September, 2019

Information Extraction Tool Text2Alm: From Narratives to Action Language System Descriptions

Craig Olson
Yuliya Lierler

Information Extraction Tool TEXT2ALM*: From Narratives to Action Language System Descriptions* ∗

CRAIG OLSON and YULIYA LIERLER

University of Nebraska Omaha
6001 Dodge St, Omaha, NE 68182, USA
(*e-mail:* cdolson,ylierler@unomaha.edu)

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

In this work we design a narrative understanding tool TEXT2ALM. This tool uses an action language $\mathcal{ALM}$ to perform inferences on complex interactions of events described in narratives. The methodology used to implement the TEXT2ALM system was originally outlined by Lierler, Inclezan, and Gelfond (13) via a manual process of converting a narrative to an $\mathcal{ALM}$ model. It relies on a conglomeration of resources and techniques from two distinct fields of artificial intelligence, namely, natural language processing and knowledge representation and reasoning. The effectiveness of system TEXT2ALM is measured by its ability to correctly answer questions from the bAbI tasks published by Facebook Research in 2015. This tool matched or exceeded the performance of state-of-the-art machine learning methods in six of the seven tested tasks. We also illustrate that the TEXT2ALM approach generalizes to a broader spectrum of narratives.

KEYWORDS: Question answering, information extraction, action languages

1 Introduction

The field of Information Extraction (IE) is concerned with gathering snippets of meaning from text and storing the derived data in structured, machine interpretable form. Consider a sentence

BBDO South in Atlanta, which handles corporate advertising for Georgia-Pacific, will assume additional duties for brands like Angel Soft, said Ken Haldin, a spokesman for Georgia-Pacific from Atlanta.

A sample IE system that focuses on identifying organizations and their corporate locations may extract the following predicates from this sentence:

$$locatedIn(BBDOSouth, Atlanta) \quad locatedIn(GeorgiaPacific, Atlanta)$$

These predicates can then be stored either in a relational database or a logic program, and queried accordingly by well-known methods in computer science. Thus, IE allows us to turn unstructured data present in text into structured data easily accessible for automated querying.

In this paper, we focus on an IE system that is capable of processing simple narratives with *action verbs*, in particular, verbs that express physical acts such as *go*, *give*, and *put*. Consider a sample narrative that we refer to as the *JS* discourse:

∗ We would like to thank Parvathi Chundi, Nicholas Hippen, Brian Hodges, Joseph Meyer, Gang Ling, and Ryan Schuetzler for their valuable feedback. We appreciate the insights from Michael Gelfond, Daniela Inclezan, Edward Wertz, and Yuanlin Zhang on their work on language $\mathcal{ALM}$, the COREALMLIB library, and system CALM.

John traveled to the hallway. (1)

Sandra journeyed to the hallway. (2)

The actions *travel* and *journey* in the narrative describe changes to the narrative's environment, and can be coupled with the readers commonsense knowledge to form and alter the readers mental picture for the narrative. For example, after reading sentence (1), a human knows that *John* is the subject of the sentence and *traveled* is an action verb describing an action performed by *John*. A human also knows that *traveled* describes the act of motion, and specifically that *John*'s location changes from an arbitrary initial location to a new destination, the *hallway*. Lierler et al. (13) outline a methodology for constructing a Question Answering (QA) system by utilizing IE techniques. Their methodology focuses on performing inferences using the complex interactions of events in narratives. Their process utilizes an action language $\mathcal{ALM}$ (8) and an extension of the VERBNET lexicon (19; 11). Language $\mathcal{ALM}$ enables a system to structure knowledge regarding complex interactions of events and implicit background knowledge in a straight-forward and modularized manner. The knowledge represented in $\mathcal{ALM}$ is processed by means of logic programming under answer set semantics and can be used to derive inferences about a given text. The proposed methodology in (13) assumes the extension of the VERBNET lexicon with interpretable semantic annotations in $\mathcal{ALM}$. The VERBNET lexicon groups English verbs into classes allowing us to infer that such verbs as *travel* and *journey* practically refer to the same class of events.

The processes described in (13) are exemplified via two sample narratives processed manually. The authors translated those narratives to $\mathcal{ALM}$ programs by hand and wrote the supporting $\mathcal{ALM}$ modules to capture knowledge as needed. To produce $\mathcal{ALM}$ system descriptions for considered narratives, the method by Lierler et al. (13) utilizes NLP resources, such as semantic role labeler LTH (9), parser and co-reference resolution tools of CORENLP (15), and lexical resources PROPBANK (20) and SEMLINK (3). Ling (14) used these resources to automate parts of the method in the TEXT2DRS system. In particular, TEXT2DRS extracts entities, events, and their relations from a given action-based narrative. A narrative understanding system developed within this work, TEXT2ALM, utilizes TEXT2DRS and automates the remainder of the method outlined in (13). When considering the *JS* discourse as an example, system TEXT2ALM produces a set of facts in spirit of the following:

$$move(john, hallway, 0) \quad move(sandra, hallway, 1) \tag{3}$$

$$loc\_in(john, hallway, 1) \quad loc\_in(john, hallway, 2) \quad loc\_in(sandra, hallway, 2), \tag{4}$$

where $0, 1, 2$ are time points associated with occurrences of described actions in the *JS* discourse. Intuitively, time point 0 corresponds to a time prior to utterance of sentence (1). Time point 1 corresponds to a time upon the completion of the event described in (1). Facts in (3) and (4) allow us to provide grounds for answering questions related to the *JS* discourse such as:

