state transitions that are possible for each of the different types of objects in the model. This is related to the identification of Finite State Machines and transition rules with TIM (Fox and Long, 1998) and analysis of Domain Transition Graphs in FAST DOWNWARD (Helmert, 2006). Building on (Fox and Long, 1998), the transition rules are specified in terms of *properties*, where a property is a predicate subscripted by a number between 1 and the arity of the predicate, so that every predicate of arity $n$ defines $n$ properties. For typed PDDL domains we extend this to refer to the type of a property: for a property with subscript $i$ its type corresponds to the type of the argument in predicate position $i$. For example,

*(controls ?p - person ?g - genie)*,

defines the properties *controls*$_1$ and *controls*$_2$, of type *person* and *genie* respectively. The analysis requires special treatment for negative preconditions (Cresswell et al., 2002). Predicates that are used as negative preconditions (e.g., *loves* in the *love-spell* action in Figure 2) are each paired with a new predicate that represents the negative of the predicate. We use the predicate name with a prefix '$\neg$' to denote these new predicates (e.g., *loves* is paired with $\neg$*loves*). As a consequence, these predicates lead to the construction of negative properties (e.g., $\neg$*loves*$_1$), allowing the approach to operate correctly in the case of negative preconditions.

Transition rules take the form $E \Rightarrow S \rightarrow F$ (read as *"E enables the transition from S to F"*) and where the three components are bags of zero or more properties built for each action in the domain model and for each type of object in the actions' parameters. Empty bags are denoted $\{\}$ in the following examples. The rules are constructed as follows[2]:

*E (enablers)* pre-condition properties *unchanged* by the action
*S (start)* pre-condition properties *deleted* (i.e. *lost*) by the action
*F (finish)* properties that are *achieved (gained)* by the action

As illustration, consider the Aladdin domain (Figure 2), and the identification of transition rules for the action *love-spell.* The action parameters **?g - genie ?p1 ?p2** - person are considered as follows:

**?g** The action contains only the property $\neg$*confined*$_1$ of type *genie.* This is unchanged by the action and is added to *E.* There are no other properties hence *S* and *F* are empty. Since this transition rule is empty (does not describe a state change as described above), it is discarded.

**?p1** The properties *alive*$_1$, $\neg$*loves*$_1$ and *loves*$_1$ are of type *person.* The property *alive*$_1$ is required and not changed by the action and is added to *E*, $\neg$*loves*$_1$ is deleted by the action and added to *S*, and *loves*$_1$ is achieved by the transition so is added to *F*. The resulting rule that is output is as follows: $\text{alive}_1 \Rightarrow \neg\text{loves}_1 \rightarrow \text{loves}_1$ (see rule #1 in Figure 3).

**?p2** The properties *alive*$_2$ (enables), $\neg$*loves*$_2$ (deleted) and *loves*$_2$ (achieved) are identified and added to *E*, *S* and *F*. The resulting rule output is: $\text{alive}_2 \Rightarrow \neg\text{loves}_2 \rightarrow \text{loves}_2$ (see rule #2 in Figure 3).

Hence the analysis of the action *love-spell* yields 2 transition rules for type *person* and none for type *genie* (rules #1 and #2 in Figure 3 which lists all the transition rules constructed for the selected actions shown in Figure 2).

[2] Note: rules can contain empty components, however any transitions where both *S* and *F* are empty are ignored since they do not describe any change of state for that type of object.

| Type | Action | State Transition Rules: $E \Rightarrow S \rightarrow F$ | | | | | Rule # |
|---|---|---|---|---|---|---|---|
| person | love-spell | alive$_1$ | $\Rightarrow$ | $\neg$loves$_1$ | $\rightarrow$ | loves$_1$ | 1 |
| | | alive$_2$ | $\Rightarrow$ | $\neg$loves$_2$ | $\rightarrow$ | loves$_2$ | 2 |
| | summon | has$_2$ at$_1$ | $\Rightarrow$ | { } | $\rightarrow$ | controls$_1$ | 3 |
| monster | slay | at$_1$ | $\Rightarrow$ | alive$_1$ | $\rightarrow$ | { } | 4 |
| male | marry | loves$_1$, | $\Rightarrow$ | single$_1$ | $\rightarrow$ | married$_1$ | 5 |
| | | loves$_2$, .. | | | | married$_2$ | |
| | fall-in-love | alive$_1$ | $\Rightarrow$ | $\neg$loves$_1$ | $\rightarrow$ | loves$_1$ | 6 |
| princess | marry | loves$_2$ | $\Rightarrow$ | single$_1$ | $\rightarrow$ | married$_2$ | 7 |
| | | loves$_1$... | | | | married$_1$ | |
| | fall-in-love | beautiful$_1$ | $\Rightarrow$ | $\neg$loves$_2$ | $\rightarrow$ | loves$_2$ | 8 |
| | | at$_1$ alive$_1$ | | | | | |
| genie | summon | alive$_1$ | $\Rightarrow$ | in$_1$ | $\rightarrow$ | controls$_2$ | 9 |
| | | | | confined$_1$ | | at$_1$ | |
| | | | | | | $\neg$confined$_1$ | |

Fig. 3: Example State Transition Rules for the Aladdin Domain. Rules represent how the effects of narrative actions can change the partial states of different types of objects. Rules take the form $E \Rightarrow S \rightarrow F$ where: $E$ is a set of properties that enable the transition: $S$ are the properties given up by the transition; and $F$ are properties acquired by the transition. The negative of a property $x$, used for analysing negative preconditions, is shown as $\neg x$. As an example, for the type *person* the action *summon* requires that the person initially *has* (a lamp in which the genie is confined) and is *at* (a location). Note that this transition is possible for all sub-types of type *person* (i.e. the types *knight, king* and *princess*).

## 5.2 Identification of Core Transitions

Once constructed the transition rules are searched to identify contrary transitions and properties which suggest core candidate transitions for action generation. We consider each of these in turn in the following subsections.

### *5.2.1 Candidate Actions from Contrary TRANSITIONS*

For any type of object for which a transition is specified in the rules we postulate that the contrary transition has a natural interpretation in the domain and that we would also expect to find it in the set of transitions. Hence, any transition whose contrary does not appear in the set of rules is proposed as a candidate for alternative action generation.

As an example from the Aladdin domain, consider transition rules 1 and 2 for action *love-spell* and rule 6 for *fall-in-love* shown in Figure 3. For any

| Linguistic Resources | Antonyms for "marry" dissociate | **divorce** | separate |
|---|---|---|---|
| Merriam-Webster | 1 | **2** | 1 |
| BigHuge Thesaurus | 0 | **0** | 0 |
| Power Thesaurus | 1 | **3** | 1 |
| Totals | 2 | **5** | 2 |

Fig. 4: Antonym selection for the action marry (only those returned by multiple providers are shown). The weights for each antonym are listed below it. They are summed and the highest ranked is selected. In this case *divorce* is selected.

*person* (or sub-type: *king, knight, princess*) the transition $\neg loves \rightarrow loves$[3] can be made via *love-spell* and in addition for any *male* (or sub-type *king, knight*) the transition can also be made via the action *fall-in-love*. However for all these types of objects the contrary transition $loves \rightarrow \neg loves$ is missing, i.e. there is no way for a love spell to be undone or for someone to fall out of love. Hence the actions *fall-in-love* and *love-spell* are flagged as candidates for alternative action generation. All of the transition rules are traversed and the names of any actions for which contrary transitions are missing are added to the set of candidates for generation. For the rules in Figure 3 the set of candidates after this analysis is:

$$\{\textit{love-spell, summon, slay, marry, fall-in-love}\}.$$

### *5.2.2 Candidate Actions from Contrary PROPERTIES*

The rationale for identifying these candidates is based on consideration of possible causes of failure of narrative plans when intended action execution is affected by changes in the environment. From the transition rules shown in Figure 3 we observe that some properties are only ever required as enablers and that such properties cannot be changed by the effects of narrative actions (examples of this are $\text{beautiful}_1$ for type princess and $\text{alive}_1$ for person). For such properties, we postulate that their contrary has a natural interpretation in the domain and hence propose the gaining and losing of such properties as candidate actions. The natural validity of such transitions becomes more apparent if one considers typical fairy tale situations in which characters are killed or forced into eternal sleep, or their appearance is changed. For the rules in Figure 3, the properties $alive_1$ and $beautiful_1$ are identified. Hence the set of candidates added after this analysis is:

$$\{\textit{gain(alive), lose(alive), gain(beautiful), lose(beautiful)}\}.$$

[3] For brevity we omit the numeric subscript on the property, however this example applies equally to both $loves_1$ and $loves_2$.

