Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)

Framer: Planning Models from Natural Language Action Descriptions

Alan Lindsay,[1] Jonathon Read,[2] João F. Ferreira,[1,3]
Thomas Hayton,[1] Julie Porteous,[1] Peter Gregory[1]
[1]Digital Futures Institute, School of Computing, Teesside University, UK.
[2]Ocado Technology, Hatfield, UK.
[3]HASLab/INESC TEC, Universidade do Minho, 4704-553 Braga, Portugal.
firstinitial.lastname@tees.ac.uk | jonathon.read@ocado.com

Abstract

In this paper, we describe an approach for learning planning domain models directly from natural language (NL) descriptions of activity sequences. The modelling problem has been identified as a bottleneck for the widespread exploitation of various technologies in Artificial Intelligence, including automated planners. There have been great advances in modelling assisting and model generation tools, including a wide range of domain model acquisition tools. However, for modelling tools, there is the underlying assumption that the user can formulate the problem using some formal language. And even in the case of the domain model acquisition tools, there is still a requirement to specify input plans in an easily machine readable format. Providing this type of input is impractical for many potential users. This motivates us to generate planning domain models directly from NL descriptions, as this would provide an important step in extending the widespread adoption of planning techniques. We start from NL descriptions of actions and use NL analysis to construct structured representations, from which we construct formal representations of the action sequences. The generated action sequences provide the necessary structured input for inducing a PDDL domain, using domain model acquisition technology. In order to capture a concise planning model, we use an estimate of functional similarity, so sentences that describe similar behaviours are represented by the same planning operator. We validate our approach with a user study, where participants are tasked with describing the activities occurring in several videos. Then our system is used to learn planning domain models using the participants' NL input. We demonstrate that our approach is effective at learning models on these tasks.

Introduction

Modelling problems appropriately for use by a computer program has been identified as a key bottleneck in the exploitation of various AI technologies. In Automated Planning, this has inspired a growing body of work that aims to support the modelling process including domain acquisition tools, which learn a formal domain model of a system from some form of input data. There is interest in applying domain model acquisition across a range of research and application areas. For example within the business process community (Hoffmann, Weber, and Kraft 2012) and space applications (Frank et al. 2011). An extended version of the *LOCM* domain model acquisition system (Cresswell, McCluskey, and West 2009) has also been used to help in the development of a puzzle game (Ersen and Sariel 2015) based on spatio-temporal reasoning. Web Service Composition is another area in which domain model acquisition techniques have been used (Walsh and Littman 2008). These tools vary in the specifics of the input language, such as example action sequences (Cresswell, McCluskey, and West 2009; Cresswell and Gregory 2011), or action sequences and a partial domain model (McCluskey et al. 2009; Richardson 2008); the query system by which they acquire the input data, which is typically static training sets, although there are examples working with an interactive querying system (Walsh and Littman 2008; Mehta, Tadepalli, and Fern 2011); and the target model language, including STRIPS (Cresswell, McCluskey, and West 2009; Cresswell and Gregory 2011), probabilistic (Mourão, Petrick, and Steedman 2010), and numeric (Gregory and Lindsay 2016; Hayton et al. 2016). However, in each case the user is left the responsibility of defining a formal representation for the solution.

Defining these logical formalisms and applying them consistently requires time and experience in both the target domain and in the representation language, which many potential users will not have. It is therefore important to consider alternative input languages, such as Natural Language (Goldwasser and Roth 2011). Natural Language (NL) input is the most natural way for humans to interact and it is no surprise that there is much interest in using NL as input for computer systems. In day-to-day life, Siri and its competitors are controlled by simple spoken word input, but can activate complex procedures on our phones. In the RoboCup@Home competitions robots are controlled by task descriptions and are automatically translated into a series of simple actions that can be performed on the robot. And NL *lessons* have been used to learn partial representations of the world dynamics for game-like environments (Goldwasser and Roth 2011). A key aspect of these systems is an underlying language, which the NL input is mapped onto. For example, in the case of RoboCup@Home, an input of 'go to the living room' might be mapped onto quite a different representation, using the action name 'move' and requiring a set of parameters that break the movement into smaller steps between connected

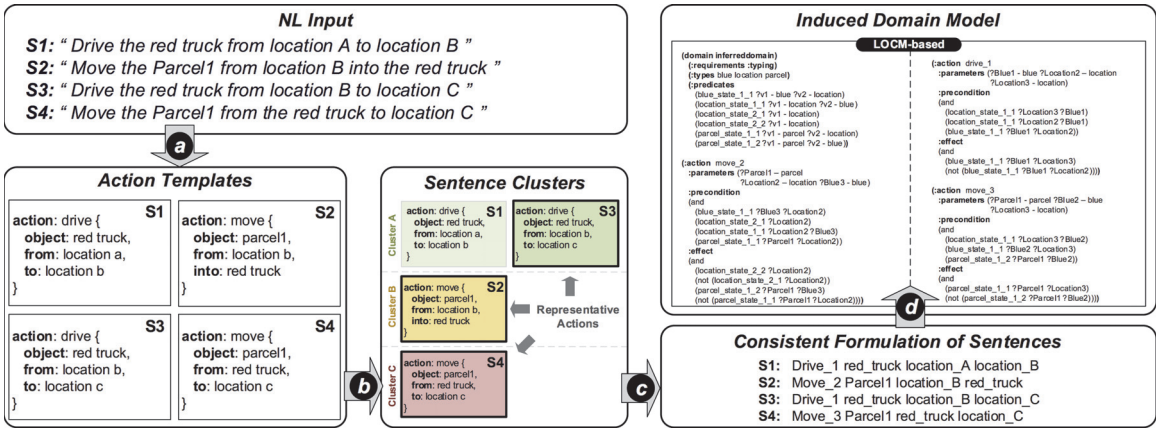


Figure 1: System overview: NL sentences are transformed into reduced representations (action templates) **(a)** that are clustered based on similarity **(b)**. Consistent formulations of the original sentences are then extracted **(c)** and a PDDL domain model is induced using the domain model acquisition tool LOCM **(d)**.

rooms. Therefore in these domains there is still a requirement for a domain engineer to develop the formal representation.

