Abstract
This paper describes a methodology for designing Question Answering systems that utilize an action language $ALM$ to allow inferences based on complex interactions of events described in texts. This methodology assumes the extension of the VERBNET lexicon with interpretable semantic annotations in $ALM$ and specifies the use of several other NLP resources to produce $ALM$ system descriptions for input discourses.

1 Introduction
We propose a methodology for designing Question Answering (QA) systems that uses state-of-the-art techniques from the field of Natural Language Processing (NLP) complementing them with the latest advances from the field of Knowledge Representation and Reasoning (KRR).

The applicability of KRR for the design and implementation of QA systems was explored by Baral et al. (2004) who demonstrated the suitability of the KRR language Answer Set Prolog (ASP) (Gelfond and Lifschitz, 1991) for this purpose. Balduccini et al. (2008) continued this line of research and described a QA system with an intended wide coverage, based on KRR techniques and NLP tools. Todorova and Gelfond (2011; 2012) addressed the problem from a different angle. They focused on texts restricted to a controlled natural language tailored to motion verbs and concentrated on answering difficult questions requiring counting. Their knowledge base was written in a higher-level KRR language than ASP, a so-called action language called $ALM$ (Inclezan and Gelfond, 2016).

The system by Todorova and Gelfond processed multiple-sentence texts exemplified by

\begin{align*}
\text{Ann went to the room.} & \quad (1) \\
\text{Michael left the room.} & \quad (2)
\end{align*}

and derived inferences based on information in these sentences to answer questions such as

\begin{align*}
\text{Is Michael inside the room (at the end of the story)?} & \quad (3) \\
\text{Is Ann inside the room (at the end of the story)?} & \quad (4) \\
\text{Is the room empty (at the end of the story)?} & \quad (5)
\end{align*}

In the sequel, we refer to the text composed of sentences (1) and (2) as the $MA$ discourse.

Our goal is to build upon the methodology outlined by Todorova and Gelfond by putting more emphasis on the organization of KRR libraries and the NLP stages of QA. We remove some of the constraints assumed by their system: we do not limit ourselves to the motion domain and we avoid a commitment to a controlled language. Our long term goal is to have a methodology that encompasses texts containing a large collection of action verbs (e.g., go, give, put). In this paper, we test the feasibility of such a proposal by allowing change of possession verbs such as grab, grasp, yank in addition to motion verbs. Consider a discourse that contains the sentence

\text{Michael grasped the suitcase.} \quad (6)
uttered between sentences (1) and (2). We name it the MAS discourse. We illustrate that a system developed according to our methodology is able to infer that at the time when *Michael* was grasping the *suitcase*, its location was the *room* — the location of *Michael*, and that the *suitcase* is no longer in the *room* at the end of the described scenario.

The main feature of our approach is the use of *ALM* as a language for encoding the meaning of action verbs (i.e., the effects and constraints for the execution of the actions they denote). In addition, we propose to extend an NLP resource about verbs, VERBNET (Kipper-Schuler, 2005; Palmer, 2006), with *ALM* based semantic annotations.

The paper is structured as follows. Section 2 starts by introducing the KRR action language *ALM* by illustrating how a knowledge engineer can utilize this language to formalize the scenarios described by the *MA* and *MAS* discourses and query the respective formalization. In the process, generic modules that capture the knowledge about such actions/verbs as *move*, *go*, and *grasp* are developed. Section 3 focuses on the methods that will allow us to automatically produce *ALM* descriptions that capture information present in discourses of interest by utilizing the arsenal of modern NLP lexicons and tools including VERBNET, PROPANK (Palmer et al., 2005; Palmer, 2005), SEMLINK (Bonial et al., 2013b,a), Ontonotes Sense Groupings (CLEAR, 2008), LTH (Johansson and Nugues, 2007b,a), and CORENLP (Manning et al., 2014). We end with conclusions and future work.

## 2 The MA and MAS discourses formalized in *ALM*

Action language *ALM* is a recent representative of KRR languages for modeling knowledge about domains in which changes are caused by the occurrence of actions. An important feature of *ALM* is its ability to capture the commonality of actions *go* and *leave* by defining them as instances of the same action class that we refer to as *MOVE*, and thus encode the relation that exists in natural language between the corresponding verbs. Other logic formalisms have been used for other NLP tasks (e.g., Recognizing Textual Entailment (NIST, 2008)), but unlike *ALM* they cannot perform temporal reasoning (MacCartney and Manning, 2007; Harmeling, 2009) or reasoning by cases (Bos and Markert, 2005). This makes them less suitable for answering questions about discourses describing sequences of events.

