Extending Narrative Planning Domains with Linguistic Resources

Julie Porteous
School of Science, RMIT University
Melbourne, Australia. julie.porteous@rmit.edu.au

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

Alan Lindsay
University of Huddersfield
Huddersfield, UK. alan.lindsay@hw.ac.uk

Marc Cavazza
University of Greenwich
London, UK. m.cavazza@greenwich.ac.uk

ABSTRACT

Interactive Narrative (IN) is an emerging application of planning for control of virtual character behaviours. Despite popularity in research, more widespread adoption of planning has been hindered by the difficulty of authoring planning domain models as INs require models which are robust to dynamic environments and capable of generating a diversity of solutions. To reduce this authoring burden we explored automated extension of partially developed planning domains to address *robustness* and *diversity*. We introduce a novel algorithm, THYPE, which proposes additional types of characters and objects to extend a domain model. The extensions increase robustness by enlarging the range of ways to enact planned behaviours. In the paper we embed THYPE within a modular and extensible framework that allows multiple off-line generated extensions to be combined. We present results of a user study that show THYPE suggestions are plausible. We also empirically demonstrate the ability of THYPE extensions to increase the robustness of the models, demonstrate the modular nature of our extension framework by combining THYPE with another extension approach dedicated to recovery from plan failure and present results which show that enhanced performance results from combining multiple extensions.

KEYWORDS

Socially interactive agents; Virtual Humans; Interactive Narrative

ACM Reference Format:
Julie Porteous, João F. Ferreira, Alan Lindsay, and Marc Cavazza. 2020. Extending Narrative Planning Domains with Linguistic Resources. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

One approach to the control of Virtual Characters in Interactive Narrative (IN) applications has been to use 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 [18, 21, 27, 30]. 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 [34].

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

However, developing planning domain models is challenging and this is compounded for IN applications due to the issues of *robustness* and *diversity*. Domain models for use in IN systems need to be *robust*: cover world state variations resulting from dynamic changes to the narrative world (e.g. user interactions) and enable appropriate recovery via replanning. Domain models also need to be *diverse*: to allow for generation of diverse sets of plans to ensure replayability. One further complication is the need for non-technical authors to create the content to populate the models.

Whilst a number of tools have been developed to assist domain modelling [5, 25], maintain models [1] and automate the process [3, 14, 29], the potential to assist the modelling process through automated extension of partially developed models remains largely unexplored. Yet 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 that support the generation of variants faces the difficulty of encoding what these variants could be. In other words, the paradox of interactive narrative also applies to its authoring component: a situation which has hitherto not received sufficient attention.

Our starting point was reasoning about potential sources of execution failure as a drive for narrative evolution, as well as alternative use of narrative resources (including virtual characters and narrative objects as these are a translation, in Planning terms, of the authoring intention for narrative generation). As illustration, consider the crime drama example in Figure 1 with a plan which has characters committing crimes, with others performing arrests and so on. Also shown are possible user interactions: one where the user renders the detective unable to arrest the suspect (e.g. shoot the detective); and another where they render the car unroadworthy (e.g. shoot the car tyres). A more robust domain model would cover these states and enable recovery planning. This could be achieved by extending the domain model with additional types of virtual characters and narrative objects able to arrest the criminal (e.g. the addition of a different type to detective) and different types of transport (e.g. introduce a new type of motor vehicle). Such an extended domain is also more diverse: capable of generating a wider range of narratives.

We have developed a novel algorithm, called THYPE, that proposes precisely this form of domain model extensions: additional types of virtual characters and narrative objects. These are identified using semantic relationship information sourced from online linguistic resources, taking advantage of the linguistic properties inherent in PDDL actions and domain predicates. Given an input