In this work we consider the problem of generating planning action representations automatically from a collection of NL sentences that are descriptions of actions. The benefit of this is that we can target domain model acquisition systems, but do not have to manually define a formal representation for the solution. The key challenge is to generate an appropriate formal representation, from which a general and concise model can be induced, and that best represents the input sentences. Our approach, illustrated in Figure 1, can be summarised as follows. The first step (a) generates a reduced representation of each sentence, an action template, which captures the main action as well as the objects that are mentioned and an indication of their roles in the sentence. The second step (b) uses a measure of *functional* similarity, based on the reduced representations, in order to cluster sentences into operator sets. A consistent action representation is then established by defining a mapping between the members of each cluster. The action sequences are then rewritten (c) using these action representations and used to induce a PDDL domain model (d) using the domain model acquisition tool, *LOCM2* (Cresswell and Gregory 2011).

In our implemented system user interaction is allowed during the development of the representation, although it is not required at any step. There are three key aspects where the user can influence the representation: the user can rephrase sentences where an action was not detected; they can modify the automatically selected sentence clusters (operator sets); and finally they can correct or fill-in missing action roles (parameters). In the evaluation, we demonstrate that our approach can accurately identify behaviour groups in NL sentences describing actions, rewrite related sentences using a single representation and be robust to certain inconsistencies. These sentences were used to induce a PDDL model in three domains: Towers of Hanoi, Logistics and Tyreworld.

Background

Here we present relevant background in planning, domain model acquisition and NLP approaches.

The description of a planning problem in PDDL (McDermott et al. 1998) 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. It is the domain model which is the target output of our approach. A domain model is a tuple, $\mathbb{D} = \langle \mathbb{O}, \mathbb{P} \rangle$, defining the sets of operators, $\mathbb{O}$, and predicates, $\mathbb{P}$. An operator, $\mathcal{O} \in \mathbb{O}$, is represented by an *operator header*: a unique symbol (operator name) and a list of typed variables (parameters). The *operator body* consists of three sets of predicates: the preconditions, and the add and delete effects. An action, $\mathcal{A}$, is a planning operator, $\mathcal{O}$, that has been instantiated with problem constants (parameters, preconditions and effects) and an *action header* is a name and a list of constants (the instantiated parameters).

Our approach builds over work in domain model acquisition, which exploits assumptions in the structure of the input to learn planning models using a minimal input language (Cresswell and Gregory 2011). *LOCM* (Cresswell, McCluskey, and West 2009; Cresswell and Gregory 2011; Gregory and Cresswell 2015; Gregory and Lindsay 2016) is a family of domain model acquisition systems that operate from a set of example plans and generates a planning domain in the standard Planning Domain Definition Language (PDDL). The *LOCM* procedure uncovers the structure embedded in the action sequences in order to identify FSM descriptions for each object type. In order to do this, the system observes the transitions that an individual world object makes and then generalises these behaviours to types based on the parameter position that the objects take in each action. The structures captured by the FSMs are then used to define a domain model and from this and the input sequences, problem descriptions can be generated.

We use Stanford CoreNLP (Manning et al. 2014), a publicly-available and widely-used annotation pipeline for

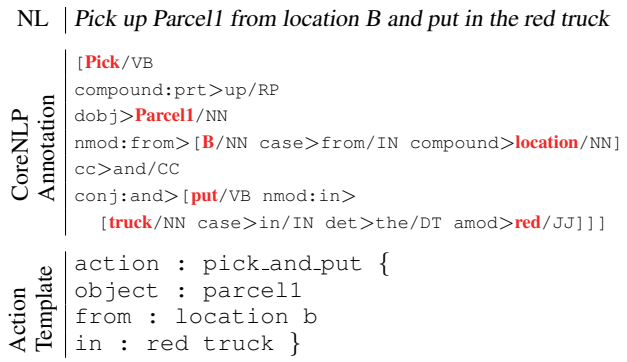


Figure 2: Example NL sentence input, with its CoreNLP annotation and resulting action template after rewrite rules: from CoreNLP annotation, **verb**, **subject** and **object** of the sentence form the action name and arguments (see text).

natural language analysis. Of most relevance to the current work are the syntactic parsing annotations CoreNLP produces. Syntactic analysis in CoreNLP is a two-stage process. Firstly, phrase structure trees are generated using statistical analysis of datasets containing many examples of manually annotated sentence parses (Klein and Manning 2003). Secondly, these phrase structure trees are converted to dependency parse graphs using a series of manually-curated rules based on patterns observed in the phrase structure trees (de Marneffe, MacCartney, and Manning 2006).

Extracting action templates from NL input

The first step in our approach (see Figure 1(a)) is generation of action templates: reduced representations of input sentences, which capture the main action, objects that are mentioned and an indication of their roles in the sentence. For this we utilise the dependency graphs output by CoreNLP, illustrated in Figure 2. The structure of this middle representation must be further simplified to move closer to a predicate logic representation. This is achieved through a recursive set of rules that crawl the dependency graph, transforming the relations based on their types. Most importantly, the root verb of the sentence forms the basis of the action name, while the verb's subject and objects form the arguments. Conjunctions introduce new clauses of the sentence, which form further predicates. Other relation types such as modifiers and compounds are used to transform the names of the predicates and arguments. The input of this process is a sentence and the output is a collection of slot label (e.g., *in*) and slot filler (e.g., *red truck*) pairs. Where an action is detected, one or more of the slot fillers will identify action name elements. The other slot labels indicate the associated slot filler's *role* in the sentence.