Question:	Ground:
Is *John* inside the *hallway* at the end of the story (time 2)?	$loc\_in(john, hallway, 2)$
Who is in the *hallway* at the end of the story?	$loc\_in(John, hallway, 2)$ $loc\_in(sandra, hallway, 2)$

We note that modern NLP tools and resources prove to be sufficient to extract facts (3) given the *JS* discourse. Yet, inferring facts such as (4) requires complex reasoning about specific ac-

tions present in a given discourse and modeling such common sense knowledge as *inertia axiom* (stating that *things normally stay as they are*) (12). System TEXT2ALM combines the advances in NLP and knowledge representation and reasoning (KRR) to tackle the complexities of converting narratives such as the *JS* discourse into a structured form such as facts in (3-4).

The effectiveness of system TEXT2ALM is measured by its ability to answer questions from the bAbI tasks (22). These tasks were proposed by Facebook Research in 2015 as a benchmark for evaluating basic capabilities of QA systems in twenty categories. Each of the twenty bAbI QA tasks is composed of narratives and questions, where 1000 questions are given in training set and 1000 questions are given in a testing set. We extend the information extraction component of the TEXT2ALM by a specialized QA processing module to tackle seven of the bAbI tasks containing narratives with action verbs. Tool TEXT2ALM matched or exceeded the performance of modern machine learning methods in six of these tasks. We also illustrate that the TEXT2ALM approach generalizes to a broader spectrum of narratives than present in bAbI.

We start the paper by a review of relevant tools and resources stemming from NLP and KRR communities. We then proceed to describe the architecture of the TEXT2ALM system implemented in this work. We conclude by providing the evaluation data on the system.

2 Background

NLP Resource VERBNET: VERBNET is a domain-independent English verb lexicon organized into a hierarchical set of verb classes (19; 11). The verb classes aim to achieve syntactic and semantic coherence between members of a class. Each class is characterized by a set of verbs and their thematic roles. For example, the verb *run* is a member of the VERBNET class RUN-51.3.2. This class is characterized by

- 96 members including verbs such as *bolt, frolic, scamper,* and *weave*,
- four thematic roles, namely, *theme*, *initial location*, *trajectory* and *destination*,
- two subbranches: RUN-51.3.2-1 and RUN-51.3.2-2. For instance, RUN-51.3.2-2 has members *gallop, skip,* and *strut*, and has additional thematic roles *agent*, *result*, and *source*.

Dynamic Domains, Transition Diagrams, and Action Language $\mathcal{ALM}$: *Action languages* are formal KRR languages that provide convenient syntactic constructs to represent knowledge about dynamic domains. The knowledge is compiled into a transition diagram, where nodes correspond to possible states of a considered dynamic domain and edges correspond to actions/events whose occurrence signal transitions in the dynamic system. The *JS* discourse exemplifies a narrative modeling a dynamic domain with three entities *John, Sandra, hallway* and four actions, specifically:

ajin – *John* travels into the *hallway*	*ajout* – *John* travels out of the *hallway*
asin – *Sandra* travels into the *hallway*	*asout* – *Sandra* travels out of the *hallway*

Scenarios of a dynamic domain correspond to *trajectories* in the domain's transition diagram. Trajectories are sequences of alternating states and actions. A trajectory captures the sequence of events, starting with the initial state associated with time point 0. Each edge is associated with the time point incrementing by 1.

In this work we utilize an advanced action language $\mathcal{ALM}$ (8) to model dynamic domains of given narratives. This language can represent knowledge pertaining to the commonalities of similar actions through means of logic programming under answer set semantics. This is a crucial feature of the language that made it especially fit for this work. In addition, there are efficient solving techniques available for $\mathcal{ALM}$. In particular, a translation from $\mathcal{ALM}$ to logic programs under answer set semantics (answer set programs) was proposed by Inclezan and Gelfond in (8). In turn, answer set programming is a prominent subfield of automated reasoning supported by a

```
system description JS_discourse
  theory JS_discourse_theory
    module JS_discourse_module
      sort declarations
        points, agents :: universe
        move :: actions
          attributes
            actor : agents -> booleans
            origin : points -> booleans
            destination : points -> booleans
      function declarations
        fluents
          basic
            loc_in : agents * points -> booleans
      axioms
        dynamic causal laws
          occurs(X) causes loc_in(A,D) if instance(X,move), actor(X,A),
                                          destination(X,D).
        executability conditions
          impossible occurs(X) if instance(X,move), actor(X,A), loc_in(A,P),
                                  origin(X,O), P!=O.
          impossible occurs(X) if instance(X,move), actor(X,A), loc_in(A,P),
                                  destination(X,D), P=D.
  structure john_and_sandra
    instances
      john, sandra in agents
      hallway in points
      ajin in move
        actor(john) = true
        destination(hallway) = true
      asin in move
        actor(sandra) = true
        destination(hallway) = true
```

Fig. 1: An $\mathcal{ALM}$ system description formalizing the *JS* discourse dynamic domain

plead of efficient answer set solvers (tools that find solutions to answer set programs). Here we use system CALM (21) that translates $\mathcal{ALM}$ theories into answer set programs in the language of answer set solver SPARC (1). We then use system SPARC to find solutions for $\mathcal{ALM}$ theories of interest.