## 5.3 Antonymic Labeling

In order to ensure human readability of the automatically generated PDDL structures we generate labels for new domain content based on antonyms of the names of actions and properties in the original domain. The use of antonyms is justified since candidate actions specify transitions between partial states that represent opposition such as, (*married, single*) or (*beautiful*, $\neg$*beautiful*) and these are used as part of a heuristic approach to action generation which requires transition to a partial state to include the opposite state as a precondition (this is discussed further in section 5.4).

Labels are generated using antonyms drawn from a range of publicly available lexical resources (as in (Lin et al., 2003; Turney, 2008)): multiple resources are used since robustness has been shown to be improved when using this strategy (Nazar and Janssen, 2010). In our experiments we used the following on-line lexical resources: WordNet (WN) (Fellbaum, 1998) for synset expansion of queried words, Merriam-Webster (MW), BigHuge Thesaurus (BT), and PowerThesaurus (PT)[4].

### *5.3.1 Antonyms for Single Words*

To generate antonyms, ANTON sends strings, such as a verb representing an action name, to each of the linguistic resources. The output of these differs slightly: WN organises words into synonym groups, called Synsets, which ANTON uses initially to expand the queried word; MW and BT are organised by definition and return an HTML page with tagged content which ANTON parses, using the tags, to extract the antonyms for each definition of the expanded query; PT provides a page specially for the antonyms of words and lists these with an associated score based on the frequency of use of the antonym in texts and is updated to reflect user opinion.

ANTON maps the response of each provider into a list of candidate antonyms with associated integer values. This value signifies the likelihood of the candidate being a desired antonym. For MW, BT and WN this value is based on the number of definitions that are antonymic with the candidate. For PT the value is a rating of the general use of the candidate in texts. The weights of the returned candidates are summed to obtain a single weight for each candidate, ignoring sources that did not list it (a voting strategy) to get a measure of the likelihood that the word is a strong candidate. The candidate(s) with the highest weight is selected to be used for the label. Any ties are broken by ordering the candidates alphabetically and selecting the first to ensure deterministic selection. As an illustration, Figure 4 shows the process of generating a label for the contrary action to *marry*, which in this case is *divorce*. We note that divorce is only the contrary of marry if the pair are already married, otherwise the contrary would be something like “refuse to marry” or “not marry”

[4] Resources available on-line: WordNet: `https://wordnet.princeton.edu`, Merriam Webster: `http://www.dictionaryapi.com`, Power Thesaurus: `http://www.powerthesaurus.org`, Big Huge Thesaurus: `http://words.bighugelabs.com`

① Action from Missing Transitions

```
(:action marry
 :parameters (?m-male ?p-princess)
 :precondition (and
   (single ?m) (single ?p)
   (loves ?m ?p) (loves ?p ?m) ...)
 :effect (and
  Ⓐ (married ?m ?p) (married ?p ?m)
  Ⓑ (not (single ?m)) (not (single ?p))))
```

```
(:action divorce
 :parameters (?m-male ?p-princess)
 :precondition (and
   (married ?m ?p) (married ?p ?m) Ⓐ
   (motivated-to-divorce ?m ?p) ...)
:effect (and
(single ?m) (single ?p) Ⓑ
(not (married ?m ?p)) ...) Ⓐ
```

② Action from Missing Properties

```
:action become-beautiful
  :parameters (?p - princess)
  :precondition (and (not (beautiful ?p)))
  :effect (and (beautiful ?p)))
```

```
(:action become-ugly
  :parameters (?p - princess)
  :precondition (and (beautiful ?p))
  :effect (and (not (beautiful ?p))))
```

③ New Predicates: e.g. replace (not (beautiful ?p) with (ugly ?p)

```
(:action become-beautiful
  :parameters (?p - princess)
  :precondition (and (ugly ?p))
  :effect (and (beautiful ?p) (not (ugly ?p))))
```

Fig. 5: Example of Action Generation for candidate actions. ① For Missing Transitions new actions are generated using the original action as a template with original action (marry) and newly generated action (divorce); the name is generated using ANTON naming; the pre-conditions are those predicates achieved by the action Ⓐ plus an enabling condition; the positive effects are the pre-conditions of the initial action Ⓑ. ② For Missing Properties new actions are generated for: gaining the property, *become-beautiful*; and gaining the contrary property (i.e. losing the property), which is named using ANTON, in this case the action *become-ugly*. For more detail refer to text.

(consider Beauty and the Beast). Our heuristic approach, where transition to a state requires the opposite state as a pre-condition, justifies the use of antonymy. Relaxing this assumption could form the basis of additional action creation but is beyond the scope of the work presented in this paper.

Antonymic naming with ANTON offers the possibility of replacing predicates that appear as negative pre-conditions with new predicates that exactly complement their positive use, as in (Gazen and Knoblock, 1997). For example, consider the negated predicate that appears in the pre-conditions of the action *become-beautiful* in Figure 5. If this translation were to be applied to the domain then this would be replaced with the new predicate so that the action pre-condition becomes *(ugly ?p-princess)*. Also, the post-conditions require extension to handle the gaining and losing of the property. As illustration the updated action is shown in Figure 5 ③. We observe that addition of predicates via this translation may further enhance domain model surveyability.

### *5.3.2 Antonyms for Multiword Expressions*

Most of the actions and properties to be renamed in the Aladdin domain are single words with the exception of *fall-in-love* and *love-spell* which have been named using multiword expressions. Dictionary providers perform poorly with multiword expressions (Sag et al., 2002) and hence these expressions require additional processing.

To find antonyms of multiword expressions we pair the individual words with their antonyms as predicted by the method described above (except for the case of prepositions, whose candidates we lookup in a small lexicon). For example, hate-spell yields [(love, hate), (spell, unspell)]. Candidate antonymic expressions are then generated by enumerating the combinations other than the original (e.g. (love-unspell, hate-spell, hate-unspell). The most likely candidate is selected by estimating probabilities with an n-gram language model (BerkleyLM; (Pauls and Klein, 2011)) trained on text extracted from weblogs. In this example the model estimates that hate-spell is most likely ($P = 3.1 \times 10^{-5}$) while love-unspell and hate-unspell are both least likely ($P = 1.0 \times 10^{-100}$). Hence *hate-spell* is returned as the antonym for the multiword expression *love-spell*.

## 5.4 Generating New Actions

### *5.4.1 Actions from Contrary Transitions*

For actions representing "missing" contrary transitions new actions are generated from the original action named in the set of candidates (as discussed in section 5.2.1). Here we illustrate this process with reference to the Aladdin action *marry* and the generation of its contrary, as shown in Figure 5, which illustrates the process of extending the set of default actions to create potential for generating new stories. For the name of the new action the ANTON generated label is used: in this case the string *marry* yields the label *divorce*. The parameters for the new action are the same as for the original action. The rationale for this being that the same objects will participate in the action and the resulting transitions.

There are two aspects to the construction of the action pre-conditions. Firstly the set of predicates which are *achieved* by the original action are added to the pre-conditions of the new contrary action: recall from the discussion in section 5.3 that this heuristic approach justifies the use of antonymy. For the action *divorce* these are the predicates which require that the couple are already married, labelled A in Figure 5. The second aspect of the pre-condition construction is the inclusion of any *enabling* conditions required for the transition, however the precise nature of these condition(s) is unclear: for example, are they the same or different to those for the original action? The Aladdin domain exhibits instances of both possibilities: the enabling condition for *divorce* is unlikely to be the same as for *marry* whereas it may well be the case that the same enablers might be required for *summon*-ing the genie from the lamp and putting it back in. Our solution, for the purposes of experimental evaluation and automation of our approach, was to introduce a generic enabling condition capturing the requirement of the necessary motivation to participate in the action formed by prefixing the ANTON generated action name with *motivated-to* and introducing an additional action to ensure the enabling condition can be achieved (as illustration see the predicate *motivated-to-divorce* in Figure 5).

The post-conditions for the new action come directly from the original action: the positive post-conditions are the predicates that are *deleted* by the original action (e.g. *(single ?m)* is deleted by *marry* and achieved by *divorce*); and the negative post-conditions are the predicates that are *achieved* by the original (e.g. *(married ?m ?p)* is achieved by *marry* and hence deleted by *divorce*).

### *5.4.2 Actions from Contrary Properties*

For candidate actions that represent the gaining or losing of a property then new actions are generated to enable these transitions. Here we illustrate this process for the candidate actions to gain and lose the property *beautiful* (discussed earlier in section 5.2.2).