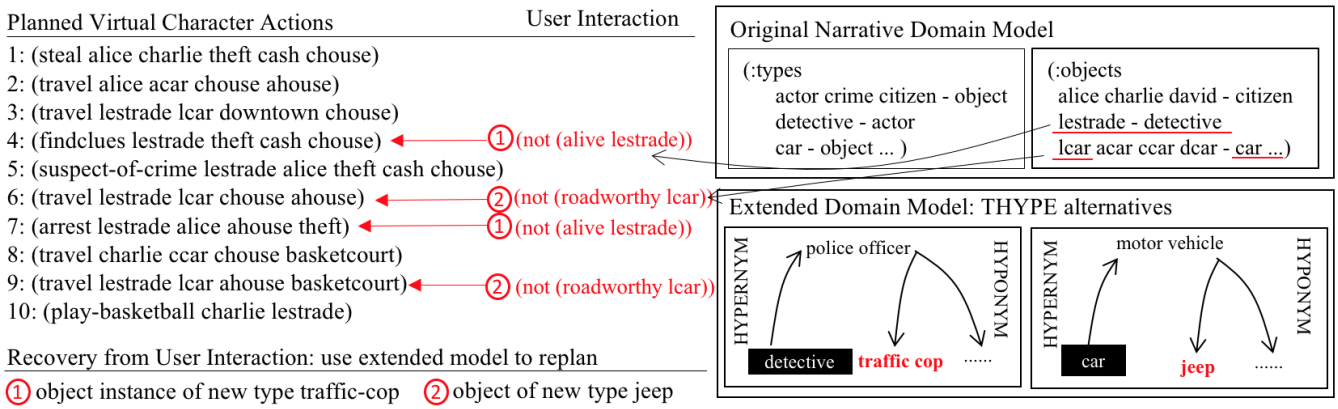

Figure 1: Crime Drama Example (from [13]) showing plan for Virtual Characters and possible User Interaction causing execution failure: ① the detective (lestrade) not alive; ② the car (lcar) not roadworthy. Possible THYPE domain model extensions are shown: traffic cop and jeep as alternatives to detective and car respectively. System recovery from user interaction is possible via replanning using object instances of these new types of virtual character and narrative object (see Section 3 for detail).

domain model, represented using a typed language such as PDDL 2.1 [7], THYPE outputs alternative suggestions for types of virtual characters and narrative objects in the domain.

In this paper we overview the THYPE algorithm and evaluate its ability to increase domain model robustness and diversity. We also embed it within an extensible modular framework that allows it to be combined with other extension mechanisms. We present the results of evaluation in a range of domains which support our expectations that THYPE extensions make models more robust and diverse both on their own and in conjunction with other extensions.

2 RELATED WORK

A growing body of work in the AI Planning literature has considered the implications of incomplete models. In [2, 33] the planning model is extended during execution in order to incorporate initially unknown information. In [2], 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 re-planning, which can lead to shorter plans. In [2], it is assumed that the types of objects are known a priori, whereas a more general framework is developed in [32, 33], where the surrounding objects are used to build a context and a general image labeller is used to identify objects. *Model-lite* planning [12, 31] 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 [8] which was later extended to planning task revision [9] which considers how a given problem description can be updated when a plan solution is not possible. We observe that the approaches of [8] and [9] 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 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 [25] and the PDDL plug-in to VSCODE of Dolejsi et al [5], along with approaches to support the maintenance of existing models [1] and other approaches that aim to automate the process entirely, such as the LOCM family of algorithms (e.g. [4] and [3]), the FRAMER approach of [14], Janghorbani et al [11] and Walsh et al [26]. 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 only approach which has sought to automatically generate extensions to partially developed narrative domain models is the ANTON approach of Porteous et al [19]. With ANTON, domain models were analysed, during off-line processing, for the absence of actions enabling reverse transitions. Such missing transitions were used to suggest new actions to be used to extend the domain model. With THYPE we introduce an approach to domain model extension which is complementary to ANTON. Indeed, in our evaluation we demonstrate that combination of both approaches results in domains with enhanced robustness.

3 MOTIVATING EXAMPLE

To motivate our approach to domain model extension we consider a narrative domain model loosely based on the crime drama domain introduced in [13]. This domain features a range of different types of characters including police detectives and inspectors, along with citizens. The baseline domain model can generate narrative plans that see virtual characters committing crimes, moving around between locations in the world, leaving clues, being arrested, getting angry and even playing basketball to calm down. Figure 1 shows an example plan for control of such virtual character behaviours.

When embedded in an interactive narrative system, planning operates inside an execution loop where user interaction can change the state of the planning world and hence require re-planning or

repair (as in Young's "gun" example [34]). Hence in order to be able to respond to dynamic changes to the story world and to continue the story through to the intended ending, which tends to be one of a limited set of domain values, the domain model needs to be robust i.e. able to recover. In Figure 1 some possible user interactions are shown: ① the user renders the detective unable to arrest the suspect (e.g. they shoot the detective); ② renders the car unroadworthy (e.g. shoot the car tyres). The effects of such user interactions simply make the original goal impossible to reach and ignore real-world remediation; the latter could be captured through making the model more robust and diverse as follows:

- The model can be made more robust with the addition of different types of virtual characters and narrative objects. For example different type of police officer to perform the arrest, such as a *traffic cop* and different type of motor vehicle, such as *jeep*.
- Diversity relates to generation of diverse and consistent narratives: domain model consistency is something which would be enhanced via system generated extensions (i.e. no human error). THYPE extension would increase diversity as new virtual characters and narrative objects open up a wider range of narratives.