We illustrate the syntax and semantics of $\mathcal{ALM}$ using the *JS* discourse dynamic domain by first defining an $\mathcal{ALM}$ *system description* and then an $\mathcal{ALM}$ *history* for this discourse. In language $\mathcal{ALM}$, a dynamic domain is described via a *system description* that captures a transition diagram specifying the behavior of a given domain. An $\mathcal{ALM}$ system description consists of a theory and a structure. A *theory* is comprised of a hierarchy of modules, where a module represents a unit of general knowledge describing relevant sorts, properties, and the effects of actions. The *structure* declares instances of entities and actions of the domain. Figure 1 illustrates these concepts with the $\mathcal{ALM}$ formalization of the *JS* discourse domain.

The *JS* discourse theory uses a single module to represent the knowledge relevant to the domain. The module declares the sorts (*agents*, *points*, *move*) and the property (`loc_in`) to represent entities and attributes of the domain. *Actions* utilize attributes to define the roles of participating entities. For instance, *destination* is an attribute of *move* that denotes the final location of the mover. Here we ask a reader to draw a parallel between the notions of an attribute and a VERBNET thematic role.

The *JS* discourse theory also defines two types of axioms, dynamic causal laws and executability conditions, to represent commonsense knowledge associated with a *move* action. The dynamic

causal law states that if a *move* action occurs with a given *actor* and *destination*, then the *actor*'s location becomes that of the *destination*. The executability conditions restrict an action from occurring if the action is an instance of *move*, where the *actor* and *actor*'s location are defined, but either (i) the *actor*'s location is not equal to the *origin* of the *move* event or (ii) the *actor*'s location is already the *destination*.

An $\mathcal{ALM}$ structure in Figure 1 defines the entities and actions from the *JS* discourse. For example, it states that *john* and *sandra* are *agents*. Also, action *ajin* is declared as an instance of *move* where *john* is the *actor* and *hallway* is the *destination*.

An $\mathcal{ALM}$ system description can be coupled with a *history*. A history is a particular scenario described by observations about the values of properties and occurring events. In the case of narratives, a history describes the sequence of events by stating occurrences of specific actions at given time points. For instance, the *JS* discourse history contains the events

- *John moves* to the *hallway* at the beginning of the story (an action *ajin* occurs at time 0) and
- *Sandra moves* to the *hallway* at the next point of the story (an action *asin* occurs at time 1).

The following history is appended to the end of the system description in Figure 1 to form an $\mathcal{ALM}$ program for the *JS* disocurse. We note that *happened* is a keyword that captures the occurrence of actions.

```
history
   happened(ajin, 0).
   happened(asin, 1).
```

An $\mathcal{ALM}$ Solver CALM: System CALM is an $\mathcal{ALM}$ solver developed at Texas Tech University by Wertz, Chandrasekan, and Zhang (21). It uses an $\mathcal{ALM}$ program to produce a "model" for an encoded dynamic domain. The engine for system CALM (i) constructs a logic program under stable model/answer set semantics (6), whose answer sets/solutions are in one-to-one correspondence with the models of the $\mathcal{ALM}$ program, and (ii) uses an answer set solver SPARC (1) for finding these models. In this manner, CALM processes the knowledge represented by an $\mathcal{ALM}$ program to enable reasoning capabilities. The $\mathcal{ALM}$ program in Figure 1 follows the CALM syntax. However, system CALM requires two additional components for this program to be executable. The user must specify (i) the computational task and (ii) the max time point considered.

In our work we utilize the fact that system CALM can solve a task of temporal projection, which is the process of determining the effects of a given sequence of actions executed from a given initial situation (which may be not fully determined). In the case of a narrative the initial situation is often unknown, whereas the sequence of actions are provided by the discourse. Inferring the effects of actions allows us to answer questions about the narrative's domain. We insert the following statement in the $\mathcal{ALM}$ program prior to the history to perform temporal projection:

```
temporal projection
```

Additionally, CALM requires the max number of steps to be stated. Intuitively, we see this number as an upper bound on the "length" of considered trajectories. This information denotes the final state's time point in temporal projection problems. We insert the following line in the $\mathcal{ALM}$ program to define the max steps for the *JS* discourse $\mathcal{ALM}$ program:

```
max steps 3
```

For the case of the temporal projection task, a model of an $\mathcal{ALM}$ program is a trajectory in the transition system captured by the $\mathcal{ALM}$ program that is "compatible" with the provided history.

```
occurs(X) causes location(O, D) if instance(X, move),
                                   object(X, O),
                                   destination(X, D),
                                   instance(D, spatial_entity).
```

Fig. 2: Commonsense Knowledge Axiom from the COREALMLIB *motion* module.

A compatible model correlate to the answer set solved by CALM. For the *JS* discourse $\mathcal{ALM}$ program, the CALM computes a model that includes the following expressions:

```
happened(ajin, 0),        happened(asin, 1),
loc_in(john, hallway, 1), loc_in(sandra, hallway, 2), loc_in(john, hallway, 2)
```

$\mathcal{ALM}$ *Knowledge Base* COREALMLIB: The COREALMLIB is an $\mathcal{ALM}$ library of generic commonsense knowledge for modeling dynamic domains developed by Inclezan (7). The library's foundation is the Component Library or CLib (2), which is a collection of general, reusable, and interrelated components of knowledge. CLib was populated with knowledge stemming from linguistic and ontological resources, such as VERBNET, WORDNET, FRAMENET, a thesaurus, and an English dictionary. The COREALMLIB was formed by translating CLib into $\mathcal{ALM}$ to obtain descriptions of 123 action classes grouped into 43 reusable modules. The modules are organized into a hierarchical structure, and contain action classes and axioms to support commonsense reasoning. An example of one such axiom from the *motion* module is provided in Figure 2. This axiom states that if a *move* action occurs where *O* is the object moving and *D* is a *spatial entity* and the destination, then the *location* of *O* becomes *D*.