There are many possible ways that users could formulate sentences to describe what happens during a single action. This structure therefore provides a reduced representation that identifies the main actors in an action and the roles that they play in the sentence. It should be noted that our approach relies on consistent object references (e.g., 'the red truck' in Figure 1) throughout the action sequence. In order to maintain the same level of granularity as the input descriptions, internal action predicates are merged to join a single action template. In Figure 2 the internal `put` predicate is merged into the `pick` action, creating a single action template for `pick_and_put`. It is important to notice that this structure is similar to the one generated for S2 in Figure 1 for quite a different sentence structure.

Partitioning sentences into operator sets

The next step is to identify an appropriate partitioning of the reduced sentences that best represent them, in order to derive a set of consistent and general operator descriptions (as shown in Figure 1(b)). Our approach is to define a distance measure between sentence pairs and use it to identify clusters. As each cluster represents a planning operator in the final domain model, we also consider, in the following discussion, how the clustering can be controlled in order to effect the generality of the generated domain model.

Functional distance between sentences

The ideal distance measure would estimate the difference between the underlying functional process described in the sentences. This would require a rich description of the underlying process, whereas we consider how this can be estimated using the sentences in isolation. However, we do not want to directly estimate the distance between the sentences. Instead we want to estimate the structural difference between the sentences, with specific focus on the key action of the sentence. In particular, we want to exploit the structure of the generated templates (previous section), which identifies the main action of the sentence and provides a context for the purpose (or role) of each object in the sentence.

For example, consider these sentences:

S5 *The truck has moved from Aberdeen to Dundee*
S6 *A green box was put onto the truck in Dundee*
S7 *The truck was driven from Dundee to Stirling*

Sentences S5 and S7 are very similar in structure and probably describe a similar underlying behaviour. In particular, we observe that the words *moved* and *driven* are in fact often used interchangeably and that representing them with a single operator would lead to a more concise and general model. In contrast, sentence S6 differs in the specific verb as well as the structure of the sentence and therefore we would expect it to be represented in a planning model with a different planning operator. In general then we would expect that those templates that are *similar* to each other might be represented by a single operator. However, it is not as simple as collecting similar verbs. For example in Figure 1, S2 and S4 share the same verb, however, describe different behaviours, which can only be distinguished by the roles of the objects in the sentences.

We first describe our approach for estimating the similarity of individual symbols (i.e., the distance between them) and then build from this to a complete similarity measure.

Distance between terms In this application, we are specifically interested in whether words can be used in place of each other and are therefore synonymous with each other. We

use a collection of online lexical resources[1] in order to generate a set of weighted synonyms (a similar approach was used to find antonyms for action and predicate names in (Porteous et al. 2015; Lindsay et al. 2015)). Online lexical resources provide a source of typical synonyms without relying on the user to identify similar terms; however, the quality of output can be inconsistent and therefore it is prudent to combine several sources. Each source, S_i, can be seen as a function providing a vote and its score for each word pair can be normalised to a value between 0 and 1 (higher scores for higher correspondence). The sources can also be parameterised by the parts of speech (POS) of the generated synonyms, which is useful in the case of estimating action type similarity, as these tend to be verbs. We define the similarity function, $\texttt{SIM}(w_0, w_1, POS)$, which estimates the similarity between words w_0 and w_1, with the set of accepted POS, as:

$$\texttt{SIM}(w_0, w_1, POS) = \frac{1}{n} \sum_{i=1}^{n} S_i(w_0, w_1, POS)$$

In the case of multiple word terms, the above function is generalised using the Levenshtein distance (Levenshtein 1966) of the two word sequences (using a symbol for each word) and using the complement of the similarity scores (distance measure) as a partial match cost. This provides a measure of correspondence between the sequences, while also respecting ordering.

Similarity measure The similarity measure is based on the idea that if the actions involved in a pair of sentences are similar and the roles of the objects are similar then we expect that the function that the sentences are describing is similar.

The similarity measure for templates τ_0 and τ_1, denoted by $\delta(\tau_0, \tau_1)$, is computed from two values:

- $\texttt{SAN}(\tau_0, \tau_1)$: The similarity of the action names (entries in slots with label *action*) of τ_0 and τ_1;
- $\texttt{SSL}(\tau_0, \tau_1)$: The average similarity for each non-action slot label (each role) from τ_0 to τ_1, where for each role of τ_0 the closest matching role of τ_1 is selected.

These values rely on the similarity $\texttt{SIM}(w_0, w_1, POS)$, defined above, which is used with action names with the argument, $POS{=}verb$ and for roles with argument, $POS{=}*$ (any part of speech).

The similarity measure can be computed as:

$$\delta(\tau_0, \tau_1) = \gamma \times \texttt{SAN}(\tau_0, \tau_1) + (1{-}\gamma) \times \texttt{SSL}(\tau_0, \tau_1)$$

The parameter γ controls the importance in similarity between roles and action names.

Cluster-based approach to operator sets selection

Clustering identifies groups of elements, which are similar (or close) to the elements in their own group, while being dissimilar (or far away from) elements in other groups. It is a hard problem in general, especially in domains where the number of clusters cannot be guessed. However, it is a very well studied area and many off-the-shelf toolkits exist. In this specific problem we want to cluster the action templates into similar groups. We can use the distance matrix for the distance between templates pairs for clustering, which saves defining a projection of a template into a space.