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 detective and car in Figure 1). This is predicated on the fact that linguistic hierarchies such as ConceptNet [15, 24], can be used as partial ontologies and for common sense reasoning.

We also note that robustness and diversity can be increased by the addition of missing actions capturing reverse state transitions (the result reported in [19]). For example, in the crime example this could include generation of a missing action that repairs a flat tyre. In our evaluation (Section 6) we show that combination of THYPE with this approach further enhances robustness and diversity.

4 GENERATING ALTERNATIVE TYPES

We introduce an algorithm for automatically generating plausible additional types of virtual characters and narrative objects with which to extend an existing domain model[1]. The algorithm 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 algorithm aims 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 algorithm as **THYPE**, to denote the extraction of **T**ypes from **HY**ponyms and hy**PE**rnyms.

[1]We assume a typed planning representation such as PDDL 2.1 [7, 16].

Algorithm 1 THYPE: generate alternate types of objects.

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

12: function FILTERCOMMONSENSE(ho, t)
13:     for all h ∈ ho do
14:         if ISRELATEDCN(h, t) ∧ ¬ISANTONYMCN(h, t) ∧
15:           ¬ISDISTINCTFROMCN(h, t) then
16:             ho′ ← ho′ + h
17:         end if
18:     end for
19:     if LENGTH(ho′) > MAX_NUM then
20:         for all h ∈ ho′ ∧ ¬ISACN(h, t) do
21:             ho′ ← ho′ − h
22:         end for
23:         if LENGTH(ho′) < MIN_NUM then
24:             ho′ ← GETTOPRELATED(ho, MAX_NUM)
25:         end if
26:     end if
27:     return ho′
28: end function
```

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 [6], although approaches such as Word2Vec [22] 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 [22].

At this stage there can be multiple meanings of a word and hence 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 [17].

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*

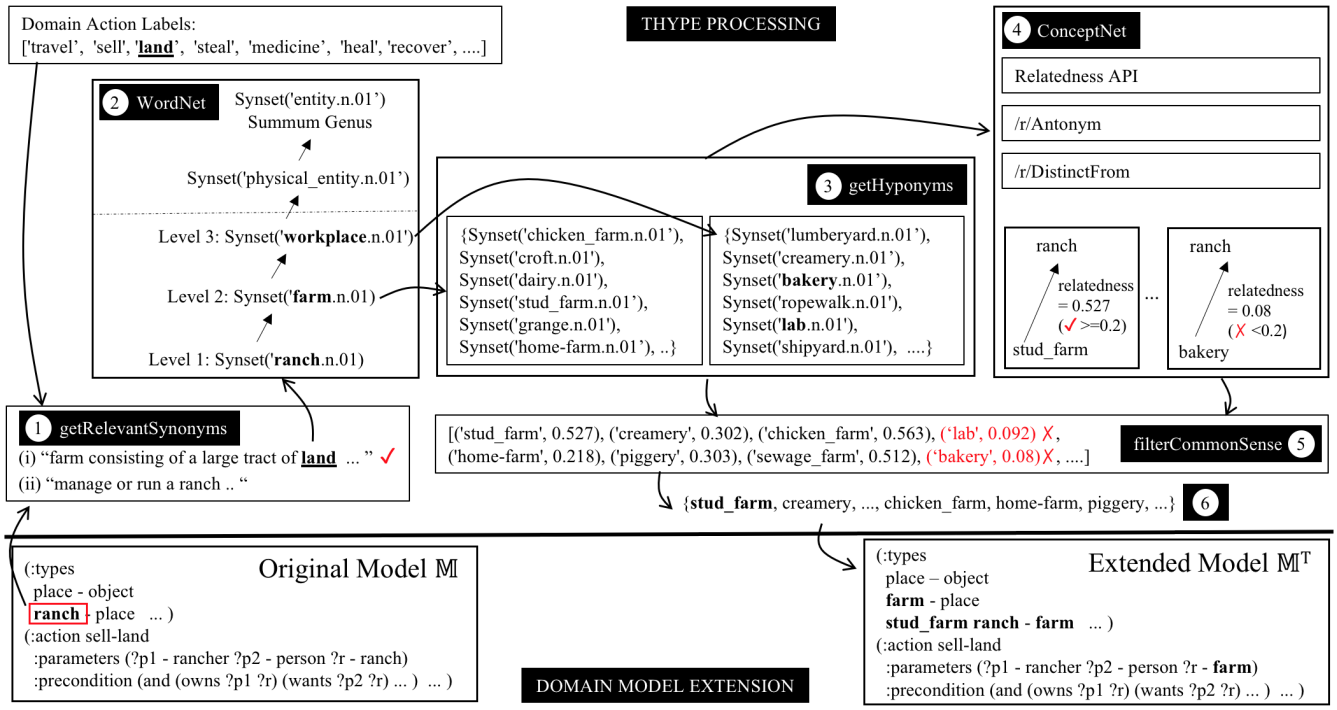

Figure 2: Example of THYPE generation for *ranch* from original model: (1) use action labels to filter meanings of the term; (2) WordNet hierarchies are accessed to level 3 to get Hypernyms; (3) get Hyponyms for all selected Hypernyms; (4) ConceptNet lookup is used as part of common sense filtering;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

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 [15, 24], a common sense knowledge based 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 returns a value greater or equal than 0.2; this value was found to be adequate experimentally). 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).

We have implemented this algorithm[2] and used it in experiments with a range of narrative domains taken from those in the literature.

[2]Our implementation is in Python 3. We use the packages `nltk` (with its corpus *wordnet*) and `requests` (to query ConceptNet's REST API).

	Type	①	②	③	④	Type	①	②	③	④
Aladdin	dragon	4	27	20 ✓	-	knight	2	19	14 ✓	-
	genie	1	22	6 ✓	-	princess	1	57	12 ✓	-
	king	10	15	10 ✓	-					
Crime	car	5	54	40	24 ✓	detective	2	34	22	20 ✓
	citizen	1	412	36	20 ✓	inspector	2	31	21	20 ✓
	crime	2	50	23	20 ✓	place	16	77	16 ✓	-
Medical	doctor	4	24	20 ✓	-	patient	2	416	50	20 ✓
	family	8	62	11 ✓	-	symptom	2	122	86	20 ✓