For actions representing gaining a property the name is the property label prefixed with *become* and a single parameter variable of the type of the property. The action pre-condition is the negation of the property, since this action represents the transition from a partial state where the property is absent. Since the property is being gained, there is a single positive post-condition, the property itself. This is illustrated for action *become-beautiful* in Figure 5.

For actions representing losing a property (or the gaining of the contrary property) the name is generated from the ANTON label prefixed with *become*, there is a single parameter variable of the appropriate type for the property, the pre-condition is the original property and there is a single negative post-condition representing the losing of the property (as illustration, action *become-ugly* is shown in Figure 5).

As discussed earlier (section 5.3) the generation of labels for created content affords the possibility to replace predicates that appear as negative action pre-conditions with new predicates that exactly complement their use. For example, if the contrary predicate *(ugly ?p-princess)* was introduced to the domain model then this could replace the negative pre-condition *(not (beautiful ?p))* in the action *become-beautiful*. In addition the negative effects of the action would need to be extended to include *(not (ugly ?p))* reflecting the fact that this property is lost as a result of the transition.

**Algorithm 1** THYPE: generate alternate types of objects.

```
1: function THYPE($t$, $A$)
2:     $ss \leftarrow$ GETRELEVANTSYNONYMS($t, A$)
3:     for all $s \in ss$ do
4:         $he \leftarrow he +$ GETHYPERNYMS($s$, HYPERNYM_LEVEL)
5:     end for
6:     for all $h \in he$ do
7:         $ho \leftarrow ho +$ GETHYPONYMS($h$)
8:     end for
9:     $ho \leftarrow$ FILTERCOMMONSENSE($ho, t$)
10:    return $ho$
11: end function

12: function FILTERCOMMONSENSE($ho$, $t$)
13:    for all $h \in ho$ do
14:        if ISRELATEDCN($h, t$) $\wedge \neg$ISANTONYMCN($h, t$) $\wedge$
15:            $\neg$ISDISTINCTFROMCN($h, t$) then
16:            $ho' \leftarrow ho' + h$
17:        end if
18:    end for
19:    if LENGTH($ho'$) $>$ MAX_NUM then
20:        for all $h \in ho' \wedge \neg$ISACN($h, t$) do
21:            $ho' \leftarrow ho' - h$
22:        end for
23:        if LENGTH($ho'$) $<$ MIN_NUM then
24:            $ho' \leftarrow$ GETTOPRELATED($ho$, MAX_NUM)
25:        end if
26:    end if
27:    return $ho'$
28: end function
```

## 6 THYPE: Generating Alternative Types

In this section we overview a second domain model extension operation for automatically generating plausible additional types of virtual characters and narrative objects with which to extend an existing narrative planning domain model. The operation is based on the observation that the natural language labels used to name planning domain elements can be used to generate related concepts using lexical relations between the predicate labels and other terms in online lexical resources. Thus the aim is to identify plausible alternative types from online linguistic resources, via traversal of *hypernym* and *hyponym* semantic relations (where hypernyms give a more general semantic term and hyponyms give a larger set of more specific terms) in combination with commonsense reasoning to maximize the relevance of the suggested additional types. We refer to this as THYPE, to denote the extraction of **t**ypes from **hy**ponyms and hy**pe**rnyms.

An outline to the generation process is shown in Algorithm 1, with THYPE being the main function. Input is some named type of object, $t$, in the planning domain model and the set of action names in the current domain model, $A$. The first part of the algorithm consists in getting the set of hypernyms for type $t$ using online resources (lines 2–5). Our implementation uses WordNet 3.0 (Fellbaum, 1998), although approaches such as Word2Vec (Sahin, 2017) can also be used to extract resources. We chose WordNet because it is compatible with a larger variety of semantic domains, without having to collect text samples for each single narrative domain. As it does not depend on specific corpora it can thus cover the multiple application domains we wanted to test. This contrasts with more recent methods for hypernym or antonym generation based on Word2Vec (Sahin, 2017).

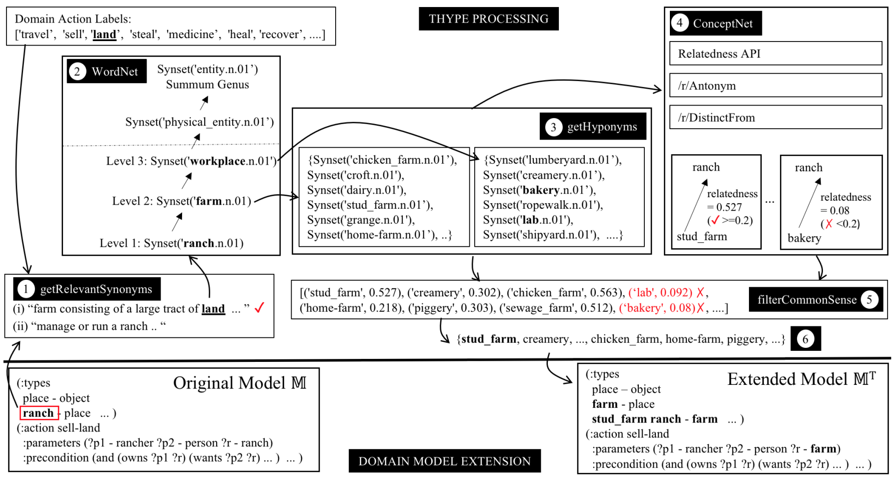
Since Wordnet can have multiple meanings for a word, we filter the set of synonyms of the term to reduce the number that are considered and the consequent number of alternate types returned by THYPE. To do this we look for definitions (in the text from WordNet) whose words intersect with action names in the domain model. This is achieved via the function GETRELEVANTSYNONYMS which selects the most likely synonyms (nouns) of $t$ on the basis of intersection with $A$, the names of the actions in the current version of the domain model. This filtering approach is based on semantic cohesion in a manner inspired by (Morris and Hirst, 1991).

When searching for hypernyms of a given term, it is possible to specify how many levels of the tree from the term to the *summum genus* (i.e. the most generic element of the hierarchy), we retain (`HYPERNYM_LEVEL` in line 4). This is based on the observation that THYPE fails to provide relevant suggestions when looking at hypernyms of already generic terms, something that we measure by the distance between the term and summum genus. In our experiments, we used 3 levels, as this worked best in practice in terms of quality and performance (in other experiments we observed that increases in level yielded no significant improvements but degraded performance). As a result, for each type, we retained 3 terms: the original term and 2 hypernyms.

Once the set of hypernyms is obtained, then a set of candidate alternative types is collected, *ho*, by considering each hypernym and extracting its hyponyms (line 6-8). Finally, the set of hyponyms is filtered using common sense reasoning (line 9). To do this, we use ConceptNet (Liu and Singh, 2004; Speer et al., 2017), a common sense knowledge base developed from natural language units.

Common sense reasoning is performed by the function FILTERCOMMONSENSE. The rationale for the filtering is to place some semantic restrictions on the alternative characters or narrative objects to be added to the domain model: thus we are interested in those terms which are related and not those which are antonyms. The function returns a set $ho'$, which is a filtered version of the input set of alternatives *ho*. The filtering process is based on whether alternatives from *ho* are related to type $t$, according to specific relations provided by ConceptNet (lines 13–18). First, we only consider alternatives that are related to $t$ (ISRELATED returns true if the `/relatedness` API[5] returns a value greater or equal than 0.2; this value was found to be adequate experi-

[5] The `/relatedness` API is like `/related`, but instead of ranking the top related terms to a given query, it returns the relatedness value for a particular pair of terms.