We adopt the Partitioning Around Medoids (PAM) implementation of the k-medoids method (Kaufmann and Rousseeuw 1987). This approach partitions the data into k clusters, each associated with a representative data point, considered the most central in the cluster. The specific benefits for this work are that the algorithm partitions the objects and operates from the dissimilarity matrix of the data points, allowing us to use an arbitrary distance score.

Model generality Selecting an appropriate clustering is an interesting problem and one that will effect the generality of the final representation. There may be more than one correct partitioning of the sentences into correct behaviour groups. For example, consider the various encodings of stacking behaviours in PDDL: Depots, Towers of Hanoi, and multiple representations of Blocksworld. Our default approach is to calculate the average *silhouette* score for the clusters (Rousseeuw 1987), which evaluates the clusters by averaging the similarity within clusters and dissimilarity between clusters, with respect to the distance measure. This can only be evaluated accurately for at least 2 clusters, so we first test to determine whether more than 1 cluster is appropriate (Duda, Hart, and others 1973). The optimal average silhouette score indicates a good trade-off between the size of k and the amount of dissimilarity in each cluster.

Our system supports interaction at this stage, allowing the user to pick between different values of k, but also changing the clusters. It is interesting to notice that the user organises (their own) NL sentences into behaviour groups and therefore does not need to interpret any abstracted representation.

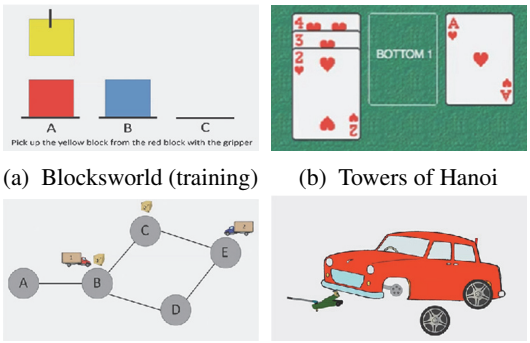
Generating a domain model

We have presented our approach for selecting the operator sets that determine the main language for the generated domain model. In this section we construct a planning model that represents the dynamics captured in the NL action descriptions. The first step is to define the action language of the planning model and this is achieved by demonstrating a consistent formalisation within each group of templates that have been partitioned into a single operator set (as shown in Figure 1, step (c)). This supports the rewriting of the sentences as sequences of action headers, which is a sufficient input for domain model acquisition (Figure 1, step (d)). We conclude the section by considering how missing values (parameters) can be addressed.

Formalised representation of the sentences

We use the centre most element (the medoid and therefore a natural output from the clustering algorithm) as the basis for the operator description. For the associated template we define an operator header as follows: the name is the concatenation of the terms with *action* slot labels (joined with a symbol, e.g., '-'); and the parameter list is represented by the (non-action) slot labels (i.e., the slot fillers represent instantiations of those parameters). The translation of the medoid sentence into the language of the action model is a

[1]Merriam-Webster http://www.dictionaryapi.com; Big Huge Thesaurus http://words.bighugelabs.com; Power Thesaurus http://www.powerthesaurus.org



(a) Blocksworld (training) (b) Towers of Hanoi

(c) Logistics (d) Tyreworld

Figure 3: Screenshots from the videos used in the evaluation.

space-separated concatenation of the name and the slot filler (multiple words joined with a symbol, e.g., '-'), for each of the slot labels in order.

To translate one of the other templates τ' of the cluster, we establish a mapping between the slot labels of the medoid template, $\tau^{\mathbf{M}}$, and those of τ'. This is done by establishing a best match between the slot labels of the templates, by maximising the overall similarity of labels, using the (previously defined) function: $\texttt{SIM}(\text{slot-label}_{\tau^{\mathbf{M}}}, \text{slot-label}_{\tau'}, *)$. The sentence is then constructed in a similar way, except for a specific slot label, the mapping is used to identify the corresponding slot label in τ' and use its entry instead of $\tau^{\mathbf{M}}$. In the case that a mapping is not found for a tag in $y\%$ of the members of a cluster then the tag is pruned. As in some cases a description will have more information than is necessary, this filtering process aims to identify the important parameters for the operator.

The output of this process are sequences of action headers that each instantiate one of the operator headers implied by the behaviour groups. It has been shown that, within certain restrictions, sequences of action headers provide sufficient evidence of the dynamic structure of planning domains and can be used directly to induce a domain model (Cresswell and Gregory 2011). It is presenting the key objects in consistent orderings (achieved through our mapping approach) that allows *LOCM* to uncover the inherent structures. It is then the job of *LOCM* to identify the key relationships between the parameters of actions, which it then encodes as predicates in the induced domain model.

Missing parameter values

In practice, a user may not always mention all of the objects involved in an action, or may not be consistent with the objects mentioned. There is no guaranteed method of inferring the missing parameters as the correct dynamics of the system (even if they can be expressed in STRIPS) are unknown. Thus the system supports user interaction at this stage, allowing the user to both fill in missing parameters and correct incorrect parameters. The representative sentence for each cluster is used as a template to rewrite each of the sentences of the cluster. The slot fillers of the representative sentence are replaced with the relevant fillers from the member sentence (see the discussion for plan rewriting in the evaluation). Missing values are indicated and can be filled in by the user. The main benefit of this approach is that the user can interact with the system using only NL.

