Action Languages and Question Answering

Yuliya Lierler
University of Nebraska Omaha
ylierler@unomaha.edu

Daniela Inclezan
Miami University
inclezd@miamioh.edu

Michael Gelfond
Texas Tech University
michael.gelfond@ttu.edu

Abstract

This paper describes a methodology for designing Question Answering systems that utilize an action language $\mathcal{ALM}$ 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 $\mathcal{ALM}$ and specifies the use of several other NLP resources to produce $\mathcal{ALM}$ 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 $\mathcal{ALM}$ (Inclezan and Gelfond, 2016).

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

Ann went to the room. (1)

Michael left the room. (2)

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

Is Michael inside the room (at the end of the story)? (3)

Is Ann inside the room (at the end of the story)? (4)

Is the room empty (at the end of the story)? (5)

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

Michael grasped the suitcase. (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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ based semantic annotations.

The paper is structured as follows. Section 2 starts by introducing the KRR action language $\mathcal{ALM}$ 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 $\mathcal{ALM}$ descriptions that capture information present in discourses of interest by utilizing the arsenal of modern NLP lexicons and tools including VERBNET, PROPBANK (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 $\mathcal{ALM}$

Action language $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ framework can be used to perform inferences required to answer questions (3-5). The section concludes with the $\mathcal{ALM}$ formalization of the *MAS* discourse.

***MA* discourse via $\mathcal{ALM}$**: 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 $\mathcal{ALM}$ module `basic_motion` for modeling the MOVE action class: Figure 1 (LHS) shows the $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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

```
(LHS)
module basic_motion
 sort declarations
  things, points :: universe
  agents :: things
  move :: actions
   attributes
    actor : agents
    origin, dest : points
function declarations
 fluents
  basic
   loc_in : things * points -> booleans
axioms
 occurs(X) causes loc_in(A,D)
   if instance(X,move),
      actor(X)=A,
      dest(X)=D.
 occurs(X) causes -loc_in(A,O)
   if instance(X,move),
      actor(X)=A,
      origin(X)=O.
 impossible occurs(X) if instance(X,move),
                         actor(X)=A,
                         origin(X)=O,
                         -loc_in(A,O).
 impossible occurs(X) if instance(X,move),
                         actor(X)=A,
                         dest(X)=D,
                         loc_in(A,D).
```

```
(RHS)
module basic_motion_verbnet
 sort declarations
  concrete :: universe

  escape :: actions
   attributes
    theme : concrete
    initial_location, destination : concrete
function declarations
 fluents
  basic
   loc_in : concrete * concrete -> booleans
axioms
 occurs(X) causes loc_in(A,D)
   if instance(X,escape),
      theme(X)=A,
      destination(X)=D.
 occurs(X) causes -loc_in(A, O)
   if instance(X,escape),
      theme(X)=A,
      initial_location(X)=O.
 impossible occurs(X) if instance(X, escape),
                         theme(X)=A,
                         initial_location(X)=O,
                         -loc_in(A,O).
 impossible occurs(X) if instance(X,escape),
                         theme(X)=A,
                         destination(X)=D,
                         loc_in(A,D).
```

Figure 1: LHS: $\mathcal{ALM}$ module `basic_motion` capturing knowledge about action class MOVE; RHS: the same module restated using the VERBNET lexicon terminology.

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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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`.

$\mathcal{ALM}$ 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 $\mathcal{ALM}$.

Informative piece 2 or $\mathcal{ALM}$ system description for the *MA* discourse: In $\mathcal{ALM}$, 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 $\mathcal{ALM}$ code describing "module basic_motion" has to be inserted

```
(LHS)
system description discourse_ma
  theory discourse_ma
    import module basic_motion


  structure discourse_ma
    instances
     ann  in agents
     michael in agents
     room in points
     e1 in move
       actor = ann
       dest = room
     e2 in move
       actor = michael
       origin = room
```

```
(RHS)
system description discourse_ma_verbnet
  theory discourse_ma_verbnet
    import module basic_motion_verbnet
    module drs_michael_and_ann
      ann, michael, room :: universe
  structure discourse_ma_verbnet
    instances
      r1 in ann, concrete
      r2 in room, concrete
      r3 in michael, concrete
      e1 in escape
        theme = r1
        destination = r2
      e2 in escape
        theme = r3
        initial_location = r2
```

Figure 2: LHS: $\mathcal{ALM}$ system description capturing parts of the *MA* discourse using module `basic_motion`; RHS: the same system description restated using `basic_motion_verbnet`.