THYPE PROCESSING
Domain Action Labels:
['travel', 'sell', 'land', 'steal', 'medicine', 'heal', 'recover', ....]
2 WordNet
Synset('entity.n.01')
Summum Genus
Synset('physical_entity.n.01')
Level 3: Synset('workplace.n.01')
Level 2: Synset('farm.n.01)
Level 1: Synset('ranch.n.01)
3 getHyponyms
4 ConceptNet
Relatedness API
/r/Antonym
/r/DistinctFrom
ranch
relatedness = 0.527 (✓ >=0.2)
stud_farm
...
ranch
relatedness = 0.08 (✗ <0.2)
bakery
1 getRelevantSynonyms
(i) "farm consisting of a large tract of land ... " ✓
(ii) "manage or run a ranch .. "
filterCommonSense 5
6
Original Model $\mathbb{M}$
Extended Model $\mathbb{M}^T$
DOMAIN MODEL EXTENSION

```
{Synset('chicken_farm.n.01'),
Synset('croft.n.01'),
Synset('dairy.n.01'),
Synset('stud_farm.n.01'),
Synset('grange.n.01'),
Synset('home-farm.n.01'), ..}
```

```
{Synset('lumberyard.n.01'),
Synset('creamery.n.01'),
Synset('bakery.n.01'),
Synset('ropewalk.n.01'),
Synset('lab.n.01'),
Synset('shipyard.n.01'),  ....}
```

```
[('stud_farm', 0.527), ('creamery', 0.302), ('chicken_farm', 0.563), ('lab', 0.092) ✗,
('home-farm', 0.218), ('piggery', 0.303), ('sewage_farm', 0.512), ('bakery', 0.08)✗, ....]
```

```
{stud_farm, creamery, ..., chicken_farm, home-farm, piggery, ...}
```

```
(:types
  place - object
  ranch - place   ... )
(:action sell-land
  :parameters (?p1 - rancher ?p2 - person ?r - ranch)
  :precondition (and (owns ?p1 ?r) (wants ?p2 ?r) ... )  ... )
```

```
(:types
  place – object
  farm - place
  stud_farm ranch - farm   ... )
(:action sell-land
  :parameters (?p1 - rancher ?p2 - person ?r - farm)
  :precondition (and (owns ?p1 ?r) (wants ?p2 ?r) ... )  ... )
```

Fig. 6: Example of THYPE generation for *ranch* from original model: (1) use action labels to filter term meanings; (2) WordNet hierarchies accessed to level 3 to get Hypernyms; (3) get Hyponyms for all selected Hypernyms; (4) ConceptNet lookup is used as part of common sense filtering; (5) relatedness is used to reject Hyponyms under threshold value of 0.2; (6) example selection from THYPE alternatives to extend domain model (see text for further detail).

mentally). Of these, we exclude the terms that are antonyms or distinct (using the `/r/Antonym` and `/r/DistinctFrom` relations respectively).

We are interested in keeping the number of alternatives returned by THYPE to a reasonable number, to avoid spurious expansion that would not comply with the underlying actions and PDDL operators. In our experiments we took this to be 20 (variable `MAX_NUM`). If the number of hyponyms is reasonable, $ho' \leq$ `MAX_NUM` (line 19), no further common sense reasoning is performed and this set of alternatives, $ho'$, is returned. Should it happen that $ho' >$ `MAX_NUM` (line 19) further processing is performed to bring this within the reasonable bound. However we wanted to avoid removing too many suggestions at this stage and hence we introduced an additional variable, `MIN_NUM`, to control the lower bound of the final set of alternatives. Our implementation used 5 as the default value of `MIN_NUM`. Then we further filter the set $ho'$ so it excludes alternatives that are not considered a subtype of $t$ in ConceptNet (lines 19–22). For this, we use the relation `/r/isA`. If the resulting set of alternatives is too small, we return a subset of the original set $ho$ with the alternatives that are more related to $t$ (according to ConceptNet's `/relatedness` API, lines 23–25).

### 6.1 An Example of THYPE in Action

As illustration, consider the application of THYPE to *ranch* from our PDDL encoding of a Western domain (Ware, 2014), (Figure 6). Shown is a part of the original PDDL domain model, $\mathbb{M}$, from which types and action labels have been extracted: input to THYPE is a type from the domain model and the action labels. For *ranch*, GETRELEVANTSYNONYMS ①, accesses the WordNet hierarchies which returns the two meanings shown in the figure. Our cohesion-based filtering relates the occurrence of "land" in the definition to the labels of the action *sell-land* defined in the domain. Hence this meaning is selected and used to obtain the set of hypernyms *he*. ② shows the WordNet tree from the selected meaning to the summum genus. We use a HYPERNYM_LEVEL of 3, and hence the set of hypernyms includes "farm" and "workplace".

The next step is to get the associated hyponyms for each hypernym in *he* ③. For this example the number of hyponyms before any common sense reasoning is 45. This is because the set of hypernyms includes the quite general term "workplace", and thus some of the hyponyms include unrelated terms such as "bakery" and "lab". Thus common sense reasoning is applied to reduce the number of alternatives on the basis of relatedness (via lookup in ConceptNet ④ and common sense filtering ⑤). It can be seen that after common sense reasoning, these unrelated terms are eliminated and the number of alternative types is reduced to 12 in this case, which can then be shown to a human domain modeller to select which ones to use to extend the domain. It is important to note that the final set contains alternatives originated by all hypernyms. For example, the final set contains the alternatives "creamery" and "stud_farm", which are hyponyms of "workplace" and "farm" respectively.

Also illustrated in Figure 6 is selection of THYPE alternatives to extend the original PDDL domain model, $\mathbb{M}$: in this case *stud_farm* is shown selected to produce the THYPE extended domain model, $\mathbb{M}^T$ (see the next section for further detail on this process).

## 7 Framework for Domain Model Extension

In this section we overview a modular and extensible framework that allows the combination of multiple off-line generated extensions into an original domain model, $\mathbb{M}$. Consider a library of language extensions, $\mathbb{E}$, as a collection of functions, $e : \mathbb{M} \mapsto \mathbb{M}^e$, such that the model, $\mathbb{M}$, is extended through some process into a richer model, denoted $\mathbb{M}^e$.

In the next section we detail how an original domain model, $\mathbb{M}$, can be extended with the addition of extensions such as those generated by the ANTON and THYPE operations overviewed in sections 7.2 and 7.1 respectively. We consider both individual extensions and combinations of multiple extensions.

We denote the THYPE and ANTON extensions of $\mathbb{M}$ as $\mathbb{M}^T$ and $\mathbb{M}^A$ respectively and use $\mathbb{M}^{TA}$ for the use of these extension methods in combination. Note that because THYPE adds new types of objects and ANTON adds new

| Original Model $\mathbb{M}$ | Extended Model $\mathbb{M}^T$ | |
|---|---|---|
| (:types | (:types | ① |
| person - object | person - object | |
| king - person | *sovereign* - person | |
| ...) | king *emperor - sovereign ...*) | |
| (:action order | (:action order | ② |
| :parameters | :parameters | |
| (?k - king ...) | (*?k - sovereign ...*) | |

Fig. 7: Aladdin Example, from Riedl and Young (2010), showing addition of a new type of character, *emperor*, which has been generated by THYPE and selected by an author as an alternative to king: ① new type emperor is placed at same level as king and parent type sovereign is added; ② new parent type *sovereign* replaces original type king in all action parameter occurrences.

actions the extension mechanisms are commutative and thus the order of application is unimportant. As a consequence $\mathbb{M}^{AT}$ is equivalent to $\mathbb{M}^{TA}$.

## 7.1 $\mathbb{M}^T$ Domain Extension with THYPE

THYPE proposed domain extensions are added to the original domain model $\mathbb{M}$ to create $\mathbb{M}^T$ as follows:
– For type $t$ in $\mathbb{M}$, the selected alternate types $t'$ are placed at the same level of the type hierarchy in $\mathbb{M}^T$.
– A new parent type, the hypernym shared by $t$ and members of $t'$ (from WordNet), is introduced.
– The new parent type replaces the original type as appropriate in actions, thus enabling the use of the set of different types.

As illustration, consider the type king in the Aladdin domain with type hierarchy in $\mathbb{M}$ as shown in Figure 7. Suppose a domain modeller has selected *emperor* as an alternative type of character with which to extend the domain. This new entity is automatically added to the domain model with the hypernym of king as parent type (i.e., *sovereign*) with the new type placed as child type, inheriting properties from the parent. In addition, references to type king in $\mathbb{M}$ are amended in $\mathbb{M}^T$ to the new parent type, *sovereign*, with corresponding changes to parameters in all action occurrences (such as the action *order* in Figure 7).

## 7.2 $\mathbb{M}^A$ Domain Extension with ANTON

ANTON extends the set of actions in the domain model with contrary (i.e. opposite) actions. It proceeds via construction of sets of state transition rules representing the partial transitions defined by the domain model. From these rules, domain model extensions are constructed via analysis of contrary transitions. For all transitions in the original model, the expectation is that their opposite transitions have a natural interpretation in the domain and should also be in the model. Thus if any are found to be missing they are proposed as extensions to the model (subject to domain model author approval). The actions are constructed, as in (Porteous et al., 2015), as follows:

- **Name**: generated with antonyms from online resources.
- **Parameters**: the same as those in the action from $\mathbb{M}$ which is being extended and which this action represents the opposite.
- **Pre-conditions**: are formed from enabling pre-conditions of original action in $\mathbb{M}$ i.e. predicates required to apply the action and unchanged by it; and post-conditions of the original action in $\mathbb{M}$ become pre-conditions.
- **Post-conditions**: are formed from the post-conditions of the original action in $\mathbb{M}$. The positive effects of the new action are any predicates deleted by the original, and the negative effects are those that were gained by the original action.