The default behaviour in this case is to break the plan into two action sequences by removing the partially specified action. That is, for a plan $\pi = a_0, \ldots, a_i, \ldots, a_n$, and partially specified action, a_i, we create two plan fragments: $\pi^{F1} = a_0, \ldots, a_{i-1}$ and $\pi^{F2} = a_{i+1}, \ldots, a_n$. These fragments can then be used as input to the domain acquisition system instead of the complete plan descriptions.

Evaluation

In this section we present a case study examining the approach developed in this paper. This is split into two sections: the first examines the viability of acquiring suitable input sentences from users; the second examines whether operator sets can be synthesised that generalise user sentences and planning models induced.

Acquiring action descriptions

In order to obtain the action descriptions we asked naive users to explain animations that depicted action sequences from a collection of typical planning benchmark domains. The users were provided with recommendations for constructing the sentences and were provided guidance when their sentences did not meet these recommendations. In each case we noted the type of deviation that was made and we are therefore able to provide an indication of the areas where training is required, or opportunities for supporting similar approaches with inference. The specific descriptions made by each user still provide a wide variety of inputs.

We gathered sentences from 10 participants with a mixture of backgrounds and no experience of PDDL or related languages. In each session we collected 39 action descriptions between 3 domains. The session started by reading an introduction to the study and several recommendations for the sentences, including general properties of the sentences, e.g., that it was a stand-alone sentence without co-reference (not enforced). The key recommendations were: that the sentence should always include an explicit description of the starting situation of the main object before the action (e.g., pick up the blue block *from the red block*); and to use consistent referencing for objects; for example if you refer to an object as 'the red block' to always use this name. The participants were then shown an example video and sentence in the Blocksworld domain (Figure 3a). The participants described actions for animated videos in the following domains:

- **Towers of Hanoi:** The benchmark domain, visualised with cards and not pegs (Figure 3b). A card can only be placed on top of a card with a lower rank. The player must move the cards so that all 4 cards are piled on the right-most stack. We used the first 8 moves of a solution.
- **Logistics:** A standard logistics domain with misplaced packages that must be relocated using trucks (Figure 3c). The goal indicates the final configuration of the packages. The videos presented one small (4 step) and one longer (15 step) scenario.

	Ref. Variation	More info.	Action detection	Others
Hanoi	3	5	1	2
Logistics	0	7	3	4
Tyres	0	4	4	0

Table 1: The number of instances for each category of guidance required by participants during sentence collection (from a total of 390 action descriptions collected). The categories are: referential variation, more required information, no action detected in the sentence, and others (concentration and participant instigated querying of verb use).

```
Open the boot of the red car
Take out the jack from the boot of the red car
Put the jack underneath the front of the red car
Lift the front of the red car with the jack
Remove the front wheel of the red car
Replace the front wheel of the red car
Let down the front of the red car with the jack
Take the jack from underneath the front of the red car
Put the jack into the boot of the red car
Close the boot of the red car
```

Figure 4: An example of the input NL sentences (from participant p5) describing the actions in the Tyreworld scenario.

- **Tyreworld:** A subset of the benchmark domain. The scenario involves jacking a car up by opening the boot, removing the jack, raising the car and removing the wheel (Figure 3d), before reversing the process. An example of a participants sentences are presented in Figure 4.

Table 1 presents a coding of the guidance that was required during participant construction of input sentences. These can be largely divided into two areas: parsing the inputs and guidance on specific content of the inputs. The main parsing issue involved a failure of the parser to extract actions from the sentences. We were able to test the user sentences as they were constructed and therefore discover an alternative quickly. In these cases (e.g., 'load' or 'lower' at the beginning of a sentence) we asked the participant to consider using an alternative verb.

The main limitation observed in the user sentences was missing out one of the key actors in the action. It is perhaps unsurprising that in the first sentences for Logistics and Towers of Hanoi the participants often did not mention the starting location when describing moving a card, truck or package. Although it should be noted that the participants had been explicitly asked to do so using the Blocksworld example. Missing information was less of a problem in the Tyreworld domain. The most common omission was not mentioning the jack's role in lowering the car.

In Towers of Hanoi, some of the participants described the cards moving between columns, or an enumeration of the cards in the involved stacks. Use of alternative referencing encodings (referential variation) was not observed in other domains. Of course these alternative descriptions are valid and more importantly might be appropriate for alternative model acquisition target languages.

There were certain aspects of the participants' sentences

	p1	**p2**	**p3**	**p4**	**p5**	**p6**	**p7**	**p8**	**p9**	**p10**
Hanoi	0(1)	0(3)∗	0(2)∗	0(1)	0(1)∗	0(1)	0(2)∗	0(2)∗	0(4)∗	0(2)∗
Log.	0(3)	0(3)	1(4)∗	0(3)	0(3)	1(3)	0(3)	1(4)∗	0(3)	0(3)
Tyres	2(6)	2(6)∗	2(6)	0(5)	3(5)∗	1(5)	1(5)∗	1(5)∗	2(6)∗	1(5)∗

Table 2: Number of errors and clusters (in parenthesis) for the sentences for each participant (p1–p10) in the three domains. Each cell records the number of wrongly allocated sentences and the cluster count in parenthesis. The symbol ∗ indicates at least one k (not selected by silhouettes) partitions the sentences into distinct behaviours.

that were surprisingly good. There were relatively few typos and only one occurrence of entering the wrong event. There was only one occasion where a participant changed the way they were recording a behaviour during a scenario in such a way that the following sentence did not include enough information (from a lack of concentration). Although we had to request more information on several occasions, the participants typically continued to provide this in subsequent descriptions (within that scenario).