3 System TEXT2ALM Architecture

Lierler, Inclezan, and Gelfond (13) outline a methodology for designing IE/QA systems to make inferences based on complex interactions of events in narratives. This methodology is exemplified with two sample narratives completed manually by the authors. System TEXT2ALM automates this process. Figure 3 pretenses the architecture of the system. It implements four main tasks/processes:

1. TEXT2DRS Processing – Entity, Event, and Relation Extraction
2. DRS2ALM Processing – Creation of $\mathcal{ALM}$ Program
3. CALM Processing – $\mathcal{ALM}$ Model Generation and Interpretation
4. QA Processing

Figure 3 denotes each process by its own column. Ovals identify inputs and outputs. Systems or resources are represented with white, grey, and black rectangles. White rectangles denote existing, unmodified resources. Grey rectangles are used for existing, but modified resources. Black rectangles signify newly developed subsystems. The first three processes form the core of TEXT2ALM, seen as an IE system. The QA Processing component is specific to the bAbI QA benchmark that we use to illustrate the validity of the approach advocated by TEXT2ALM. The system's source code is available at `https://github.com/cdolson19/Text2ALM`.

3.1 TEXT2DRS *Processing*

The method by Lierler et al. (13) utilizes NLP resources, such as semantic role labeler LTH (9), parsing and coreference resolution tools of CORENLP (15), and lexical resources PROPBANK (20) and SEMLINK (3) to produce $\mathcal{ALM}$ system descriptions for considered narratives. System TEXT2DRS (14) was developed with these resources to deliver a tool that extracts entities, events, and their relations from given narratives. The TEXT2DRS tool formed the starting point in the development of TEXT2ALM due to its ability to extract basic entity and relational information from a narrative.

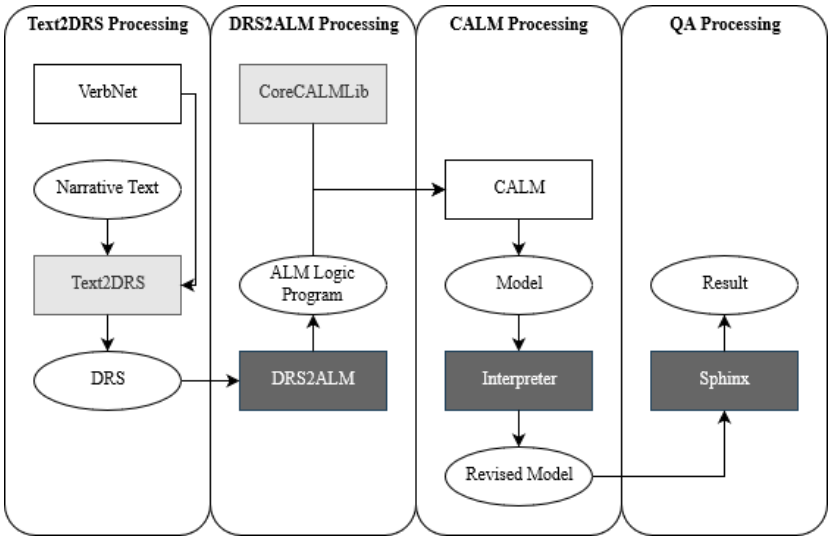

Fig. 3: System TEXT2ALM Architecture

```
entity(r1).           entity(r2).              entity(r3).
property(r1, "John"). property(r2, "hallway"). property(r3, "Sandra").

event(e1).                                  event(e2).
eventType(e1, "run-51.3.2-1").              eventType(e2, "run-51.3.2-1").
eventTime(e1, 0).                           eventTime(e2, 1).
eventArgument(e1, "Theme", r1).             eventArgument(e2, "Theme", r3).
eventArgument(e1, "Destination", r2).       eventArgument(e2, "Destination", r2).
```

Fig. 4: DRS for the *JS* discourse.

The output of the TEXT2DRS system is called a discourse representation structure, or DRS (10). A DRS captures key information present in discourse in a structured form. For example, Figure 4 presents the DRS for the *JS* discourse.

This DRS states that there are three entities and two events that take part in the *JS* narrative. The DRS assigns names, or referents, to the entities ($r1, r2$, and $r3$) and the events ($e1, e2$). For instance, entity $r1$ and event $e1$ denote *John* and an event representing the VERBNET class RUN-51.3.2-1, respectively. The *theme* (which is one of the thematic roles associated with RUN-51.3.2-1) of event $e1$ is entity $r1$ (or, *John*) and the *destination* of this event is entity $r2$ (or, *hallway*). Event $e1$ occurs at time point 0, while event $e2$ occurs at time point 1. We refer an interested reader to the work by Ling (14) for the details of the TEXT2DRS component. In realms of this project, TEXT2DRS was modified to accommodate VERBNET v 3.3 (in place of VERBNET v 2), which provides broader coverage of verbs.