Our expectation is that robustness will be further increased when multiple model extensions are combined. To evaluate this we use THYPE in combination with ANTON.

## 8 Evaluation

In this paper our focus has been the ability of THYPE and ANTON domain model extensions to increase *robustness* and *diversity*. In sections 8.1 to 8.3 below, we present details of evaluation of these aspects.

In earlier work we evaluated the ability of THYPE and ANTON to generate alternatives in the context of a given narrative domain: both in terms of the number of alternatives proposed along with their plausibility and readability. We include some discussion of these results in section 8.4 but do not report them in full. For further detail see Porteous et al. (2015) and (2020).

### 8.1 Domain Models for Evaluation

For the evaluation we created PDDL 2.1 representations of a range of narrative planning domains. These were selected because they had appeared in the literature, provided a range of narrative contexts and had been modelled independently by different narrative authors. These domains are: Aladdin (Riedl and Young, 2010), Crime Drama (Kartal et al., 2014), Medical Drama (Porteous et al., 2015), Red Riding Hood (Riedl, 2009), Western (Ware, 2014)[6].

For each of the domains used in the evaluation, extended versions were created from the original domain model $\mathbb{M}$.

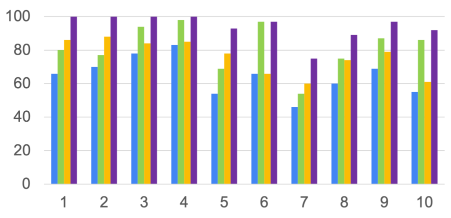
[6] Domains available to download from https://porteousjulie.bitbucket.io/

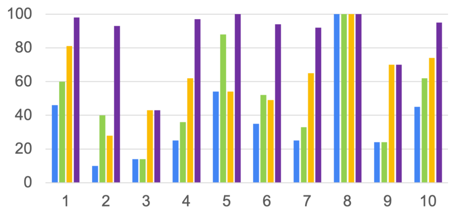
| Domain | Original Model $\mathbb{M}$ |  | ANTON Model $\mathbb{M}^A$ | THYPE Model $\mathbb{M}^T$ |
|---|---|---|---|---|
|  | #A | #T | #A | #T |
| Aladdin | 12 | 12 | 21 | 18 |
| Crime | 9 | 8 | 11 | 12 |
| Medical | 10 | 18 | 19 | 20 |
| Red | 5 | 10 | 9 | 14 |
| Western | 19 | 10 | 25 | 14 |

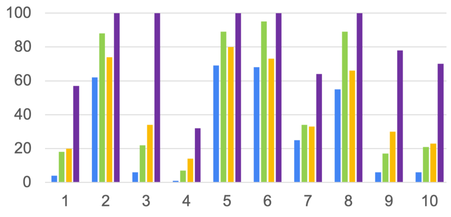
Fig. 8: For each domain used in the evaluation, figure shows: number of actions #A and types of object #T for the Original Domain Model $\mathbb{M}$; the number of actions in the ANTON extended domain model, $\mathbb{M}^A$; and the number of types in the THYPE extended domain model, $\mathbb{M}^T$.

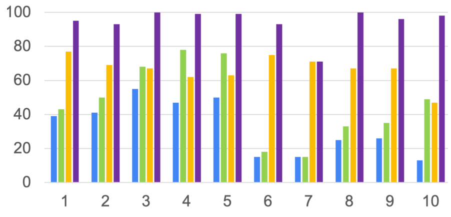
- $\mathbb{M}^T$: THYPE extended Domain Model
  This model was generated by selecting additional types from those proposed by the system and adding them to the model. The method for selecting extensions was to use those which we assigned a plausibility ranking of **"good"**, using the same plausibility ranking adopted in our earlier work (Porteous et al., 2020), as a proxy for author in the loop content selection (reflecting the genre dependent nature of these additions and need for choice from proposed alternatives).
  In addition, instances of each new type were added to narrative planning problem instances: a single instance of each of the new types was added to the problem instances in the test set. Using a naming convention for these object instances which adds a digit to the type name. For example, for the Aladdin domain and the new type *fairy* an object instance would be named *fairy1* (as shown in Figure 1).
- $\mathbb{M}^A$: ANTON extended domain model
  This model was generated for individual comparison with $\mathbb{M}^T$. We used our implementation of ANTON (Porteous et al., 2015) to generate additional actions for each of the domains. These actions were then added to $\mathbb{M}$ to create $\mathbb{M}^A$.
- $\mathbb{M}^{TA}$: THYPE and ANTON extended domain model
  This model was generated by adding all of the THYPE and ANTON extensions to the original domain model (discussed in the earlier sections) $\mathbb{M}$. Note that because THYPE adds new types of objects and ANTON adds new actions the extension mechanisms are commutative and thus the order of application is unimportant. As a consequence $\mathbb{M}^{AT}$ is equivalent to $\mathbb{M}^{TA}$.

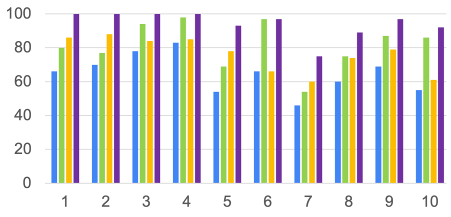
For each of the domains used in the evaluation, the size of the original domain model and these extended models, is shown in Figure 8. Listed are the number of actions #A and types of objects #T for $\mathbb{M}^A$ and $\mathbb{M}^T$.


(a) ALADDIN: 🟩$\mathbb{M}^T$, 🟨$\mathbb{M}^A$ outperform 🟦$\mathbb{M}$ 100%; 🟩$\mathbb{M}^T$ outperform 🟨$\mathbb{M}^A$ 60%


(b) CRIME: 🟩$\mathbb{M}^T$, 🟨$\mathbb{M}^A$ outperform 🟦$\mathbb{M}$ 70%; 🟨$\mathbb{M}^A$ outperform 🟩$\mathbb{M}^T$ 60%


(c) MEDICAL: 🟩$\mathbb{M}^T$, 🟨$\mathbb{M}^A$ outperform 🟦$\mathbb{M}$ 100%; 🟩$\mathbb{M}^T$ and 🟨$\mathbb{M}^A$ perform equally


(d) RED RIDING: 🟩$\mathbb{M}^T$, 🟨$\mathbb{M}^A$ outperform 🟦$\mathbb{M}$ 90%; 🟨$\mathbb{M}^A$ outperforms 🟩$\mathbb{M}^T$ 60%


(e) WESTERN: 🟩$\mathbb{M}^T$, 🟨$\mathbb{M}^A$ outperform 🟦$\mathbb{M}$ 100%; 🟩$\mathbb{M}^T$ outperforms 🟨$\mathbb{M}^A$ 60%

KEY
🟦 $\mathbb{M}$ original Domain Model
🟩 $\mathbb{M}^T$ THYPE Extended Model
🟨 $\mathbb{M}^A$ ANTON Extended Model
🟪 $\mathbb{M}^{TA}$ ANTON & THYPE Extended Model

Fig. 9: Robustness Results: comparing ability of 🟩 $\mathbb{M}^T$, 🟨 $\mathbb{M}^A$, and 🟪 $\mathbb{M}^{TA}$ to continue to original goal in execution failure simulation (5 domains, 10 problems, 100 fail-restarts on each); 🟦 $\mathbb{M}$ included for comparison. (1) Performance of $\mathbb{M}^T$ and $\mathbb{M}^A$ are consistently good across all domains and outperform $\mathbb{M}$ throughout. (2) $\mathbb{M}^{TA}$ outperforms all other models and demonstrates the performance gains that can result from combining extensions (see text for detail).

### 8.2 Robustness of Extended Domain Models

To explore the ability of extended models to support plan generation through to the original goal in a dynamically changing environment we conducted a series of simulations with $\mathbb{M}$, $\mathbb{M}^T$, $\mathbb{M}^A$ and $\mathbb{M}^{TA}$. The intention was to simulate the execution of an Interactive Narrative System, where user interactions can change the state of the narrative world leading to execution failure. These simulated interactions, their frequency and impact on the narrative world, in terms of precondition failure, were selected at random. The rationale for random selection of interaction points and frequency was based on the possibility, in general, that users can interact freely at any stage of a narrative and as frequently as they choose. The rationale for random precondition failure was to demonstrate the increased robustness of the domain models in general: without the need to approximate most likely human-behaviour.