In a final step, we normalised user references, e.g., 'point A' and 'A' were both mapped to 'location A'. This was in order to assist with reference disambiguation and parsing, e.g., 'A' is a word and parsed differently from 'B' or 'C'.

In general, the main limitation of the sentences was specific missing information. Therefore, considering how this information can be recovered from alternative sources, including additional user input is key future work in extending the applicability of this approach.

Inducing planning models

In this part we take each participant's input separately and learn a domain model using the process as presented.

Identifying behaviour groups The first stage is clustering the sentences into individual behaviour groupings. Table 2 shows the number of errors in splitting the sentences into behaviours. In Logistics there are three main behaviours and the clustering approach typically divides the sentences accordingly. There were several cases (p6, p7, p8, p10) where the same verb was used for distinct behaviours, e.g., using 'moved' for driving, loading and unloading. However, the clustering was robust to this, although in some of these cases other causes impacted on the performance. This demonstrates the importance of using the roles as part of the similarity measure. In fact the only source of error in Logistics was inconsistency in the sentences. This happened both in verb use (p3) and different role identifiers (p6, p8, p10).

Whereas in Logistics there seem to be clear distinct behaviours, there is more ambiguity in the other domains. In the Towers of Hanoi examples there are 4 behaviours that can be distinguished: from empty, to empty; from empty, to card; from card, to empty; and from card, to card. In some cases the participants made consistent distinctions and some of these behaviours were isolated. In Table 3 we present the number of behaviours that were correctly isolated using different values of k in the clustering algorithm. In some cases, no distinction in language was made and only a single be-

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
1 cluster	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2 clusters	✗	✓	✓	✗	✓	✗	✓	✓	✓	✓
3 clusters	✗	✓	✓	✗	✗	✗	1	✗	✓	✓
4 clusters	✗	✓	1	✗	✗	✗	✗	✗	✓	1

Table 3: Tower of Hanoi results for participants (p1-p10). For the 4 possible behaviours (i.e. clusters $k = 1$ to max. 4.): ✓ indicates the participant correctly distinguished the behaviour; ✓ the number of clusters selected by the average silhouette score; '1' indicates a single change was required; and ✗ failure to distinguish behaviour (see text).

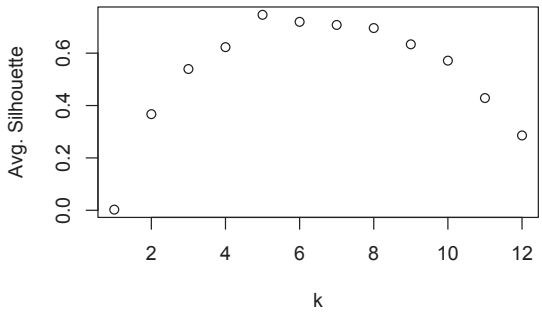


Figure 5: Plot of average silhouette scores for different values of k in Tyreworld (participant p5).

haviour was identified (e.g., p1, p4 and p6). The 'to empty' and 'empty to empty' behaviours were the most commonly distinguished.

The performance of the silhouette selection is not as effective in Tyreworld. The number of samples from each participant is small and this is particularly relevant in Tyreworld where there can be 10 different behaviours (depending on participant encoding). It is important to note that there are 7 out of 10 cases where the distance measure distinguished behaviours for some value of k. The silhouette plot presented in Figure 5 illustrates how close the silhouette scores were for p5 to a correct partitioning at k=8.

In Logistics and Hanoi, the γ (the bias between action name and roles in the distance measure) values: 0.33, 0.5 and 0.66, generated the same clusters. In Tyreworld there are small differences in the order the sentences break into separate clusters as k is increased. However, there is only one change in silhouette score (p2) and by $k = 8$ (approximately the number of behaviours) all clusters are the same.

In general over the 3 domains there are only 4 instances where there is not a valid partitioning for some value of k. This provides support that the selected distance measure is an effective approach for identifying behaviours. However, choosing amongst correct partitionings is still an interesting problem, as it can influence the generality of the induced model. However, pragmatically selecting one that has least missing values could be considered.

Formalising the behaviour representations In this part we assume that the behaviours have been correctly identified and split into different clusters (i.e., not necessarily the one

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
Hanoi	✓	R	1R	✓	✓	✓	✓	✓	R∗	R
Logistics	✓	✓	✓	✓	✓	✓	✓	1∗	✓	1
Tyres	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 4: Whether the correctly clustered sentences induced a model. ✓: a PDDL model was induced; n: some fixes were required and then a model was induced; $*$: a partial model was induced before fixes; R: a correct representation was constructed.

selected by silhouettes). A representative was selected for each cluster and then a mapping was made onto each member of the cluster to identify the best match for each of the (unfiltered) representative's roles. We set the parameter for tag filtering at $y = 20\%$.

As we have seen above (e.g., Table 3) in Towers of Hanoi, the participants varied in their chosen description strategy. When selecting a single cluster, the action headers for participants p1, p4, p5, p6, p7 and p8 are equivalent to the PDDL benchmark model. Participants: p2, p3, p9, p10, made distinctions between the different behaviours. In these cases the short action sequence that they were asked to describe was insufficient to provide enough examples for LOCM to induce a model properly. However, for participants, p2, p3 and p10, the final operator headers correctly described the actions. For p9, the rewriting rules used during parsing removed some of the important content and so the resulting operator headers, while correct for the information, did not contain all the important objects. In each case except 9, if additional sentences are added (we added 15 sentences) using a consistent method of description and the sentences are separated into four clusters then a PDDL similar to the 4-operator Blocksworld model is induced.