3.2 DRS2ALM *Processing*

The DRS2ALM subsystem is concerned with combining commonsense knowledge related to events in a discourse with the information from the DRS generated by TEXT2DRS. The goal of this process is to produce an $\mathcal{ALM}$ program consisting of a system description and a history for the scenario described by the narrative. The system description is composed of a theory containing relevant commonsense knowledge and a structure that is unique for a given narrative.

Since the structure is specific to a given narrative, it is created using the information from a narrative's DRS. Meanwhile the theory represents the commonsense knowledge associated with a narrative's actions. Thus, the theory depends on a general, reusable knowledge base pertaining to actions. The COREALMLIB knowledge base was modified to form CORECALMLIB to fit this need of the TEXT2ALM system. We organize this section by (1) explaining how CORECALMLIB was obtained and (2) provide details on how a narrative's $\mathcal{ALM}$ program is generated.

Library CORECALMLIB: To obtain the CORECALMLIB knowledge base, the following modifications to the COREALMLIB were made:

1. Syntactic adjustments
2. Property extractions
3. VERBNET extensions
4. Axiom changes

First, syntactic adjustments were implemented to make the library compatible with the CALM syntax. Second, we observed that the COREALMLIB has instances where properties (fluents) with the same name are declared in multiple modules. Yet semantically, these properties are assumed to be the same across all modules. We found this approach counter-intuitive from the point of knowledge-base design, thus we extracted all fluent declarations from COREALMLIB modules and created new modules whose purpose was to declare fluents. These modules were organized by their properties and grouped similar properties together. The original COREALMLIB modules now import the necessary properties as needed. Regarding VERBNET extensions, COREALMLIB was further modified by adding a module for every VERBNET class we observed in the bAbI QA task training sets. We discuss these training sets in detail in Section 4. In particular, 52 of VERBNET's 274 classes were formalized with modules in CORECALMLIB. Each VERBNET module defines a sort for that verb class that inherits from one of the 123 action classes stemming from COREALMLIB. Specifically, we utilize 15 action classes formalized in COREALMLIB, stemming from 9 of its total 43 modules. Thematic roles from the VERBNET lexicon are then mapped via state constraints to the attributes associated with actions already used by the COREALMLIB library. These VERBNET modules are stored in a CORECALMLIB sub-library that we call VN_CLASS_LIBRARY. Lastly, we modified and added axioms into some COREALMLIB modules after identifying pieces of knowledge that were not represented within the original library. When not considering fluent extractions, fluents were altered or added to only four modules from the original COREALMLIB. This supports the hypothesis that COREALMLIB can provide an effective baseline for commonsense reasoning about actions. All modifications to the COREALMLIB to form CORECALMLIB are explained further in (18).

$\mathcal{ALM}$ *Program Generation*: The DRS2ALM processing step generates an $\mathcal{ALM}$ program for a given discourse by combining the information in a narrative's DRS and the CORECALMLIB library. We first examine the theory in the program's system description. We start by identifying the general knowledge associated with a narrative's domain by importing the VERBNET modules from the CORECALMLIB for all VERBNET classes associated with a narrative. These provide the commonsense knowledge backbone for the actions in the narrative. Then, we define a new module unique to the narrative. This module declares entities from the narrative as new sorts inheriting from base CORECALMLIB sorts. We chose to declare the narrative's entities as new sorts to provide more flexibility to define additional, unique attributes associated with the entities if the need arises. However, to declare these new sorts we must identify the CORECALMLIB parent sort to inherit from. We rely on the VERBNET thematic roles associated with an entity to make this selection. We grouped VERBNET thematic roles into four parent sorts of CORECALMLIB by reviewing the thematic roles associated with the VERBNET classes in the training sets and attempting to map these to the most similar sorts defined by the original COREALMLIB.

Parent Sort	living_entity	place	spatial_entity	entity
Associated Thematic Roles	Actor, Agent Beneficiary Cause, Co-Agent Co-Theme, Recipient Experiencer Participant, Patient Theme, Undergoer	Location Place	Destination Initity_location Source	Instrument, Material Pivot, Product Duration Stimulus, Time, Extent Trajectory, Initial_time Topic, Value, Goal Result, Attribute Final_time, Frequency

Fig. 5: VERBNET Thematic Roles to COREALMLIB Sorts

Figure 5 presents the groupings. If an entity is associated with roles from different categories, we use a prioritized sort order defined as follows:

$$living\_entity >> place >> spatial\_entity >> entity,$$

where $>>$ is transitive and states that the left argument has a higher priority than the right one.

We now turn our attention to the process of generating the structure and history for the $\mathcal{ALM}$ program. The structure declares the specific entities and events from the narrative. Entity IDs from a given narrative's DRS are defined as instances of the corresponding entity sort from the theory. Events are also declared as instances of their associated VERBNET class sort, and the entities related to events are listed as attributes of these events. The history states the order and timepoints in which narrative's events happened. We extract this information from the arguments expressed in DRS. To exemplify the described process, Figure 6 presents the $\mathcal{ALM}$ program output by the DRS2ALM Processing stage applied towards the *JS* discourse DRS in Figure 4. Note that the $\mathcal{ALM}$ theory in Figure 6 imports the VERBNET module for RUN-51.3.2-1 from the VN_CLASS_LIBRARY. The two events in the *JS* discourse were identified as members of the VERBNET class RUN-51.3.2-1. Thus, the module associated with this class is imported to retrieve the knowledge relevant to RUN events in the *JS* discourse domain.