For each run the simulation firstly generated a narrative plan for the input planning problem (using METRIC-FF Hoffmann (2003)) and stepped through each action in the plan. At random points of the simulated execution a random selection of the current actions preconditions were updated in the current state of the world so that it could not be applied. At this point of failure, the simulation tried to regenerate the remainder of the plan from the current state still using the original goal. Whenever the system was unable to generate a plan the system reported failure and restarted, otherwise it continued to the original goal, reported success and restarted. The results of simulation runs, for 10 problem instances for each of the 5 domains, with 100 simulated failures and restarts on each problem instance, are shown in Figure 9.

Our expectation was that addition of extensions would increase the robustness of the domain models. In particular we hypothesized that: (i) the THYPE and ANTON extended models, $\mathbb{M}^T$ and $\mathbb{M}^A$, would both be more robust than $\mathbb{M}$; and (ii) the domain model featuring combined extensions, $\mathbb{M}^{AT}$, would outperform all other models, with respect to robustness.

To evaluate the robustness of THYPE extended domain models consider the results for $\mathbb{M}^T$, the THYPE extended domain model (see Figure 9). Our expectation was that this would improve simulation performance over the original domain $\mathbb{M}$ which is very clearly the case: 100% of the runs for Aladdin, Medical Drama and Western; 90% for Red Riding Hood; and 70% for Crime Drama. The instances where THYPE makes no impact occur when a virtual character is specified in the goal itself and where state changes negate this e.g. in Crime Drama and Red Riding Hood when characters are left $\neg alive$ but can't be replaced by an alternate type of character.

The results for the ANTON extended domain model, $\mathbb{M}^A$, are also shown in Figure 9. Here, our expectation was that this would also improve simulation performance over the original domain $\mathbb{M}$. The results for $\mathbb{M}^T$ show that it performs consistently well, yielding similar performance improvements to $\mathbb{M}^A$. There are differences between instances where ANTON extensions have impact, in comparison to THYPE. For example, THYPE makes no impact when a virtual character features in the goal and cannot be replaced by an alternative type of character should state changes leave them $\neg alive$. In contrast, ANTON extensions would include the reverse action to bring a character back to life (see examples in section 8.4). On the basis of this observation we anticipate that combination of domain model extensions would yield additional performance gains via complementary impact from both sets of extensions.

To evaluate the robustness of a combination of domain model extensions we should consider the results for $\mathbb{M}^{TA}$. On all problem instances across the domains, where the individual extensions improved performance, this model outperforms all others (i.e. exceptions are the aforementioned Crime and Red Riding Hood examples and single instances of Aladdin and Western where AN-

| **Domain** | Aladdin | Crime Drama | Medical | Red Riding Hood | Western |
|---|---|---|---|---|---|
| **Diversity** $\mathbb{M}^A$ | 80% | 70% | 80% | 40 % | 60 % |

Fig. 10: Narrative Diversity Results: percentage of instances where diverse narratives could be generated using $\mathbb{M}^A$ (these are instances where narratives could also be generated using the original domain model $\mathbb{M}$).

TON has no impact). This is to be expected as it leverages the robustness gains from both sets of extensions. For example, consider Aladdin, where narrative continuation following execution failure can require both ANTON extensions such as *divorce* (to render a character single and able to marry) and THYPE generated alternative types of characters such as suitors. We conclude that the results support our expectations of the robustness of extended domains: for $\mathbb{M}^T$, $\mathbb{M}^A$and $\mathbb{M}^{TA}$.

## 8.3 Diversity of Extended Domain Models

Domain models for use in Interactive Narrative (IN) systems need to allow for the generation of diverse sets of narratives to allow for system replayability. The domain model extension operations overviewed in this paper achieve this through the addition of new narrative actions, ANTON, and new types of virtual characters and narrative objects, THYPE, which open up a wider range of narrative possibilities. As an example, consider the very different narratives for the Aladdin domain discussed in section 2 and illustrated in Figure 1.

It is self evident that the extension of domain models to include additional narrative actions and objects will increase the diversity of narratives that can be generated. Nevertheless to demonstrate the potential of domain model extension to support diversity we conducted a series of experiments with the set of narrative domains described earlier, using the original domain model, $\mathbb{M}$, and the ANTON extended model, $\mathbb{M}^A$. We hypothesized that, as contrary transitions generated by ANTON introduce the possibility of decisions being changed (e.g. in Aladdin, putting the genie back in the lamp and enabling another agent to take control), the diversity of narrative generation would be much higher for $\mathbb{M}^A$ than for $\mathbb{M}$. We note that the use of distance measures in this way as part of narrative evaluation is consistent with a growing trend in narrative research (e.g. Jones and Isbell (2014)).

For the experiments we used the domains introduced earlier, (referred to as Aladdin, Crime, Medical, Red Riding Hood and Western), and the same set of random narrative planning instances. These instances were ones for which a narrative could be generated using the original domain model $\mathbb{M}$ and METRIC-FF[7]. Then for each such instance a narrative was generated using the original domain model $\mathbb{M}$ and a set of diverse narratives were generated with the AN-

[7] METRIC-FF: http://fai.cs.uni-saarland.de/hoffmann/metric-ff.html

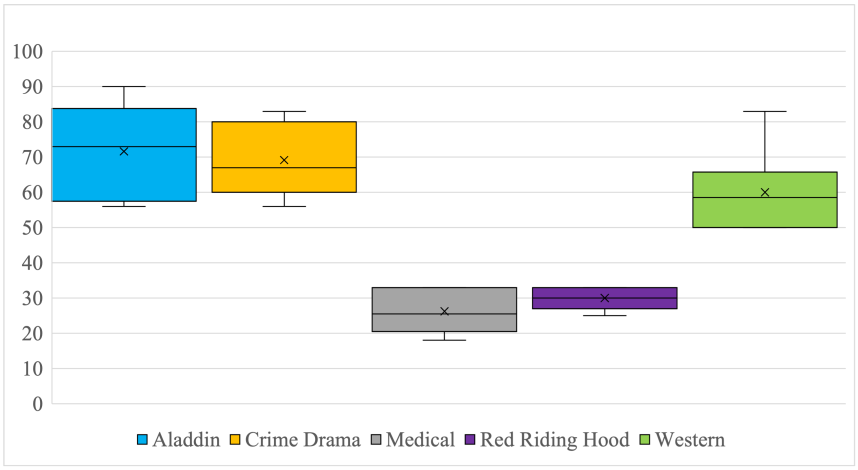


Fig. 11: Narrative Diversity Results: ability of $\mathbb{M}^A$ to generate diverse narratives (5 domains, 10 problems per domain). Plot shows the distance between narratives generated by the original model, $\mathbb{M}$, and the most diverse $\mathbb{M}^A$ narrative (expressed as % of the maximum possible, with distance between narratives measured using Levenshtein distance[8]). Results support our expectation of increased narrative diversity resulting from domain extension via introduction of the ANTON generated contrary transitions. For further detail see text.

TON extended domain $\mathbb{M}^A$, adopting the approach of Coman and Muñoz-Avila (2011). Then the original narrative was compared with the diverse set of narratives to find the ANTON narrative that was most different to the original. For comparison the difference between narratives was measured using Levenshtein Distance (Levenshtein, 1966; Jones and Pevzner, 2004) which counts the edit distance[8] between two strings. To obtain suitable strings for comparison, action names in the narratives were mapped to unique characters.

The results of these experiments are shown in Figure 10 and 11. Figure 10 shows the number of narrative instances for which the ANTON extended domain model, $\mathbb{M}^A$, was able to generate diverse narratives. We observe that the diversity is increased consistently across the domains although there are instances in which no diverse solutions can be generated. This is to be expected as it depends on the nature of the planning instance: for example, for Aladdin instances where an alternative narrative is possible using an action to put the genie back in the lamp, making it possible for another agent to take control, then this is possible using $\mathbb{M}^A$.