In Logistics the participants used predominantly consistent sentences within each of the three behaviour groups. Out of four inconsistencies, there were two cases (p8 and p10) that prevented the correct slot fillers to be identified. For example, one participant mixed 'onto' and 'into' and these words are not identified as synonyms by the selected sources. However, p3 changed their explanation of taking a package out of a truck from: 'The Parcel1 has been taken out of the red truck at location C' to 'The Parcel2 has been removed from the blue truck at location E'. The sources matched both 'taken' and 'removed' as synonyms, as well as the roles 'from' and 'out'. This highlights how the use of synonyms help to generalise over some of the variation in descriptions.

In Tyreworld the main structure is in sequential applicability of operators. One participant captured a more factored model, representing the jack moving out from the car boot to the ground, round the car and then to a position underneath the car (with its return journey). This provided a traversal structure, whereas most of the other descriptions used distinguishing language between the behaviours, e.g., putting the jack in and taking it out of the boot.

The learnt PDDL model Once consistent formulations of the input NL sentences had been extracted, the resulting action sequences were formatted, as action headers, and input

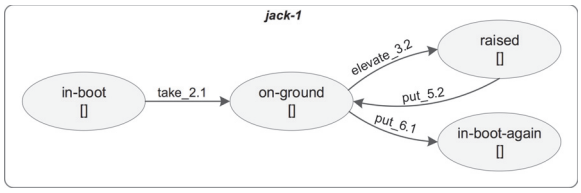


Figure 6: Induced FSM for a jack in Tyreworld. The states have been labelled for presentation.

to LOCM. LOCM generates a collection of FSMs that characterise the dynamics of the problem domain. For example, Figure 6 presents the structure identified for the jack object for a participant's sequences for Tyreworld. These FSMs can be used to induce a planning domain model (e.g., the Logistics domain for the sentences of p6 is presented in Figure 1) and in combination with an action sequence, to generate a problem model. For example, a planning problem was generated using p6's descriptions of the Logistics scenario (a larger example of sentences S1-S4) and Figure 7 presents a plan generated to solve that problem.

During processing, the mapping from the original sentence to the action template is retained and the parameter positions of the operator are identified in each sentence. This provides us with a template for rewriting a generated plan by fitting the arguments into the sentence. For example, in Figure 7, we present a plan generated using the learnt model. Using the template for the move_2 operator, plan step two can be rewritten as 'move the Parcel1 from location_B into the red_truck' (underscores retained for clarity). The development of this would be to modify the sentence using a language model, e.g., selecting determiners.

This system presents a step towards a general interface for exploiting planning technologies using only NL.

Related work

The majority of related work has aimed at mapping NL input onto an existing formal representation.

In RoboCup@Home various approaches have been adopted to define the mapping onto the grounded domain representation. For example Kollar et al. (2013) present a probabilistic approach to learning the referring expressions for robot primitives and physical locations in a region. And Mokhtari, Lopes, and Pinho (2016) present an approach to learning action schemata for high-level robot control.

In (Goldwasser and Roth 2011) the authors present an alternative approach to learning the dynamics of the world where the NL input provides a direct lesson about part of the dynamics of the environment. For example, the lesson: 'You can move any of the top cards to an empty free-cell' is a general rule that applies across several grounded situations. Each lesson is supported by a small training data set (e.g., 20 examples) to support learning from the lessons. In contrast to our approach, their system relies on a representation of the states and actions, which means their NLP approach can target an existing language.

More closely related to our work are attempts to learn planning models in the absence of a target representation.

```
drive_1 red_truck location_A location_B
move_2 Parcel1 location_B red_truck
drive_1 red_truck location_B location_E
move_3 Parcel1 red_truck location_E
drive_1 red_truck location_E location_C
move_2 Parcel2 location_C red_truck
drive_1 red_truck location_C location_A
move_3 Parcel2 red_truck location_A
move_2 Parcel1 location_E blue_truck
move_3 Parcel1 blue_truck location_E
drive_1 blue_truck location_E location_C
```

Figure 7: Logistics domain: plan generated from a larger example including sentences S1-S4 (participant p6).

These include (Sil and Yates 2011) who used text mining via a search engine to identify documents that contain words that represent target verbs or events and then uses inductive learning techniques to identify appropriate action pre- and post-conditions. Their system was able to learn action representations, although with certain restrictions such as the number of predicate arguments. Branavan et al. (2012) introduce a reinforcement learning approach which uses surface linguistic cues to learn pre-condition relation pairs from text for use during planning. The success of the learnt model relies on use of feedback automatically obtained from plan execution attempts. Yordanova (2016) presents an approach which works with input text solution plans, as a proxy for instructions, and aims to learn pre- and post-condition action representations. However this approach uses hand-coded representations of the initial and goal state for input plans.

Conclusion and future work

We believe this is the first approach that generates PDDL models directly from NL without an existing target model. Our approach harnesses a selection of existing technologies, including Standford CoreNLP, several online lexical resources, PAM, and LOCM. In our evaluation we demonstrated that the system can create formalisms from a variety of different NL representations. Although improving the robustness of the approach will be important future work, it should be noted that once a model is generated then it can be combined with the large body of existing work that looks at mapping NL onto existing formalisms. In the current approach, separating the sentences into appropriate behaviours plays an important role in determining the quality of the generated PDDL model. Important future work will explore generating models with various granularities and identifying whether they can be supported by the information content of the input sentences. Another avenue of future work is considering more intelligent ways of dealing with missing information. Our approach relies heavily on a sufficient number and length of fully specified sequences in the input. An interesting approach would be to use a predictive model to estimate the parameter selections; perhaps taking inspiration from the recommendation system approach presented in (Krivic et al. 2016) for predicting initial world object properties. Alternatively, we could target other domain acquisition systems, such as (Mourão, Petrick, and Steedman 2010) that handle noisy data.