	nurse	2	26	16 ✓	-	treatment	4	64	26	20 ✓
Red	axe	1	28	20 ✓	-	house	12	143	54	27 ✓
	hunter	4	149	15 ✓	-	girl	5	112	27	20 ✓
	grand-mother	1	5	4 ✓	-	wolf	5	19	15 ✓	-
Western	city	3	22	15 ✓	-	sheriff	1	31	15 ✓	-
	rancher	1	28	10 ✓	-	sickness	3	25	19 ✓	-
	ranch	2	45	12 ✓	-	son	2	8	6 ✓	-
	saloon	3	111	44	20 ✓					

Figure 3: Alternatives suggested by THYPE. For each domain: ① #meanings for each type of character and narrative object; ② #Hyponyms output by GETHYPONYMS; ③ #alternatives after FILTERCOMMONSENSE; ④ (if required) #alternatives after relatedness reasoning with ¬ISACN and GETTOPRELATED. For the majority, over 90% marked ✓, the number of alternatives is deemed acceptable (@20 alternatives).

The results are shown in Figure 3. For each domain model and each type in the domain the figure lists the number of alternatives for each step of the processing. We discuss these results further in the evaluation in Section 6.1 but observe here that in the vast majority of cases the number of alternatives proposed is acceptable.

EXAMPLE. As illustration consider the application of THYPE to *ranch* from our PDDL encoding of a Western domain [28] (Figure 2). 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*, GETRELEVANT SYNONYMS 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.

Domain	Good %	Good #	Unsure %	Unsure #	Poor %	Poor #	Total#
Aladdin	73	45	19	12	8	5	62
Crime Drama	45	54	22	26	33	40	120
Medical Drama	54	58	9	10	36	39	107
Red Riding Hood	51	52	34	34	15	15	101
Western	47	46	20	19	33	32	97
Total	52	255	21	101	27	131	487

Figure 4: THYPE Plausibility Rankings (gold standard) : Good (clearly plausible), Poor (not plausible) and Unsure. Across domains 50% of alternatives ranked as Good (plausible).

Also illustrated in Figure 2 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).

5 DOMAIN MODEL EXTENSION

We introduce 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 new types generated using THYPE (Section 5.1). We also consider the combination of multiple extensions and to illustrate this we extend the domain model with extensions generated using ANTON[3] [19] (Section 5.2).

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

5.1 $\mathbb{M}^T$ 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 6. 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

	Type	①	②		Type	①	②	
Aladdin	Dragon	3	2.14		Knight	2	2.01	✓✓
	Genie	3	2.42		Princess	1.75	2.04	✓✓
	King	2.75	2.47	✓				
Crime	Car	2.8	2.34		Detective	2.5	2.06	
	Citizen	1.5	2.19	✓✓	Inspector	2.25	2.17	✓✓
	Crime	2	2.22	✓✓	Place	1	2.2	✓✓
Medical	Doctor	2	1.95	✓	Patient	1.25	1.63	✓✓
	Family	1.7	2.04	✓✓	Symptom	3	2.76	✓
	Nurse	2.5	2.35	✓	Treatment	2	2.27	✓✓
Red	Axe	1.8	2.12	✓✓	House	2.7	2.46	✓
	Girl	2	2.07	✓✓	Hunter	2.75	2.04	
	Grandmother	2.5	2.77	✓✓	Wolf	2	2.15	✓✓
Western	City	2.25	2.3	✓✓	Sheriff	2	2.14	✓✓
	Rancher	2.66	2.49	✓	Sickness	2.4	2.47	✓✓
	Ranch	2	2.28	✓✓	Son	2	2.65	✓✓
	Saloon	1.4	1.57	✓✓				

Figure 5: User Validation of Plausibility Rankings (random sample of 25% of alternatives): ① gold standard rankings; ② User rankings. Numbers obtained by assigning scores (Good=3, Unsure=2, Poor=1) and averaging across #alternatives for each type. Overall the gold standard rankings are more conservative (30% of user rankings are higher (✓✓); 23% are within 0.25 (✓). These results provide support for the gold standard overall rankings (see text for more detail).

$\mathbb{M}$ 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 6).