[8] Levenshtein distance (Levenshtein, 1966) is a count of the minimum edit operations needed to transform one string to be identical to another. This was selected since it can be used to measure the distance between different length strings (important for plan comparison). The maximum possible distance between 2 strings is the shortest string length.

| Original Content | THYPE content | Definition | Rank |
|---|---|---|---|
| Aladdin | | | |
| dragon | troll | a supernatural creature supposed to live in caves or mountains | G |
| dragon | werewolf | a monster able to change appearance from human to wolf and back | U |
| Crime Drama | | | |
| car | jeep | | G |
| car | horseless-carriage | | P |

Fig. 12: Examples of THYPE generated content and user rankings (G(ood), P(oor) and U(nsure)). For the type "dragon" (Aladdin), the alternative type, "troll" was ranked G, whereas the alternative, "werewolf", was ranked U; for "car" (Crime Drama) the type "jeep" was ranked G, and "horseless-carriage" P. These rankings are genre dependent and can differ across domains e.g. a werewolf is plausible in Harry Potter but not Aladdin. Further detail see text.

The results of the narrative comparisons are plotted in Figure 11: with difference shown as a percent of the maximum possible between the original domain narrative and the most different ANTON domain narrative. The results clearly demonstrate that the extended domain increases the diversity of narratives that can be generated. This is an important result since the increased generativity resulting from use of the ANTON extended domain did not require the considerable effort of manual domain creation.

## 8.4 Plausibility and Readability of Domain Model Extensions

In earlier work we evaluated the ability of the extension operations, THYPE and ANTON, to generate alternatives in the context of a given narrative domain in terms of their plausibility and readability (see: Porteous et al. (2015) and (2020)). The conclusion of both evaluations was that the generated domain content made sense to users, was plausible, and that the semantic labels were appropriate. We do not report those results here. Rather, we include some examples to illustrate the types of semantic labels generated by THYPE and ANTON:

– THYPE: Figure 12 shows some examples generated by THYPE for the Aladdin and Crime Drama domains with user rankings of G(ood), P(oor) and U(nsure). For the Aladdin type of object, "dragon" the THYPE suggested alternatives include "troll" and "werewolf". From their definitions it can be seen that both share similar characteristics to the original domain type, such as being magical beings, but are different types of magical beings. The interesting feature here is whilst they both are plausible, this is very much genre and story world dependent. For example, a human author might decide that a werewolf makes sense in the world of Harry Potter (Rowling,

| ① ACTIONS | | | ② PROPERTIES | | |
|---|---|---|---|---|---|
| Original Domain | ANTON label | Rank | Original Domain | ANTON label | Rank |
| marry | divorce | G | alive | dead | G |
| slay | restore | U | scary | bold | P |

Fig. 13: Examples of ANTON generated labels (Aladdin Domain) and user rankings (G(ood), P(oor) and U(nsure)). ① and ② show ACTION and PROPERTY labels from the original domain and ANTON generated labels. Users ranked the appropriateness of the generated semantic labels e.g. for the action named "marry" the opposite action label "divorce" was ranked G; the opposite to action "slay", labelled "restore", was ranked U; but label "bold", opposite to "scary", was ranked P. See text for further detail and discussion.

1999) but not in the context of Aladdin. For the Crime Drama domain, the alternatives to "car" include "jeep" and "horseless-carriage". Given the setting of a current day crime drama, a horseless carriage could be ranked as a poor alternative type of object to expand the story world. However, in some story worlds this is very plausible: consider, for example, the time travelling story world of Back to the Future (Zemekis, 1985).

– ANTON Figure 13 shows some examples generated by ANTON for action labels ① and property labels ② from the original narrative domain model, with user rankings of G(ood), P(oor) and U(nsure). Recall that the ANTON semantic labels are for content representing opposite actions and properties. For the action "marry" the label for its opposite, namely "divorce", is clearly a good suggestion (ranked G). In contrast the label for the opposite to "slay" is not so obvious, which is reflected in the U ranking. This is to be expected for those actions, such as slay, where no obvious opposite label exists. It may also be that the somewhat archaic vocabulary contributes in this case. Note that the rankings given here apply to the appropriateness of the semantic labels themselves and not the action transition. For example, whether or not a character that has been slayed can be brought back to life is something that is genre dependent: it makes sense in Aladdin but not in other contexts.
  For the property labels, alive and scary, the examples illustrate similar issues. For "alive" there is an obvious label for the opposite property, namely "dead", as reflected with the G ranking. This is not the same for the property "scary", and it is not surprising that the ANTON suggested label for the opposing property is poor.

## 9 Conclusion

Narrative Planning differs from traditional planning problems in a number of ways. In view of the main applications of narrative planning, which are interactive narrative and narrative generation, plan diversity outweighs plan optimality in terms of length or use of resources. While narrative planning and traditional planning both share a requirement for robustness, the causes for failure tend to be more varied in narrative planning, hence more difficult to capture. Nowhere is this specificity more acute than when defining planning domains: narrative generation assumes a diversity of operators which remains consistent within a family of planning domains and continuing or repairing a narrative plan in front of failure again requires various yet consistent operators.

In this work, we have addressed this problem by proposing to extend narrative planning domains semi-automatically from a default baseline domain, using principled mechanisms of two types. The first one captures the logical consequences of operator failure to define contrary operators able to resume the course of planning by undoing the actions' consequences, rather than simply reversing them. The second one uses an ontological approach to diversify potential actions through the exploration of a conceptual hierarchy. In both cases, an additional requirement for the generated operators is to be human-readable: this can be achieved by associated appropriate linguistic resources to the above operations, so that both the operators' action names and individual predicates convey the essence of the new operator to a human operator having produced the default planning domain.

We have presented experiments applying these methods to various narrative planning domains available from the literature. Our results show how domains extended using ANTON and THYPE are more robust and diverse: with robustness measured in terms of the likelihood of continuation, in the event of plan execution failure, to the original goal across a range of domains; and diversity measured by the ability to generate different narratives. The results also show that combining different extension mechanisms, such as ANTON and THYPE, results in greater increases in domain robustness and diversity.

One direct application of our method would be to facilitate the authoring of more complex narrative planning domains, which has proven a bottleneck in the scalability of narrative generation systems. Despite the very encouraging results obtained by our approach through relatively simple methods, it still faces a number of limitations, and could take advantage from recent advances in language models. For instance, a better integration could be achieved between the two methods, through the use of more specific common sense models and a finer distinction between these models and linguistic resources. Considering the generation of human-readable labels, it could benefit from new resources that allow the addition of more specific embeddings, the latter being trained on text corpora pertaining to the target narrative genre for the planning domain.

**Acknowledgements** The work reported in this article was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (João F. Ferreira); and by funds from DSI Collaborative Grant CR-0016 (Julie Porteous).

## References

Amir E, Chang A (2008) Learning Partially Observable Deterministic Action Models. Journal of Artificial Intelligence Research 33(1):349–402

Bryce D, Benton J, Boldt MW (2016) Maintaining Evolving Domain Models. In: Proceedings of the 25th International Joint Conference on AI (IJCAI)

Cashmore M, Fox M, Long D, Magazzeni D, Ridder B, De Carolis V, Lane D, Maurelli F (2015) Dynamically Extending Planning Models using an Ontology. In: Proceedings of the 2nd ICAPS Workshop on Planning and Robotics (PlanRob-15)

Coman A, Muñoz-Avila H (2011) Generating Diverse Plans Using Quantitative and Qualitative Plan Distance Metrics. In: Proceedings of 25th AAAI Conference on Artificial Intelligence (AAAI)

Cresswell S, Gregory P (2011) Generalised Domain Model Acquisition from Action Traces. In: Proc. of the 21st Int. Conference on Automated Planning and Scheduling (ICAPS)

Cresswell S, Fox M, Long D (2002) Extending TIM domain analysis to handle ADL constructs. In: Knowledge Engineering Tools and Techniques for AI Planning Workshop of the 6th International Conference on Artificial Intelligence Planning Systems (AIPS)

Cresswell S, Mccluskey T, West M (2009) Acquisition of Object-Centred Domain Models from Planning Examples. In: Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)

Dolejsi J, Long D, Fox M, Besancon G (2018) PDDL Authoring and Validation Environment for Building end-to-end Planning Solutions. In: Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS)

Fellbaum C (1998) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press, URL `https://wordnet.princeton.edu/`

Fox M, Long D (1998) The Automatic Inference of State Invariants in TIM. Journal of Artificial Intelligence Research 9:367–421

Gazen BC, Knoblock C (1997) Combining the Expressiveness of UCPOP with the Efficiency of Graphplan. In: Proceedings of European Conference on Planning (ECP)

Göbelbecker M, Keller T, Eyerich P, Brenner M, Nebel B (2010) Coming Up With Good Excuses: What to do When no Plan Can be Found. In: Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)

Harrell DF, Gonzalez C, Blumenthal H, Chenzira A, Powell N, Piazza N, Best M (2010) A Cultural Computing Approach to Interactive Narrative: The Case of the Living Liberia Fabric. In: Proceedings of AAAI Fall Symposium on Computational Models of Narrative