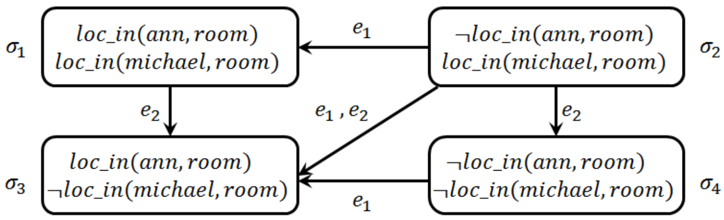

Figure 3: Transition diagram captured by the system description `discourse_ma`.

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

The system description `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 σ_2 and σ_1 says that the occurrence of e_1 may take the system from the former state to the latter. Note how action `e1` cannot occur in state σ_1 (due to the last axiom in `basic_motion`) and thus there is no arc in T going out of σ_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), `ann` is not in `room`, whereas `michael` is in `room`; at time step 0, `ann` moves to (enters) `room`; at time step 1, `michael` moves from (leaves) `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 $\langle \sigma_1, e_1, \sigma_2, e_2, \sigma_4 \rangle$ is not a trajectory.

Informative Piece 3 or $\mathcal{ALM}$ histories: A particular domain scenario defines what we call a *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 *models* of a history. The *MA* discourse provides the following history: action `e1` happens first (at time step 0), then `e2` happens (at time step 1), which we abbreviate below

$$\{hpd(e1, 0), hpd(e2, 1)\}. \tag{7}$$

Given system description `discourse_ma`, trajectory τ_1 is the only model of this history. Thus the initial state of the story conveyed by the *MA* discourse must be σ_2 and the final one must be σ_3.

Answering questions (3-5): Model τ_1 of history (7) allows us to answer questions (3-5). The final state σ_3 of the trajectory τ_1 contains literal `-loc_in(michael, room)`, which translates into the answer

(a)
```
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
      defined
        can_reach : agents * things -> booleans
 axioms
   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).
```

(b)
```
system description discourse_mas
  theory discourse_mas
    import module grasping
  structure discourse_mas
    instances
      ann in agents
      michael in agents
      suitcase in carriables
      room in points
      e1 in move
        actor = ann
        dest = room
      ea in grasp
        grasper = michael
        grasped_thing = suitcase
      e2 in move
        actor = michael
        origin = room
```

Figure 4: (a) $\mathcal{ALM}$ module defining action `grasp`; (b) System description for the *MAS* discourse.

no to question (3). State σ_3 contains `loc_in(ann, room)` that translates into answer *yes* to question (4). Presence of `loc_in(ann, room)` in state σ_3 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 σ_2 of model τ_1 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 σ_1 of τ_1. Initial state σ_2 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 $\mathcal{ALM}$ 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{f} solver CLINGO{F} available at `http://www.mbal.tk/clingof/`. Each answer set corresponds to a model of the given history. A prototype translator from $\mathcal{ALM}$ system descriptions and histories to logic programs is available at `http://tinyurl.com/z6n9fmx`.

***MAS* discourse via $\mathcal{ALM}$**: In order to model the *MAS* discourse, an $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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.

$\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$. We then discussed how these $\mathcal{ALM}$ formalizations can be used to automatically reason about these discourses. In this section, we present a proposal for automating the process of creating an $\mathcal{ALM}$ 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, $r1, r2$, and $r3$) as well as referents denoting events that the discourse describes (namely, $e1$ and $e2$). 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.

r1 r2 r3 e1 e2
entity(r1) entity(r2) entity(r3) property(r1,ann) property(r2,room) property(r3,michael) event(e1) event(e2) eventType(e1,go.01) eventTime(e1,0) eventArgument(e1,a1,r1) eventArgument(e1,a4,r2) eventType(e2,leave.01) eventTime(e2,1) eventArgumet(e2,a0,r3) eventArgument(e2,a1,r2)

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 $\mathcal{ALM}$ system description or via PROPBANK **to** VERBNET **to** $\mathcal{ALM}$: The next question that we tackle is how to map entities, properties, and history present in a given DRS into the vocabulary of $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ descriptions, these do not have formal semantics and are not computer interpretable in prescribed manner.