3.3 CALM *and QA Processing*

In the CALM Processing performed by TEXT2ALM, the CALM system is invoked on a given narrative's $\mathcal{ALM}$ program that was generated by the DRS2ALM Processing stage. The CALM system computes a model via logic programming under answer set semantics. We then perform post-processing on this model to make its content more readable for a human by replacing all entities IDs with their names from the narrative. For instance, given the $\mathcal{ALM}$ program in Figure 6, the output of the CALM Processing will include expressions:

`loc_in(John,hallway,1), loc_in(John,hallway,2), loc_in(Sandra,hallway,2).`

We note that no other `loc_in` fluents will be present in the output.

A model derived by the CALM system contains facts about the entities and events from the narrative supplemented with basic commonsense knowledge associated with the events. We use a subset of the bAbI QA tasks to test the TEXT2ALM system's IE effectiveness and implement QA capabilities within the SPHINX subsystem (see Figure 3). It utilizes regular expressions to identify the kind of question that is being asked and then query the model for relevant information to derive an answer. The SPHINX system is specific to the bAbI QA task and is not a general purpose question answering component.

Additional information on the components of system TEXT2ALM are given in (18).

```
system description js_discourse
  theory js_discourse_theory
    import t_run_51_32.m_run_51_3_2_1 from VN_class_library
    module js_discourse
      depends on t_run_51_3_2.m_run_51_3_2_1
      sorts declarations
        john :: living_entity
        hallway :: spatial_entity
        sandra :: living_entity
  structure js_discourse_structure
    instances
      r1 in john
      r2 in hallway
      r3 in sandra
      e1 in run_51_3_2_1
        vn_theme(r1) = true
        vn_destination(r2) = true
      e2 in run_51_3_2_1
        vn_theme(r3) = true
        vn_destination(r2) = true
temporal projection
max steps 3
history
  happened(e1,0).
  happened(e2 1).
```

Fig. 6: An $\mathcal{ALM}$ system description automatically created by the DRS2ALM processing

4 TEXT2ALM Evaluation

Related Work: Many modern QA systems predominately rely on machine learning techniques. However, there has recently been more work related to the design of QA systems combining advances of NLP and KRR. The TEXT2ALM system is a representative of the latter approach. Other approaches include the work by Clark, Dalvi, and Tandon (4) and Mitra and Baral (17). Mitra and Baral (17) use a training dataset to learn the knowledge relevant to the action verbs mentioned in the dataset. They posted nearly perfect test results on the bAbI tasks. However, this approach doesn't scale to narratives that utilize other action verbs which are not present in the training set, including synonymous verbs. For example, if their system is trained on bAbI training data that contains verb *travel* it will process the *JS* discourse correctly. Yet, if we alter the *JS* discourse by exchanging *travel* with a synonymous word *stroll*, their system will fail to perform inferences on this altered narrative (note that *stroll* does not occur in the bAbI training set). We address this limitation in the TEXT2ALM system because the system does not rely upon the training narratives for the commonsense knowledge. If the verbs occurring in narratives belong to VERBNET classes whose semantics have been captured within CORECALMLIB then TEXT2ALM is normally able to process them properly.

Another relevant QA approach is the work by Clark, Dalvi, and Tandon (4). This approach uses VERBNET to build a knowledge base containing rules of preconditions and effects of actions utilizing the semantic annotations that VERBNET provides for its classes. In our work, we can view $\mathcal{ALM}$ modules associated with VERBNET classes as machine interpretable alternatives to these annotations. However, Clark et al. (4) use the first and most basic action language STRIPS (5) for inference. The STRIPS language allows more limited capabilities than the $\mathcal{ALM}$ language in modeling complex interactions between events.

Evaluation: We use a subset of Facebook AI Researchs bAbI dataset (22) to evaluate system TEXT2ALM. These tasks were proposed by Facebook Research in 2015 as a benchmark for eval-

```
1 Mary moved to the bathroom.        5 Mary went back to the kitchen.
2 Sandra journeyed to the bedroom.   6 Mary went back to the garden.
3 Mary got the football there.       7 Where is the football? garden 3 6
4 John went to the kitchen.
```

Fig. 7: Example narrative and question from bAbI task 2 training set

uating basic capabilities of QA systems in twenty categories. Each of the twenty bAbI QA tasks is composed of narratives and questions, where 1000 questions are given in training set and 1000 questions are given in a testing set. The goal of the tasks are to answer the questions in the testing sets correctly while also minimizing the number of questions used from the training set to develop a solution. We evaluate the TEXT2ALM system with all 1000 questions in the testing sets for tasks 1, 2, 3, 5, 6, 7, and 8. These tasks are selected for two reasons. First, these tasks contain action-based narratives that are of focus in this work. Second, the underlying IE engine TEXT2DRS requires further development to formulate representations of more advanced sentence structures, such as those containing negation and indefinite knowledge. Figure 7 provides an example of a narrative and a question from the training set of bAbI task 2-Two Supporting Facts. For this task, a QA system must combine information from two sentences in the given narrative. The narrative in Figure 7 consists of six sentences. A question is given in line 7, followed by the answer and identifiers for the two sentences that provide information to answer the question.