5.2 $\mathbb{M}^A$ Domain Extension with ANTON

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[3] [19].

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 [19], 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.

[3]We use our original implementation of ANTON as introduced in Porteous et al [19]

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

Figure 6: Aladdin Example [21] 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.

- **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.

6 EVALUATION

There were two parts to the evaluation: (i) assess the ability of THYPE to generate alternatives in the context of a given narrative domain (both the number of alternatives proposed along with their plausibility); (ii) assess the ability of THYPE model extensions to increase robustness of extended domain models, both individually and in combination with other extension mechanisms.

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 [21], Crime Drama [13], Medical Drama [19], Red Riding Hood [20], Western [28].

6.1 THYPE generated extensions

Our implementation of THYPE (see Algorithm 1) used WordNet 3.0 [6] and ConceptNet [15, 24]. The results of our experiments with this implementation are shown in Figure 3.

6.1.1 Number of Alternatives. For our experiments we took a "reasonable" number of alternatives suggested by THYPE to be 20. Figure 3 shows the number of types suggested for each of the domains used in the evaluation. We observe that for the majority of terms, over 90%, the number of THYPE generated alternatives is reasonable i.e. within this bound. These numbers are promising in terms of the ability of the approach to control the numbers of alternatives generated. There are a few exceptions where the number is over this upper bound (e.g. *car* and *house* in Figure 3), which result from a preference in the common sense filtering for hyponyms which are in IsA relations. It is only when the number of hyponyms falls below *MIN_NUM* that this preference is relaxed and the set of hyponyms are sorted by "relatedness" and the highest ranked *MAX_NUM* are selected (20 in our experiments).

6.1.2 Plausibility of Alternatives. To assess plausibility we assigned a ranking to each of the alternatives: **Good** if we thought it

Domain	Type $\mathbb{M}$	Types $\mathbb{M}^T$, $\mathbb{M}^{TA}$	Parent $\mathbb{M}^T$, $\mathbb{M}^{TA}$
Aladdin	king	emperor	sovereign
	knight	sir	maleAristocrat
	genie	shaitan	spirit
Crime	car	jeep	vehicle
	detective	trafficCop	policeOfficer
Medical	doctor	extern	medicalPractitioner
Red	wolf	wild-dog	canine
	hunter	huntress	skilledWorker
Western	rancher	stockman	farmer
	sheriff	texasRanger	lawman

Figure 7: Figure shows example types selected from the THYPE suggestions and used to extend $\mathbb{M}$ to form $\mathbb{M}^T$ and $\mathbb{M}^{TA}$. Also added are system generated parent types which group related types and are used in PDDL action parameters (e.g. king and emperor of type sovereign as in Figure 6).

was clearly a plausible alternative; **Poor** if we thought it was clearly not a plausible alternative; or **Unsure** if it wasn't clear. The totals of our rankings, the gold standard, are summarised in Figure 4. In order to validate these rankings we conducted a user study. Since the total number of alternatives across all types and domains was close to 500, for the study we chose to randomly sample 25% of the alternatives and used those in the validation study. The study itself consisted of a series of questions which were put to a sample of 30 native English speaking adults via an online questionnaire. For each narrative domain and for each of the selected types, users were asked to rank whether they thought the alternative was plausible, using the same rankings as above: **Good, Poor** or **Unsure**.