Acknowledgements

This work is supported by EPSRC Grant EP/N017447/1.

References

Branavan, S. R. K.; Kushman, N.; Lei, T.; and Barzilay, R. 2012. Learning High-level Planning from Text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, ACL '12, 126–135. Stroudsburg, PA, USA: Association for Computational Linguistics.

Cresswell, S., and Gregory, P. 2011. Generalised domain model acquisition from action traces. In *Proc. of the 21st Int. Conf. on Automated Planning and Scheduling (ICAPS)*.

Cresswell, S. N.; McCluskey, T. L.; and West, M. M. 2009. Acquisition of Object-Centred Domain Models from Planning Examples. In *Proc. of 19th Int. Conf. on Automated Planning and Scheduling (ICAPS)*.

de Marneffe, M.-C.; MacCartney, B.; and Manning, C. D. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Conference on Language Resources and Evaluation*, 449–454.

Duda, R. O.; Hart, P. E.; et al. 1973. *Pattern classification and scene analysis*, volume 3. Wiley New York.

Ersen, M., and Sariel, S. 2015. Learning behaviors of and interactions among objects through spatio-temporal reasoning. *Computational Intelligence and AI in Games, IEEE Transactions on* 7(1):75–87.

Frank, J. D.; Clement, B. J.; Chachere, J. M.; Smith, T. B.; and Swanson, K. J. 2011. The Challenge of Configuring Model-Based Space Mission Planners. In *International Workshop on Planning and Scheduling for Space*.

Goldwasser, D., and Roth, D. 2011. Learning from natural instructions. In *Proc. of the 22nd Int. Joint Conf. on Artifical Intelligence (IJCAI)*.

Gregory, P., and Cresswell, S. 2015. Domain Model Acquisition in the Presence of Static Relations in the LOP System. In *Proc. of 25th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 97–105.

Gregory, P., and Lindsay, A. 2016. Domain Model Acquisition in Domains with Action Costs. In *Proc. of the 26th Int. Conf. on Automated Planning and Scheduling (ICAPS)*.

Hayton, T.; Gregory, P.; Lindsay, A.; and Porteous, J. 2016. Best-fit action-cost domain model acquisition and its application to authorship in interactive narrative. In *AAAI Conf. on AI and Interactive Digital Entertainment (AIIDE)*.

Hoffmann, J.; Weber, I.; and Kraft, F. M. 2012. SAP speaks PDDL: Exploiting a software-engineering model for planning in business process management. *Journal of Artificial Intelligence Research* 44:587–632.

Kaufmann, L., and Rousseeuw, P. J. 1987. Clustering by means of medoids. *Journal of Machine Learning Research*.

Klein, D., and Manning, C. D. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems*, volume 15. 3–10.

Kollar, T.; Perera, V.; Nardi, D.; ; and Veloso, M. 2013. Learning Environmental Knowledge from Task-based Human-robot Dialog. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.

Krivic, S.; Cashmore, M.; Ridder, B.; and Piater, J. 2016. Initial State Prediction in Planning. In *Proc. 31st Workshop of the UK Planning and Scheduling SIG (PlanSIG)*.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics and Control Theory* 10:707–710.

Lindsay, A.; Charles, F.; Read, J.; Porteous, J.; Cavazza, M.; and Georg, G. 2015. Generation of non-compliant behaviour in virtual medical narratives. In *Proc. of the 15th International Conference on Intelligent Virtual Agents (IVA)*.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J. R.; Bethard, S.; and McClosky, D. 2014. The stanford corenlp natural language processing toolkit. In *The Annual Meeting of the Association for Computational Linguistics (System Demonstrations)*, 55–60.

McCluskey, T. L.; Cresswell, S. N.; Richardson, N. E.; and West, M. M. 2009. Automated acquisition of action knowledge. In *International Conference on Agents and Artificial Intelligence (ICAART)*, 93–100.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL-the planning domain definition language. Technical report, Yale University.

Mehta, N.; Tadepalli, P.; and Fern, A. 2011. Efficient Learning of Action Models for Planning. In *ICAPS Planning and Learning Workshop (PAL),*.

Mokhtari, V.; Lopes, L. S.; and Pinho, A. J. 2016. Experience-Based Robot Task Learning and Planning with Goal Inference. In *Proc. of the 26th International Conference on Automated Planning and Scheduling (ICAPS)*.

Mourão, K.; Petrick, R. P. A.; and Steedman, M. 2010. Learning action effects in partially observable domains. In *Proc. 19th European Conference on AI (ECAI)*. IOS Press.

Porteous, J.; Lindsay, A.; Read, J.; Truran, M.; and Cavazza, M. 2015. Automated extension of narrative planning domains with antonymic operators. In *Proc. of the Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*.

Richardson, N. E. 2008. *An Operator Induction Tool Supporting Knowledge Engineering in Planning*. Ph.D. Dissertation, School of Computing and Engineering, University of Huddersfield, UK.

Rousseeuw, P. J. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20:53–65.

Sil, A., and Yates, A. 2011. Extracting strips representations of actions and events. In *Recent Advances in Natural Language Processing (RANLP)*.

Walsh, T. J., and Littman, M. L. 2008. Efficient Learning of Action Schemas and Web-Service Descriptions. In *Proc. of 23rd AAAI Conference on Artificial Intelligence*.

Yordanova, K. 2016. From Textual Instructions to Sensor-based Recognition of User Behaviour. In *Proc. of 21st Int. Conf. on Intelligent User Interfaces*, IUI Companion. ACM.