We start by using *ALM* to formalize the domain behind the *MA* discourse. First, we use this example to illustrate the syntax and semantics of the language. Second, we demonstrate how the *ALM* framework can be used to perform inferences required to answer questions (3-5). The section concludes with the *ALM* formalization of the *MAS* discourse.

**MA discourse via *ALM***: There are several informative pieces in the *MA* discourse:

1. the discourse refers to actions of class *MOVE* through the use of verbs *go* and *leave*. This action class immediately brings about a set of axioms associated with it. For example, we are aware that it is impossible to move an object from a point if this object is not at this point.
2. three objects (entities, or instances) are introduced, to which we refer as *ann*, *michael*, and *room*; and two events (instances of actions): *ann* moves into *room* and *michael* moves out of *room*.
3. a sequence of event occurrences is given, i.e., *first* *ann* moves into *room* and *next* *michael* moves out of *room*.

**Informative piece 1 or *ALM* module basic_motion for modeling the MOVE action class**: Figure 1 (LHS) shows the *ALM* module called *basic_motion* that can be seen as a general purpose description of knowledge/axioms about the *MOVE* action class. This module describes how the location of objects is affected by occurrences of events of type *MOVE*.

Modules in *ALM* start with the declaration of *sorts* of objects relevant to the knowledge to be encoded. In *basic_motion*, we distinguish between *things* and discrete *points* in space. We declare these two sorts as special cases of the pre-defined root sort of *ALM* called *universe*. We also declare a sort called *agents*, denoting entities capable to move by themselves, as a subsort of *things*. The knowledge engineer then proceeds to specify the relevant action classes for the domain in question. In module *basic_motion*, action class *move* is declared as a special case of the pre-defined
sort actions with three attributes (i.e., intrinsic properties): attribute actor ranging over the sort agents, and attributes origin and dest (destination) ranging over points.

Next, properties (fluents and statics) related to the domain are declared. Fluents are properties that may be changed by actions; they are divided in ALM into basic and defined. Basic fluents normally maintain their previous values, unless the occurrence of an event causes their value to change. Defined fluents allow one to specify properties in terms of other properties. Properties are modeled via functions in ALM using syntax similar to the mathematical notation for functions. In the basic motion module, the property of interest is the location loc in of things, which may be affected by the occurrence of actions of type move and is thus declared as a basic fluent. Per its specification, it is a function that maps pairs of things and points into the pre-defined sort booleans.

ALM modules conclude with axioms about described action classes and properties. The first two rules in the basic motion module capture the direct effects of actions of sort move. In particular, the first axiom states that after an occurrence of an instance of move its actor will be located at the destination. The second axiom states that after an occurrence of a move event the actor will no longer be at the origin. The last two statements describe when the action cannot be executed. In particular, the third and fourth axioms state that move cannot occur when the actor is not located at the specified origin, and when the actor is already at the destination, respectively. A slightly different version of this module can be found in (Inclezan and Gelfond, 2016), where it was used to illustrate the syntax of ALM.

Informative piece 2 or ALM system description for the MA discourse: In ALM, we describe a given domain via a system description that consists of a theory — modules organized into a hierarchy, and a structure — definitions of instances. A system description captures a transition diagram that characterizes the behavior of the given domain. Trajectories in a transition diagram correspond to possible evolutions or scenarios in the domain. We illustrate these concepts by means of the discourse_ma system description presented in Figure 2 (LHS), which corresponds to the MA discourse.

The discourse_ma theory consists of the line import module basic_motion that can be interpreted as a macro to denote that the ALM code describing “module basic_motion” has to be inserted.
LHS: \text{ALM} system description capturing parts of the MA discourse using module \text{basic\_motion}; RHS: the same system description restated using \text{basic\_motion\_verbnet}.

Figure 3: Transition diagram captured by the system description \text{discourse\_ma}.

in this place. The structure of \text{discourse\_ma} declares instances \texttt{ann}, \texttt{michael}, and \texttt{room} so that the former two are of sort \texttt{agents}, whereas the latter is of sort \texttt{points}. The two events described in the MA discourse are represented as instances \texttt{e1} and \texttt{e2} of sort \texttt{move}. The \texttt{actor} of \texttt{e1} is instance \texttt{ann} and its \texttt{dest} is \texttt{room}, while the \texttt{actor} of \texttt{e2} is \texttt{michael} and its \texttt{origin} is \texttt{room}.