Helmert M (2006) The Fast Downward planning system. Journal of AI Research 26(1):191–246

Herzig A, Menezes V, Nunes de Barros L, Wassermann R (2014) On the revision of planning tasks. In: Proceedings of the 21st European Conference on

AI (ECAI)

Hoffmann J (2003) The Metric-FF Planning System: Translating ”Ignoring Delete Lists” to Numeric State Variables. Journal of Artificial Intelligence Research 20:291–341

Janghorbani S, Modi A, Buhmann J, Kapadia M (2019) Domain Authoring Assistant for Intelligent Virtual Agent. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)

Jones JK, Isbell CL (2014) Story Similarity Measures for Drama Management with TTD-MDPs. In: Proceedings of 13th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)

Jones N, Pevzner P (2004) An Introduction to Bioinformatics Algorithms (Computational Molecular Biology). The MIT Press

Kambhampati S (2007) Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain models. In: Proceedings of the 22nd National Conference on AI (AAAI)

Kartal B, Koenig J, Guy SJ (2014) User-driven narrative variation in large story domains using monte carlo tree search. In: Proceedings of the 13th International Conference on Autonomous Agents and MultiAgent Systems, (AAMAS)

Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions and reversals. Cybernetics and Control Theory 10:707–710

Li B, Zook A, Davis N, Riedl M (2012) Goal-driven conceptual blending: A computational approach for creativity. In: Proceedings of 3rd International Conference on Computational Creativity (ICCC)

Li B, Lee-Urban S, Johnston G, Riedl M (2013) Story Generation with Crowdsourced Plot Graphs. In: Proceedings of 27th AAAI Conference on Artificial Intelligence (AAAI)

Lin D, Zhao S, Qin L, Zhou M (2003) Identifying synonyms among distributionally similar words. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)

Lindsay A, Read J, Ferreira JF, Hayton T, Porteous J, Gregory PJ (2017) Framer: Planning Models from Natural Language Action Descriptions. In: Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS)

Liu H, Singh P (2004) ConceptNet—a practical commonsense reasoning toolkit. BT technology journal 22(4):211–226

McDermott D, Ghallab M, Howe A, Knoblock C, Ram A, Veloso M, Weld D, Wilkins D (1998) PDDL-the planning domain definition language. Tech. rep., Yale University

McIntyre N, Lapata M (2009) Learning to Tell Tales: A Data-driven Approach to Story Generation. In: Proceedings of 47th Meeting of the Association for Computational Linguistics (ACL)

Morris J, Hirst G (1991) Lexical cohesion computed by thesaural relations as an indicator of the structure of text. Computational linguistics 17(1):21–48

Nazar R, Janssen M (2010) Combining resources: Taxonomy extraction from multiple dictionaries. In: Proceedings of 7th International Conference on Language Resources and Evaluation (LREC)

Orkin J, Roy DK (2012) Understanding Speech in Interactive Narratives with Crowdsourced Data. In: Proceedings of 8th Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)

Pauls A, Klein D (2011) Faster and smaller n-gram language models. In: Proceedings of 49th Meeting of the Association for Computational Linguistics (ACL)

Pérez y Pérez R, Morales N, Rodríguez L (2012) Illustrating a Computer Generated Narrative. In: Proceedings of 3rd International Conference on Computational Creativity (ICCC)

Pizzi D, Cavazza M (2008) From debugging to authoring: Adapting productivity tools to narrative content description. In: Proceedings of the 1st International Conference on Interactive Digital Storytelling (ICIDS)

Porteous J, Cavazza M, Charles F (2010) Applying planning to interactive storytelling: Narrative control using state constraints. ACM Transactions on Intelligent Systems and Technology (TIST) 1(2):10

Porteous J, Teutenberg J, Pizzi D, Cavazza M (2011) Visual Programming of Plan Dynamics using Constraints and Landmarks. In: Proceedings of the 21st Int. Conference on Automated Planning and Scheduling

Porteous J, Lindsay A, Read J, Truran M, Cavazza M (2015) Automated Extension of Narrative Planning Domains with Antonymic Operators. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)

Porteous J, Ferreira JF, Lindsay A, Cavazza M (2020) Extending Narrative Planning Domains with Linguistic Resources. In: Proceedings of 19th Conference on Autonomous Agents and MultiAgent Systems (AAMAS)

Riedl MO (2009) Incorporating Authorial Intent into Generative Narrative Systems. In: Proc. of AAAI Spring Symp. Intelligent Narrative Technologies

Riedl MO, Young RM (2004) An intent-driven planner for multi-agent story generation. In: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)

Riedl MO, Young RM (2006) From Linear Story Generation to Branching Story Graphs. IEEE Computer Graphics and Applications 26(3):23–31

Riedl MO, Young RM (2010) Narrative planning: Balancing plot and character. Journal of Artificial Intelligence Research 39:217–268

Robertson J, Young RM (2019) Perceptual experience management. IEEE Transactions on Games 11(1):15–24

Rowling JK (1999) Harry Potter and the Prisoner of Azkaban. Bloomsbury Publishing

Sag I, Baldwin T, Bond F, Copestake A, Flickinger D (2002) Multiword Expressions: A Pain in the Neck for NLP. In: Proceedings of 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)

Sahin G (2017) Extraction of Hyponymy, Meronymy and Antonymy Relation Pairs : A Brief Survey. International Journal on Natural Language Computing (IJNLC) 6(2)

Sina S, Rosenfeld A, Kraus S (2014) Generating content for scenario-based seriousgames using CrowdSourcing. In: Proceedings of 28th AAAI Conference on Artificial Intelligence (AAAI)

Speer R, Chin J, Havasi C (2017) Conceptnet 5.5: An open multilingual graph of general knowledge. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)

Swanson R, Gordon AS (2012) Say Anything: Using Textual Case-Based Reasoning to Enable Open-Domain Interactive Storytelling. ACM Trans Interact Intell Syst 2(3)

Teutenberg J, Porteous J (2013) Efficient Intent-based Narrative Generation Using Multiple Planning Agents. In: Proceedings of 12th Conference on Autonomous Agents and MultiAgent Systems (AAMAS)

Turney P (2008) A uniform approach to analogies, synonyms, antonyms, and associations. In: Proceedings of the 22nd International Conference on Computational Linguistics

Vaquero TS, Silva JR, Beck JC (2013) Post-design Analysis for Building and Refining AI Planning Systems. Eng Appl Artif Intell 26(8):1967–1979

Walsh TJ, Littman ML (2008) Efficient Learning of Action Schemas and Web-Service Descriptions. In: Proceedings of 23rd AAAI Conference on AI (AAAI)

Wang P, Rowe J, Min W, Mott B, Lester J (2018) High-Fidelity Simulated Players for Interactive Narrative Planning. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)

Ware SG (2014) A plan-based model of conflict for narrative reasoning and generation. PhD thesis, North Carolina State University, Domains available from: https://www.cs.uky.edu/˜sgware/projects/glaive/

Wu K, Yang Q, Jiang Y (2007) ARMS: An automatic knowledge engineering tool for learning action models for AI planning. The Knowledge Engineering Review 22(2):135–152

Yao L, Peng N, Weischedel RM, Knight K, Zhao D, Yan R (2019) Plan-And-Write: Towards Better Automatic Storytelling. In: Proceedings of the 33rd National Conference on AI (AAAI)

Yoon S, Kambhampati S (2007) Towards model-lite planning: A for learning & planning with incomplete domain models. In: ICAPS Workshop on Artificial Intelligence Planning and Learning

Young J, Basile V, Kunze L, Cabrio E, Hawes N (2016) Towards lifelong object learning by integrating situated robot perception and semantic web mining. In: Proceedings of the European Conference on Artificial Intelligence (ECAI)

Young J, Kunze L, Basile V, Cabrio E, Hawes N, Caputo B (2017) Semantic web-mining and deep vision for lifelong object discovery. In: IEEE International Conference on Robotics and Automation (ICRA)

Young RM (1999) Notes on the Use of Plan Structures in the Creation of Interactive Plot. In: AAAI Fall Symposium on Narrative Intelligence

Zemekis R (1985) Back To The Future
Zhuo HH, Kambhampati S (2013) Action-model Acquisition from Noisy Plan Traces. In: Proceedings of 23rd International Joint Conference on Artificial Intelligence (IJCAI)
Zhuo HH, Yang Q, Hu DH, Li L (2010) Learning complex action models with quantifiers and logical implications. Artificial Intelligence 174(18):1540–1569