Results of the user validation study are summarised in Figure 5: ① are the rankings taken from the gold standard and ② the rankings from the users. The numbers were obtained by assigning scores to rankings (**Good**=3, **Unsure**=2 and **Poor**=1) and averaging across the alternatives for each type. We allowed for some variation and took averaged user rankings that were higher or within .25 of the gold standard as validating ours (marked with ✓✓ and ✓ respectively). Here we have 83% correspondence to our gold standard rankings which represents a promising initial validation of the plausibility of the THYPE generated alternatives. Given this initial validation of our plausibility rankings, the overall results in Figure 5 are very encouraging: with over 50% of all suggestions being ranked plausible and only 28% felt to be not plausible. This is important as it shows THYPE can generate a high number of alternatives that appear plausible to laypeople which increases our expectation that expert domain modellers (i.e. authors) will also find these useful.

6.2 Robustness of Extended Domain Models

6.2.1 Generating Extended Domain Models. For each of the domains used in the evaluation, extended versions were created starting from the original domain model $\mathbb{M}$, as follows:

- $\mathbb{M}^T$ was generated by selecting additional types from those proposed by the system as shown in Figure 7. 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 of adding a digit to the type name (e.g. for Aladdin, *emperor1* of type *emperor*).
- $\mathbb{M}^A$ was generated for individual comparison with $\mathbb{M}^T$. For each of the domains the number of ANTON^3 actions added was:

	$\mathbb{M}$	$\mathbb{M}^A$		$\mathbb{M}$	$\mathbb{M}^A$	$\mathbb{M}$	$\mathbb{M}^A$
Aladdin	12	21	Crime	9	11	Medical	10	19
Red	5	9	Western	19	25			

- $\mathbb{M}^{TA}$ was generated by adding the ANTON^3 extensions into $\mathbb{M}^T$.

6.2.2 Simulation of Execution Failure. 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 were selected at random.

For each run the simulation firstly generated a narrative plan for the input planning problem (using METRIC-FF [10]) 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 8.

To evaluate the robustness of THYPE extended domain models firstly consider the results for $\mathbb{M}^T$, the THYPE extended domain model. 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 80% 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.

Another aspect of the THYPE extended domain models is comparison with an existing extension mechanism, ANTON^3. The results for $\mathbb{M}^T$ show that it performs consistently well, yielding similar performance improvements to $\mathbb{M}^A$. Taken over all of the domains, performance of $\mathbb{M}^T$ is consistently good and outperforms $\mathbb{M}^A$ on 48% of problem instances: a very encouraging result.

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 ANTON 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

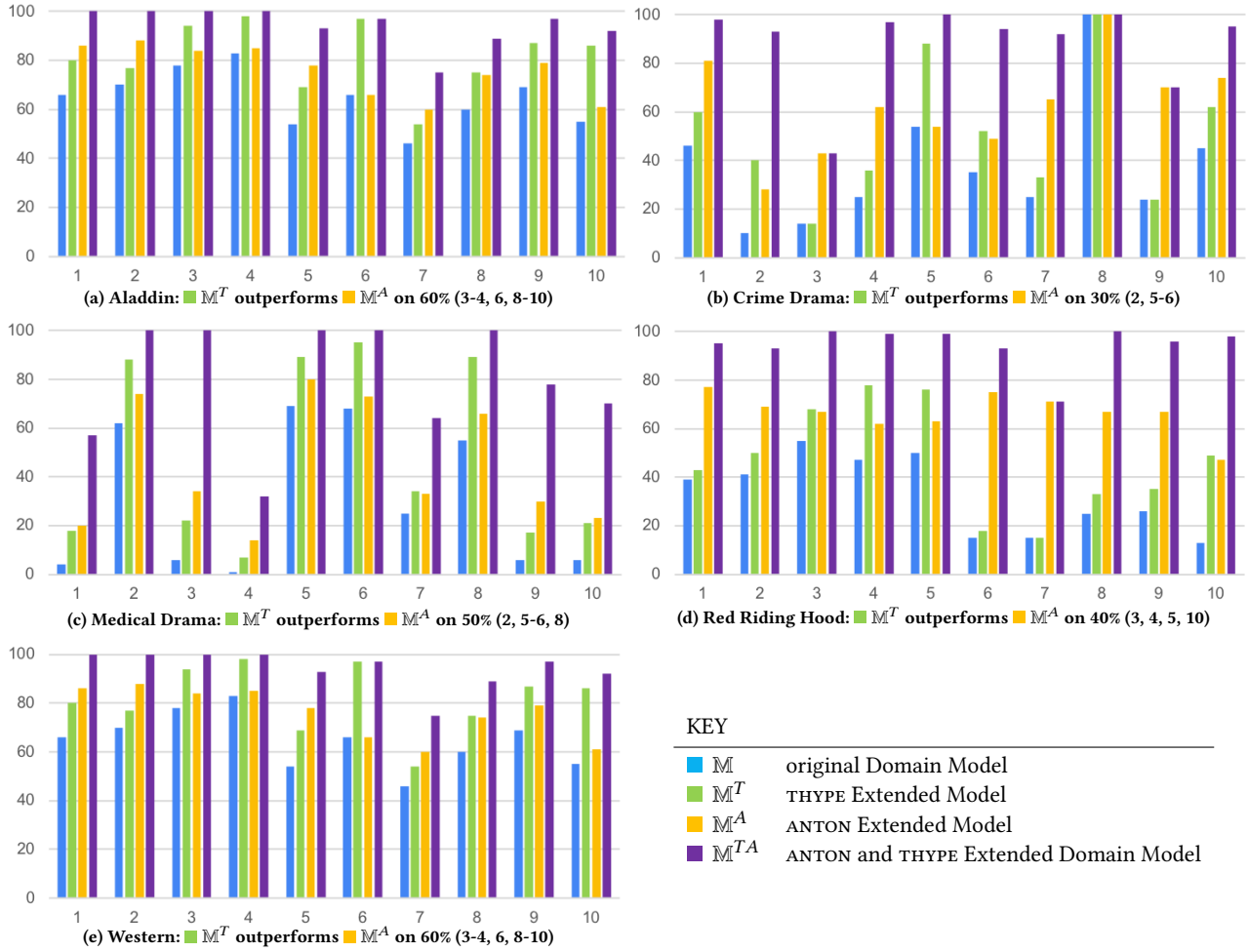