The bAbI dataset enables us to compare TEXT2ALM's IE/QA ability with other modern approaches designed for this task. The left hand side of Figure 8 compares the accuracy of the TEXT2ALM system with the machine learning approach AM+NG+NL MemNN described by Weston et al. (22). In that work, the authors compared results from 8 machine learning approaches on bAbI tasks and the AM+NG+NL MemNN (Memory Network) method performed best almost across the board. There were two exceptions among the seven tasks that we consider. For the Task 7-Counting the AM+N-GRAMS MemNN algorithm was reported to obtain a higher accuracy of 86%. Similarly, for the Task 8-Lists/Sets the AM+NONLINEAR MemNN algorithm was reported to obtain accuracy of 94%. Figure 8 also presents the details on the Inductive Rule Learning and Reasoning (IRLR) approach by (17). We cannot compare TEXT2ALM performance with the methodology by (4) because their system is not available and it has not been evaluated using the bAbI tasks.

System TEXT2ALM matches the Memory Network approach by Weston et al. (22) at 100% accuracy in tasks 1, 2, 3, and 6 and performs better on tasks 7 and 8. When compared to the methodology by Mitra and Baral (17), the Text2ALM system matches the results for tasks 1, 2, 3, 6, and 8, but is outperformed in tasks 5 and 7.

The results of the TEXT2ALM system were comparable to the industry-leading results with one outlier, namely, task 5. We investigated the reason. It turns out that the testing set frequently contained a phrase of the form:

`Entity1` *handed the* `Object` *to* `Entity2`. e.g., *Fred handed the football to Bill.*

The TEXT2ALM system failed to properly process such phrases because the semantic role labeler LTH, a subcomponent of the TEXT2DRS system, incorrectly annotated the sentence. In particular, LTH consistently considered a reading in spirit of the following: *Fred handed Bill's football away*. This annotation error prevents TEXT2DRS from adding crucial event argument to the DRS stating that `Entity2` plays the thematic role of *destination* in the phrase. Consequently, the TEXT2ALM system does not realize that possession of the object was passed from `Entity1` to `Entity2`.

	Accuracy		
bAbI Task	AM+NG+NL Mem NN	IRLR	TEXT2ALM
1-Single Sup. Facts	100	100	100
2-Two Sup. Facts	100	100	100
3-Three Sup. Facts	100	100	100
5-Three Arg. Rels.	98	100	22
6-Yes/No	100	100	100
7-Counting	85	100	96.1
8-Lists/Sets	91	100	100

Fig. 8: System Evaluations

bAbI+ Task	Accuracy	bAbI+ Task	Accuracy
1-Single Sup. Facts	97.9	6-Yes/No	99
2-Two Sup. Facts	97.8	7-Counting	96.1
3-Three Sup. Facts	97.4	8-Lists/Sets	100
5-Three Arg. Rels.	19		

Fig. 9: TEXT2ALM Evaluation on bAbI+ Tasks

Even though our system does not match the scores and breadth of testing as the approach by Mitra and Baral (17), who tested on all 20 bAbI tasks, we consider the scores obtained by TEXT2ALM interesting for several reasons. First, the approach implemented by TEXT2ALM suggests that lexical resources, such as VERBNET, PROPBANK, and SEMLINK can be utilized effectively to support IE and KRR tasks. Second, the scores support the hypothesis that a relatively few, but general, commonsense rules about the effects of actions can be used to generalize to a broad number of similar actions. To illustrate that the approach by system TEXT2ALM generalizes well, we create a variant of the bAbI task, which we call bAbI+. We obtain bAbI+ by changing 50% of action verbs occurring in testing set narratives with their synonymous counterparts. For example, 50% of instances of *travelled* and *grabbed* were replaced with *sprinted* and *seized*, respectively. In total, 13 synonymous verbs were introduced. To ensure that TEXT2ALM system handles the bAbI+ tasks, two extensions were made to its resources. First, the CORE-CALMLIB knowledge base was augmented with appropriate mappings for two more VERBNET classes. Second, two new entries in SEMLINK were introduced for verbs lacking the mappings and four modifications to existing SEMLINK entries were required (where SEMLINK is a key resource of the TEXT2DRS system). Figure 9 presents the accuracy of the TEXT2ALM system on the bAbI+ tasks.

5 Conclusion and Future Work

Lierler, Inclezan, and Gelfond (13) outline a methodology for designing IE/QA systems to make inferences based on complex interactions of events in narratives. To explore the feasibility of this methodology, we built the TEXT2ALM system to take an action-based narrative as input and output a model encoding facts about the given narrative. We tested the system over tasks 1, 2, 3, 5, 6, 7, and 8 from the bAbI QA dataset (22). System TEXT2ALM matched or outperformed the results of modern machine learning methods in all of these tasks except task 5. It also matched the results of another KRR approach (17) in tasks 1, 2, 3, 6, and 8, but did not perform as well in tasks 5 and 7. However, our approach adjusts well to narratives with a more diverse lexicon. Additionally, the ability of the CORECALMLIB to represent the interactions of events in the bAbI narratives serves as a proof of usefulness of the original COREALMLIB endeavor.

We conclude our work by listing future research directions in some areas, (i) Expanding narrative processing capabilities, (ii) Expanding QA ability, (iii) Exploring additional reasoning tasks.

The bAbI QA tasks provided basic narratives to evaluate the effectiveness of information extraction by system TEXT2ALM. However, these are basic narratives with simple sentence structures. Future work includes expanding the narrative processing capabilities of system TEXT2ALM, as well as reducing the impact of semantic role labeling errors. We need to enhance the TEXT2DRS subsystem's capabilities in order to provide more detailed IE on narratives. Also, so far we provided $\mathcal{ALM}$ annotations via the CORECALMLIB library for twenty two classes of VERBNET. In the future we intend to cover all VERBNET classes.