The system description \text{discourse\_ma} defines the transition diagram $T$ presented in Figure 3. It consists of four states labeled $\sigma_1 \cdots \sigma_4$, and five transitions labeled by actions $e_1, e_2$ that may take the dynamic system from one state to another. For example, the arc $e_1$ between states $\sigma_2$ and $\sigma_1$ says that the occurrence of $e_1$ may take the system from the former state to the latter. Note how action $e_1$ cannot occur in state $\sigma_1$ (due to the last axiom in \text{basic\_motion}) and thus there is no arc in $T$ going out of $\sigma_1$ and labeled $e_1$. The initial state of a system is associated with time step 0. Each arc in the transition diagram suggests an increment of a time step by one. A sequence $\tau_1 = \langle \sigma_2, e_1, \sigma_1, e_2, \sigma_3 \rangle$ constitutes a sample trajectory. This trajectory captures the following scenario: initially (time step 0), \texttt{ann} is not in \texttt{room}, whereas \texttt{michael} is in \texttt{room}; at time step 0, \texttt{ann} moves to (enters) \texttt{room}; at time step 1, \texttt{michael} moves from (leaves) \texttt{room}. A sequence $\tau_2 = \langle \sigma_2, e_2, \sigma_4, e_1, \sigma_3 \rangle$ exemplifies another trajectory of transition diagram $T$, whereas a sequence $\langle \sigma_1, e_1, e_2, \sigma_4 \rangle$ is not a trajectory.

\textbf{Informative Piece 3 or ALM histories:} A particular domain scenario defines what we call a \textit{history} (Gelfond and Khal, 2014) — a set of observations about fluents that hold in some states and events that happen at some states. Certain trajectories in the transition diagram encoded by a system description are compatible with a particular history, while others are not. We call the compatible trajectories \textit{models} of a history. The MA discourse provides the following history: action $e_1$ happens first (at time step 0), then $e_2$ happens (at time step 1), which we abbreviate below

$$\{\text{hpd}(e_1, 0), \text{hpd}(e_2, 1)\}. \quad (7)$$

Given system description \text{discourse\_ma}, trajectory $\tau_1$ is the only model of this history. Thus the initial state of the story conveyed by the MA discourse must be $\sigma_2$ and the final one must be $\sigma_3$.

\textbf{Answering questions (3-5):} Model $\tau_1$ of history (7) allows us to answer questions (3-5). The final state $\sigma_3$ of the trajectory $\tau_1$ contains literal $\neg\text{loc\_in(michael, room)}$, which translates into the answer

$$\{\text{hpd}(e_1, 0), \text{hpd}(e_2, 1)\}. \quad (7)$$
module grasping depends on basic_motion
sort declarations
  carriables :: things
  grasp :: actions
attributes
  grasper : agents
  grasped_thing : carriables
function declarations
fluents
  basic
    holding : agents * carriables -> booleans
    can_reach : agents * things -> booleans
defined
  occurs(X) causes holding(A,C) if instance(X,grasp),
  grasper(X)=A,
  grasped_thing(X)=C.
  can_reach(A,C) if loc_in(A)=P, loc_in(C)=P.
  impossible occurs(X) if instance(X,grasp),
  grasper(X)=A,
  grasped_thing(X)=C,
  holding(A,C).
  impossible occurs(X) if instance(X,grasp),
  grasper(X)=A,
  grasped_thing(X)=C,
  -can_reach(A,C).
  loc_in(X,P) if loc_in(C,P), holding(X,C).
  loc_in(C,P) if loc_in(X,P), holding(X,C).
  -loc_in(X,P) if -loc_in(C,P), holding(X,C).
  -loc_in(C,P) if -loc_in(X,P), holding(X,C).

Figure 4: (a) ALM module defining action grasp; (b) System description for the MAS discourse.

do to question (3). State σ₃ contains loc_in(ann, room) that translates into answer yes to question (4). Presence of loc_in(ann, room) in state σ₃ translates into answer no to question (5).

Similarly, we can answer other questions: Is Michael inside the room at the beginning of the story? Is Ann inside the room at the beginning of the story? Were Ann and Michael in the room together at some point? How many people were in the room when Ann walked in? The initial state σ₂ of model τ₁ supports the answers yes and no to the first and the second questions, respectively. The positive answer to the third question is endorsed by the intermediate state σ₁ of τ₁. Initial state σ₂ encodes the situation preceding Ann walking into the room. It supports the answer at least one to the last question.