Figure 8: Robustness Results: comparing ability of $\mathbb{M}^T$ and $\mathbb{M}^{TA}$ to continue to original goal in execution failure simulation (5 domains, 10 problems, 100 fail-restarts on each); $\mathbb{M}$ and $\mathbb{M}^A$ included for comparison. (1) Performance of $\mathbb{M}^T$ is consistently good across all domains and very similar to $\mathbb{M}^A$: $\mathbb{M}^T$ outperforms on 48% of problem instances. (2) $\mathbb{M}^{TA}$ outperforms all other models and demonstrates the performance gains that can result from combining extensions (see text for detail).

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: both for $\mathbb{M}^T$ and $\mathbb{M}^{TA}$.

7 CONCLUSION

Defining complete planning models upfront is challenging, especially for domains where plan failure can occur such as interactive narrative. In this work we have presented an off-line approach to domain model extension aimed at supporting a domain modeller extend a partially developed model. We make no assumptions about how the domains have been modelled and hence for evaluation used domains from the literature, familiar to the research community.

We have focused on two aspects of domain model extension. The first was our introduction of a novel mechanism, THYPE, for generation of alternative types to extend a domain and help increase its robustness. Results of a user study showed the majority of THYPE generated alternatives to be plausible. Results of experiments showed how the addition of system suggested types can increase the likelihood of continuation to the original goal across a range of domains. The results also show that combining different extension mechanisms, for our experiments THYPE and ANTON[3], results in greater increases in domain robustness (in terms of recovery from plan failure).

In future work, we plan to: extend THYPE to further reason about the levels of words in semantic hierarchies to get clues to the construction of detailed type hierarchies; and explore if, for specific domains, specialized knowledge graphs improve common sense filtering (e.g. using the graph in [23] with medical domains).

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; and by funds from DSI Collaborative Grant CR-0016.

REFERENCES

[1] D. Bryce, J. Benton, and M. W. Boldt. 2016. Maintaining Evolving Domain Models. In *Proc. of the 25th Int. Joint Conference on AI (IJCAI)*.

[2] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, V. De Carolis, D. Lane, and F. Maurelli. 2015. Dynamically Extending Planning Models using an Ontology. *Planning and Robotics* (2015).

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

[4] S Cresswell, T Mccluskey, and M West. 2009. Acquisition of Object-Centred Domain Models from Planning Examples. In *Proc. of the 19th Int. Conference on Automated Planning and Scheduling (ICAPS)*.

[5] J. Dolejsi, D. Long, M. Fox, and G. Besancon. 2018. PDDL Authoring and Validation Environment for Building end-to-end Planning Solutions. In *Proc. of the 28th Int. Conference on Automated Planning and Scheduling (ICAPS)*.

[6] C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press. https://wordnet.princeton.edu/

[7] Maria Fox and Derek Long. 2003. PDDL2.1: An Extension to PDDLfor Expressing Temporal Planning Domains. *J. Artif. Intell. Res.* 20 (2003), 61–124.

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

[9] A. Herzig, V. Menezes, L. Nunes de Barros, and R. Wassermann. 2014. On the revision of planning tasks. In *Proc. of the 21st European Conference on AI (ECAI)*.