The $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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* $\mathcal{ALM}$ *modules (such as* `basic_motion_verbnet`*), which provide* $\mathcal{ALM}$*-based semantic annotations for its verb classes. Then, the* VERBNET *lexicon can serve as a lookup table for finding relevant action classes and* $\mathcal{ALM}$ *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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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 $\mathcal{ALM}$ 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

Balduccini, M., C. Baral, and Y. Lierler (2008). Knowledge Representation and Question Answering. In F. van Harmelen, V. Lifschitz, and B. Porter (Eds.), *Handbook of Knowledge Representation*, pp. 779–820. Elsevier.

Baral, C., M. Gelfond, and R. Scherl (2004). Using Answer Set Programming to Answer Complex Queries. In *Workshop on Pragmatics of Question Answering at HLT-NAAC2004*.

Bonial, C., K. Stowe, and M. Palmer (2013a). *SemLink*. `https://verbs.colorado.edu/semlink/` [Accessed: 2017].

Bonial, C., K. Stowe, and M. Palmer (2013b). Renewing and revising semlink. In *Proceedings of The GenLex Workshop on Linked Data in Linguistics*.

Bos, J. (2008). Wide-coverage semantic analysis with boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, STEP '08, Stroudsburg, PA, USA, pp. 277–286. Association for Computational Linguistics.

Bos, J. and K. Markert (2005). Recognising textual entailment with logical inference. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, Stroudsburg, PA, USA, pp. 628–635. Association for Computational Linguistics.

CLEAR (2008). *Ontonotes Sense Groups*. `http://clear.colorado.edu/compsem/index.php?page=lexicalresources&sub=ontonotes` [Accessed: 2017].

Facebook Research (2016). bAbI. `https://research.fb.com/downloads/babi/` [Accessed: 2017].

Gelfond, M. and Y. Khal (2014). *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge University Press.

Gelfond, M. and V. Lifschitz (1991). Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing 9*(3/4), 365–386.

Harmeling, S. (2009, October). Inferring textual entailment with a probabilistically sound calculus*. *Nat. Lang. Eng. 15*(4), 459–477.

Inclezan, D. and M. Gelfond (2016). Modular action language. *TPLP 16*(2), 189–235.

Johansson, R. and P. Nugues (2007a). *Language Technology at LTH*. `http://nlp.cs.lth.se/` [Accessed: 2017].

Johansson, R. and P. Nugues (2007b, June). Lth: Semantic structure extraction using nonprojective dependency trees. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, Prague, Czech Republic, pp. 227–230. Association for Computational Linguistics.

Kamp, H. and U. Reyle (1993). *From discourse to logic*, Volume 1,2. Kluwer.

Kipper-Schuler, K. (2005). *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph. D. thesis, University of Pennsylvania.

Levin, B. (1993). *English verb classes and alternations : a preliminary investigation*. University Of Chicago Press.

MacCartney, B. and C. D. Manning (2007). Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, Stroudsburg, PA, USA, pp. 193–200. Association for Computational Linguistics.

Manning, C. D., M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky (2014). *Stanford CoreNLP a suite of core NLP tools*. http://stanfordnlp.github.io/CoreNLP/ [Accessed: 2017].

NIST (2008). Past RTE data. https://tac.nist.gov/data/RTE/index.html [Accessed: 2017].

Palmer, M. (2005). *Proposition Bank*. http://verbs.colorado.edu/~mpalmer/projects/ace.html [Accessed: 2017].

Palmer, M. (2006). *VerbNet*. https://verbs.colorado.edu/~mpalmer/projects/verbnet.html [Accessed: 2017].

Palmer, M., D. Gildea, and P. Kingsbury (2005, March). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics 31*(1), 71–106.

Todorova, Y. (2011). *Answering questions about dynamic domains from natural language using ASP*. Ph. D. thesis, Texas Tech University.

Todorova, Y. and M. Gelfond (2012). Toward Question Answering in Travel Domains. In *Correct Reasoning*, pp. 311–326.

Weston, J., A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolo (2016). *Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks*. https://arxiv.org/pdf/1502.05698.pdf [Accessed: 2017].