Automatically computing models of a history: Given the ALM system description and the history that correspond to a discourse in question, the task of computing models of this history relative to the system description can be automated. First, the system description is translated into a logic program under answer set semantics using the transformation defined by Inclezan and Gelfond (2016). Second, the history and a predefined module for temporal projection (Gelfond and Khal, 2014) are added to the produced logic program. Answer sets of the resulting program can be computed using an off-the-shelf ASP solver CLINGO available at http://www.mbal.tk/clingof/. Each answer set corresponds to a model of the given history. A prototype translator from ALM system descriptions and histories to logic programs is available at http://tinyurl.com/z6n9fmx.

MAS discourse via ALM: In order to model the MAS discourse, an ALM module that formalizes knowledge about actions of type GRASP is required. Figure 4 (a) presents a module called grasping that serves this purpose. It is adapted from (Inclezan and Gelfond, 2016), where it was used to illustrate the methodology of creating modular representations in ALM by encoding a classical Monkey and Bananas problem from the field of reasoning about actions and change. The first line of module grasping suggests that this module reuses sorts and/or functions explicitly declared in module basic_motion. Specifically, it reuses the fluent loc_in as the location of agents and things conditions what GRASP actions can be executed.

ALM system description for the MAS discourse: The system description for the MAS discourse,
discourse_mas, is presented in Figure 4 (b). Its theory starts with an import statement for module grasping. Given that grasping depends on module basic_motion, the meaning of this $\text{ALM}$ statement is that contents of both modules are copied into the theory of discourse_mas. Hence, within this system description we can instantiate events that are of type grasp or move. The fact that action classes grasp and move are interconnected in the definition of module grasping allows a knowledge engineer to model nontrivial interdependencies between actions. For example, an instance of action grasp causes its agent to hold a grasped object. Module grasping also encodes the knowledge that if an agent holds an object then the locations of the agent and object must be the same. These restrictions allow one to deduce that, when an instance of an action move occurs while the agent holds some object, then this object changes its location just as the agent does. The structure of the discourse_mas system description is defined similarly to that of discourse_ma.

History \{hpd(e1, 0), hpd(ea, 1), hpd(e2, 2)\} records the events described in discourse MAS. In all models of this history relative to discourse_mas (i) the location of entity suitcase is the same as that of entity michael (namely, entity room) before action instance e2 occurs; (ii) after event ea occurs michael is holding suitcase in all subsequent states; and (iii) after event e2 occurs michael is holding suitcase, and both michael and suitcase are not in room. All of these observations correspond to our expectations given the MAS discourse. Indeed, we infer that the suitcase is no longer in the room at the end of the story. Similarly, when Michael grasped the suitcase, its location was the same as the location of Michael, i.e., the room.

3 Automatic construction of $\text{ALM}$ system descriptions from discourses

In the previous section we illustrated how a knowledge engineer may encode the information carried within the MA and MAS discourses in $\text{ALM}$. We then discussed how these $\text{ALM}$ formalizations can be used to automatically reason about these discourses. In this section, we present a proposal for automating the process of creating an $\text{ALM}$ system description for an English discourse by relying on modern NLP tools such as LTH, CORENLP and existing lexical resources including Ontonotes Sense Groupings, VERBNET, PROPBANK, and SEMLINK. We stress the steps that have to be performed and how NLP tools and resources are to be used in those steps. The MA discourse is a running example in this section.

Stage 1 or Entity and relation extraction: The goal of this stage is to take an English discourse as an input and produce a so-called discourse representation structure (DRS) — a basic building block of Discourse Representation Theory (Kamp and Reyle, 1993). Figure 5 presents a DRS for the MA discourse. The top part of this DRS enumerates all of the entities, called discourse referents, that take part in the captured discourse (namely, $r_1$, $r_2$, and $r_3$) as well as referents denoting events that the discourse describes (namely, $e_1$ and $e_2$). The bottom part of the DRS captures conditions on the entities and events that follow from the discourse. The events are encoded in Neo-Davidsonian style.