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

[11] S. Janghorbani, A. Modi, J. Buhmann, and M. Kapadia. 2019. Domain Authoring Assistant for Intelligent Virtual Agent. In *Proc. of the 18th Int. Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*.

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

[13] B. Kartal, J. Koenig, and S. J. Guy. 2014. User-driven narrative variation in large story domains using monte carlo tree search. In *Proceedings of the 13th Int. Conf. on Autonomous Agents and MultiAgent Systems, (AAMAS)*.

[14] A. Lindsay, J. Read, J. F. Ferreira, T. Hayton, J. Porteous, and P. J. Gregory. 2017. Framer: Planning Models from Natural Language Action Descriptions. In *Proc. of the 27th Int. Conference on Automated Planning and Scheduling (ICAPS)*.

[15] H. Liu and P. Singh. 2004. ConceptNet—a practical commonsense reasoning tool-kit. *BT technology journal* 22, 4 (2004), 211–226.

[16] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. 1998. *PDDL-the planning domain definition language*. Technical Report. Yale University.

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

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

[19] J Porteous, A Lindsay, J Read, M Truran, and M Cavazza. 2015. Automated Extension of Narrative Planning Domains with Antonymic Operators. In *Proc. Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*.

[20] M. O. Riedl. 2009. Incorporating Authorial Intent into Generative Narrative Systems. In *Proc. of AAAI Spring Symp. Intelligent Narrative Technologies*.

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

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

[23] J. Siddle, A. Lindsay, J. F. Ferreira, J. Porteous, J. Read, F. Charles, M. Cavazza, and G. Georg. 2017. Visualization of Patient Behavior from Natural Language Recommendations. In *Proceedings of the Knowledge Capture Conference (K-CAP)*. ACM.

[24] Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[25] Tiago Stegun Vaquero, José Reinaldo Silva, and J. Christopher Beck. 2013. Post-design Analysis for Building and Refining AI Planning Systems. *Eng. Appl. Artif. Intell.* 26, 8 (Sept. 2013), 1967–1979.

[26] Thomas J Walsh and Michael L Littman. 2008. Efficient Learning of Action Schemas and Web-Service Descriptions. In *AAAI*. 714 – 719.

[27] P Wang, J Rowe, W Min, B Mott, and J Lester. 2018. High-Fidelity Simulated Players for Interactive Narrative Planning. In *Proc. of the 27th Int. Joint Conference on Artificial Intelligence (IJCAI)*.

[28] S. Ware. (Accessed: 11-11-2019). Glaive Narrative Planner. ((Accessed: 11-11-2019)). https://www.cs.uky.edu/~sgware/projects/glaive/

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

[30] L. Yao, N. Peng, R. M. Weischedel, K. Knight, D.n Zhao, and R. Yan. 2019. Plan-And-Write: Towards Better Automatic Storytelling. In *Proc. of the 33rd National Conference on AI (AAAI)*.

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

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

[33] Jay Young, Lars Kunze, Valerio Basile, Elena Cabrio, Nick Hawes, and Barbara Caputo. 2017. Semantic Web-Mining and Deep Vision for Lifelong Object Discovery. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[34] R. Michael Young. 1999. Notes on the Use of Plan Structures in the Creation of Interactive Plot. In *AAAI Fall Symposium on Narrative Intelligence*.