Questions in the bAbI QA tasks follow pre-specified formats. Therefore, system TEXT2ALM's QA ability relies on simple regular expression matching. Further research is required on representing generic questions and answers before using the system's IE abilities in other applications. Additionally, our approach should be tested on more advanced QA datasets, such as PROPARA (16). Conducting tests on the PROPARA dataset would enable us to compare the results of TEXT2ALM to the approach by (4).

Finally, we will build on TEXT2ALM's reasoning abilities. For example, the CALM model may sometimes not contain atoms that could be argued as reasonable. For example, given a narrative *The monkey is in the tree. The monkey grabs the banana.*, the CALM model will contain fluents stating that the *monkey*'s location is the *tree* at time point 1, the *monkey* is holding the *banana* at time point 2, and the *banana*'s location is the *tree* at time point 2. However, it is also natural to infer that the *banana*'s location is the *tree* when the *monkey* grabs it (time point 1). Yet, that requires reasoning that goes beyond temporal projection.

References

Evgenii Balai, Michael Gelfond & Yuanlin Zhang (2013): *Towards Answer Set Programming with Sorts*. In Pedro Cabalar & Tran Cao Son, editors: *Logic Programming and Nonmonotonic Reasoning*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 135–147.

Ken Barker, Bruce Porter & Peter Clark (2001): *A Library of Generic Concepts for Composing Knowledge Bases*. *Proceedings of the 1st International Conference on Knowledge Capture - K-CAP*, pp. 14–21, . Available at `http://portal.acm.org/citation.cfm?doid=500737.500744http://www.cs.utexas.edu/users/mfkb/papers/kcap01.pdf`.

Claire Bonial, Kevin Stowe & Martha Palmer (2013): *SemLink*. `https://verbs.colorado.edu/semlink/`.

Peter Clark, Bhavana Dalvi & Niket Tandon (2018): *What Happened? Leveraging VerbNet to Predict the Effects of Actions in Procedural Text*. *CoRR* abs/1804.05435. Available at `http://arxiv.org/abs/1804.05435`.

Richard E. Fikes & Nils J. Nilsson (1971): *Strips: A new approach to the application of theorem proving to problem solving*. Artificial Intelligence 2(3-4), pp. 189–208, .

Michael Gelfond & Vladimir Lifschitz (1988): *The stable model semantics for logic programming*. In Robert Kowalski & Kenneth Bowen, editors: Proceedings of International Logic Programming Conference and Symposium, MIT Press, pp. 1070–1080.

Daniela Inclezan (2016): *CoreALMlib: An ALM library translated from the Component Library*. Theory and Practice of Logic Programming 16(5-6), pp. 800–816, .

Daniela Inclezan & Michael Gelfond (2016): *Modular action language ALM*. *TPLP* 16(2), pp. 189–235, . Available at `http://dx.doi.org/10.1017/S1471068415000095`.

Richard Johansson & Pierre Nugues (2007): *LTH: Semantic Structure Extraction using Nonprojective Dependency Trees*. In: Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007), Association for Computational Linguistics, Prague, Czech Republic, pp. 227–230. Available at `http://www.aclweb.org/anthology/S/S07/S07-1048`.

Hans Kamp & Uwe Reyle (1993): *From discourse to logic*. 1,2, Kluwer.

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

Gottfried Wilhelm Leibniz (1995): *Philosophical Writings*. Everyman.

Yuliya Lierler, Daniela Inclezan & Michael Gelfond (2017): *Action Languages and Question Answering*. In: IWCS 2017 - 12th International Conference on Computational Semantics - Short papers.

Gang Ling (2018): *From Narrative Text to VerbNet-Based DRSes: System Text2DRS*. Project Report, `https://www.unomaha.edu/college-of-information-science-and-technology/natural-language-processing-and-knowledge-representation-lab/_files/papers/Text2Drses_system_description.pdf`.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard & David McClosky (2014): *The Stanford CoreNLP Natural Language Processing Toolkit*. Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55–60, . Available at `http://aclweb.org/anthology/P14-5010`.

Bhavana Dalvi Mishra, Lifu Huang, Niket Tandon, Wen-tau Yih & Peter Clark (2018): *Tracking State Changes in Procedural Text: A Challenge Dataset and Models for Process Paragraph Comprehension*. *CoRR* abs/1805.06975. Available at `http://arxiv.org/abs/1805.06975`.

Arindam Mitra & Chitta Baral (2016): *Addressing a Question Answering Challenge by Combining Statistical Methods with Inductive Rule Learning and Reasoning*, pp. 2779–2785.

Craig Olson (2019): *Processing Narratives by Means of Action Languages*. Master's thesis, University of Nebraska Omaha.

Martha Palmer (2018): *VerbNet*. `https://verbs.colorado.edu/verb-index/vn3.3/`.

Martha Palmer, Daniel Gildea & Paul Kingsbury (2005): *The Proposition Bank: An Annotated Corpus of Semantic Roles*. Computational Linguistics 31(1), pp. 71–106, . Available at `http://dx.doi.org/10.1162/0891201053630264`.

Edward Wertz, Anuradha Chandrasekan & Yuanlin Zhang (2018): *CALM: a Compiler for Modular Action Language ALM*. unpublished draft.

Jason Weston, Antoine Bordes, Sumit Chopra & Tomas Mikolov (2015): *Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks*. *CoRR* abs/1502.05698. Available at `http://arxiv.org/abs/1502.05698`.