<table>
<thead>
<tr>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
<th>$e_1$</th>
<th>$e_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>entity($r_1$)</td>
<td>entity($r_2$)</td>
<td>entity($r_3$)</td>
<td>property($r_1$, ann)</td>
<td>property($r_2$, room)</td>
</tr>
<tr>
<td>event($e_1$)</td>
<td>event($e_2$)</td>
<td>eventType($e_1$, go.01)</td>
<td>eventTime($e_1$, 0)</td>
<td>eventArgument($e_1$, a1, r1)</td>
</tr>
<tr>
<td>eventType($e_2$, leave.01)</td>
<td>eventTime($e_2$, 1)</td>
<td>eventArgument($e_2$, a0, r3)</td>
<td>eventArgument($e_2$, a2, r1)</td>
<td>eventArgument($e_2$, a0, r3)</td>
</tr>
</tbody>
</table>

Figure 5: Discourse representation structure for the MA discourse

To produce a DRS as exemplified, the first proposed step is to process discourse sentences using the LTH semantic role labeler. For sentences (1) and (2) of the MA discourse, LTH produces the output:

[A1 Ann] [V (go.01) went] [A4 to the room]
[A0 Michael] [V (leave.01) left] [A1 the room]
The examples above are annotated using the rolesets/labels of the predicates go.01 and leave.01 as defined in frame schemas of PropBank (version 1.7), where the suffix 01 indicates that Ontonotes Sense Groupings associates these predicates with senses 1 of verbs go and leave:

- **go.01**: motion  
  - A1: entity in motion/goer  
  - A2: extend  
  - A3: start point  
  - A4: end point  
  - AM-LOC: medium  
  - AM-DIR: direction (usually up or down)

- **leave.01**: move away from  
  - A0: entity leaving  
  - A1: place left  
  - A3: attribute/secondary predication

In a second step, we propose to process a given discourse using the Stanford CoreNLP system. Among other NLP tasks, the CoreNLP system can perform mention detection and coreference resolution. Given the MA discourse it is able to detect that there are three entities in the discourse: Michael, Ann, and the room, and that expressions the room in sentences (1) and (2) refer to the same entity.

In a third step, the output of systems LTH and CoreNLP is combined to produce a DRS for the given input. Based on the output of CoreNLP, entities r1, r2, and r3 that have a property of being ann, room, and michael, respectively, are added to the DRS. Similarly, events e1 and e2 are known to be of type go.01 and leave.01, respectively, based on the output of LTH. Relation eventArgument is populated by using the role labels assigned by LTH. The time step for the events (encoded by eventTime) is provided based on chronological order of events mentioned in the discourse, which coincides with default readings of sentences (we disregard for now markers such as before, after).

**Related NLP systems**: System BOXER (Bos, 2008) is an open-domain NLP tool that, given a discourse constructs a respective DRS. However, the discourse representation structures constructed by BOXER omit ordering of events in the discourse (i.e., contain no counterpart to "eventTime" in Figure 5), as well as details on the roles played by event arguments. Also, named entity recognition and coreference resolution components of CoreNLP perform better than BOXER.

**Stage 2 or From discourse to an ALM system description or via PropBank to VerbNet to ALM**: The next question that we tackle is how to map entities, properties, and history present in a given DRS into the vocabulary of ALM modules that capture axioms about the actions denoted by verbs occurring in this DRS? For the MA discourse, this question translates into how do we transition from the DRS in Figure 5 to an ALM scenario composed of a system description in Figure 2 and history (7)?

In order to produce a system description and a history from a DRS, first we have to link two distinct PropBank predicates go.01 and leave.01 to the same ALM action class MOVE. Second, we ought to map the semantic roles prescribed by PropBank for these predicates to the arguments of action class MOVE as prescribed by the basic motion module. Third, we have to link the entities mentioned in the given DRS in Figure 5 with the instances that compose the structure of the system description capturing this discourse. We will illustrate how these steps result in an ALM system description discourse_ma_verbnet presented in Figure 2 (RHS). It is easy to see that to a large extent the new system description is a syntactic modification of the discourse_ma system description in Figure 2 (LHS) that has been designed earlier to process the MA discourse. The discourse_ma_verbnet system description can be used in the same manner to answer questions about this discourse. Next, we present details on components required to automate the construction of this system description.

**VerbNet Lexicon**: We start by focusing on the first two of the described tasks: mapping PropBank predicates go.01 and leave.01 and their arguments into instances of the ALM action class MOVE and its attributes. We argue that it is possible to carry out such mappings in a systematic manner using theories developed by linguists pertaining to verb semantics. Levin (1993) proposed the grouping of verbs into classes based on their syntactico-semantic behavior in sentences. Verb lexicon VerbNet is organized into verb classes that extend and refine these by Levin. For instance, VerbNet class escape-51.1 contains among others verbs go and return. A direct subclass of escape-51.1 named escape-51.1-1 contains verb leave. Any subclass of a class in VerbNet inherits all of the features of its parent class, but also contains its specific entries. In addition to capturing the grouping information of the verbs, VerbNet provides the ontology of core thematic roles associated with each group.
Four (thematic) roles are identified with the classes ESCAPE-51.1 and 51.1-1: theme, initial location, destination, and trajectory. Condition concrete represents (selectional) restriction that the arguments of the verbs of this class should satisfy to form semantically coherent sentences. Intuitively, in sentence (1) an entity corresponding to Ann serves the role theme, while an entity corresponding to the room serves the role destination. Both of these entities are of concrete kind/sort. Kipper-Schuler, Section 3.1.4 (2005) describes semantic annotations provided within VerbNet for each class. However, unlike ALM descriptions, these do not have formal semantics and are not computer interpretable in prescribed manner.

The ALM declaration of action class move in Figure 1 (LHS) echoes the information present in VerbNet. We see how attribute actor of move is declared of sort agents, whereas origin and dest are declared of sort points. Intuitively, attribute names such as actor, origin, and dest serve the role of thematic roles theme, initial location, and destination, respectively. Sorts agents and points echo selectional restrictions and are designated to be of concrete kind by VerbNet. The ALM modules are meant to encode common sense knowledge about actions and thus the attribute names and sorts corresponding to these attributes are fine-grained and attempt to capture peculiarities of respective actions. Figure 1 (RHS) presents the restatement of the basic motion module in (LHS) of the same figure using VerbNet terminology and named basic_motion_verbnet. It differs from (LHS) by different name choices. The only non-syntactic change appears in sort declarations, where the (RHS) module defines a less specific sort hierarchy. We envision an extended VerbNet that is augmented with ALM modules (such as basic_motion_verbnet), which provide ALM-based semantic annotations for its verb classes. Then, the VerbNet lexicon can serve as a lookup table for finding relevant action classes and ALM modules while processing discourses. We believe that the creation of an extended VerbNet will be an important contribution to both the NLP and KRR communities.

The semlink project: The last step to address is how to translate information about entities and events in a DRS into the ALM system description capturing a given discourse. Here, the missing piece of the puzzle is the semlink project (Bonial et al., 2013b) that links together PropBank and VerbNet.

For instance, semlink contains an entry suggesting that (i) the predicate leave.01 is part of verb class 51.1-1 (a child of the class ESCAPE-15.1) (ii) the argument A0 of predicate leave.01 is mapped to role theme of verb class 51.1-1, and (iii) the argument A1 of predicate leave.01 is mapped to role initial location of class 51.1-1. These mappings are sufficient for devising a translation from information in the DRS in Figure 5 about event e2 into the respective part of the structure of the ALM system description present in Figure 2 (RHS). Thus an event of type leave.01 can be seen as an event of type ESCAPE-51.1 and in turn as an instance of action move, which is captured by escape in the basic_motion_verbnet module presented in Figure 1 (RHS). Note that this also implies that module basic_motion_verbnet should be imported into the theory of this constructed system description. Similarly, semlink contains a mapping for predicate go.01 of PropBank to a respective class in VerbNet. Yet, argument A4 of go.01 and role destination in VerbNet is missing in this mapping. Thus, Semlink has to be augmented to accommodate the mapping from A4 to destination. Nevertheless, Semlink provides a solid foundation for the PropBank-VerbNet connection.

4 Conclusions and Future Work

We proposed a methodology for building a QA system that uses KRR techniques related to the representation of actions. We focused on answering questions that require the specification of knowledge about actions. We argued that annotating the verb lexicon VerbNet with such knowledge specifications in the KRR language of ALM will allow us to utilize a variety of NLP tools. We showed that the use of multiple NLP resources provides us with the means to extract information from an input discourse sufficient to “populate” a respective ALM system description and history that in turn can be used to draw nontrivial inferences about the discourse in question. In the immediate future, we will evaluate our method on a collection of texts from project bAbI (Weston et al., 2016; Facebook Research, 2016) containing motion and change of possession verbs. In a long term, we plan to expand the VerbNet annotations to include other types of English verbs.
References


