RECOGNIZING THE EFFECTS CAUSED BY AN ACTION IN A DECLARATIVE SENTENCE

E. Ahn, W. Faris[1], and K.H. Cheng

Department of Computer Science
University of Houston
Houston, TX, 77204, USA
http://www.cs.uh.edu

UH-CS-09-02

February 18, 2009

Keywords: Action, Natural Language Processing, Artificial Intelligence, Effect, Grammar.

Abstract

An important requirement for any artificial intelligence program is its capability to understand what has been communicated to it. How does the program recognize all the potential effects when it is given a declarative sentence involving an action? There are three major bodies of knowledge that need to be learned. The first body of knowledge is the English grammar. The second body of knowledge involves each individual action. The last body of knowledge acts as a bridge between the English grammar and actions. This knowledge teaches the program how to use the learned grammar to express a thought about an action. In this paper, our program uses this knowledge to understand the intention of a given declarative sentence that uses an action verb. We will discuss what is involved in the learning for each of these bodies of knowledge and illustrate how to use them to understand a given sentence. Specifically, we discuss in detail on how the effects caused by an action referenced in a declarative sentence are recognized. Understanding the intention of a sentence, especially a sentence involving cause and effects, is an important first step for a program to perform logical reasoning. Object-Oriented paradigm is used to analyze the problem and design the solution attacking the problem.

[1]W. Faris is supported in part by GAANN: Doctoral Training in Computer and Computational Sciences, US Department of Education under grant #P200A070377.

RECOGNIZING THE EFFECTS CAUSED BY AN ACTION IN A DECLARATIVE SENTENCE

E. Ahn, W. Faris[1], and K.H. Cheng

Abstract

An important requirement for any artificial intelligence program is its capability to understand what has been communicated to it. How does the program recognize all the potential effects when it is given a declarative sentence involving an action? There are three major bodies of knowledge that need to be learned. The first body of knowledge is the English grammar. The second body of knowledge involves each individual action. The last body of knowledge acts as a bridge between the English grammar and actions. This knowledge teaches the program how to use the learned grammar to express a thought about an action. In this paper, our program uses this knowledge to understand the intention of a given declarative sentence that uses an action verb. We will discuss what is involved in the learning for each of these bodies of knowledge and illustrate how to use them to understand a given sentence. Specifically, we discuss in detail on how the effects caused by an action referenced in a declarative sentence are recognized. Understanding the intention of a sentence, especially a sentence involving cause and effects, is an important first step for a program to perform logical reasoning. Object-Oriented paradigm is used to analyze the problem and design the solution attacking the problem.

Index Terms

Action, Natural Language Processing, Artificial Intelligence, Effect, Grammar.

I. INTRODUCTION

There are several major criteria for any artificial intelligence program. One of them is its ability to communicate with the external world using a natural language. Another criterion is its ability to perform reasoning. Some reasoning is about subject matters involving cause and effects, and in order to perform reasoning about those subject matters, one needs to recognize and understand those causes and effects. Knowledge about action is an example of a body of knowledge that involves cause and effect. In this paper, we investigated the problem on how to recognize the effects caused by a specific action referenced in a given sentence. We divide the problem into several sub-problems. The first sub-problem is to learn the grammar of the English language. The second sub-problem is to learn the body of knowledge about each individual action. The third sub-problem is to learn how to bridge the grammar and the actions together, so that we can recognize, and subsequently carry out, the effects caused by the action mentioned in the given declarative sentence.

Research in natural language processing has attracted a lot of attention. In [1], some of the major challenges and barriers facing researches in the field of natural language processing have been presented. Most research tries to overcome these challenges using logic based or functional programming languages such as Miranda [2] and Haskell [3]. Several functional language parsers [4-6] have been used to develop natural language interface systems such as Lolita [7] and Windsor [8]. A comprehensive survey on natural language interfaces using the lazy functional programming approach can be found in [9]. Action verb may be added as a notational extension to first order logic [10] called the lambda notation [11], and such addition is known as semantic augmentation [12]. For systems build on first order logical programming such as Prolog, these semantic augmentations represent individual function calls. For every verb introduced, a new function for that verb needs to be written into the system, and carrying out the effects of an action may be implicit to the execution of the functions. However, it is unclear how to locate the

[1]W. Faris is supported in part by GAANN: Doctoral Training in Computer and Computational Sciences, US Department of Education under grant #P200A070377.

starting point of the reasoning process where the understanding of these effects occurs. Another major problem on using functional programming approach is that additions to accommodate a new aspect of the natural language may require a major change to many existing structures [13].

Some researchers are interested in the production of parts-of-speech tags when given a sentence. A very accurate parser of English that employs syntactic analysis [14] to produce parts-of-speech tags of a given sentence has been developed based on constraint grammar [15]. Another comprehensive part-of-speech tagging system was developed by [16]. Parts-of-speech tagging may be used in many applications, e.g., structural correspondence learning is used in [17] to produce tags in a new domain from a known domain. Recently, we have developed a sub-system to learn the grammar terms (parts-of-speech) of the English language [18]. It is a part of the communication agent of the learning program in the project: A Learning Program System (ALPS) [19] whose goal is to learn all knowledge. The initial focus of ALPS has been on the development of the memory agent of a multi-agent artificial intelligence program to store knowledge and the relationships among them. Basic capabilities, such as creating a new category, adding objects, attributes, and properties to a category have been provided. We have developed two major knowledge components of categories: hierarchy [20] and definition [21]. Hierarchy may be used to specify both generalization and containment relationships among knowledge. Definition specifies the necessary and sufficient condition that is used to classify objects as a specific category. The sub-system in [18] first learns a subset of the English grammar, and then uses the grammar to parse sentences. A key idea introduced is the role of a grammar term, which defines the intention of the term. The roles of the various grammar terms in a particular sentence allow the program to understand the exact purpose of the sentence. They serve as the bridge between the grammar knowledge world and the knowledge world that the sentence is trying to express. For our parsing algorithm, an appropriate role has been correctly identified for every part of the sentence. However, in order to fully understand the details of the sentence so as to carry out its intention, the detailed content of each individual role needs to be organized in a meaningful way. We call this organization effort by a role a satisfaction of the role. In this paper, we describe in detail how knowledge common to all kinds of actions are learned, including the effects of each individual action. These effects are used to produce the actual effects implied by a given sentence. These actual effects may be used in the future by the logic and reasoning agent of the ALPS system to perform reasoning. Then, we explain how the control in a declaration chooses the correct role object to carry out the satisfaction process given a declarative sentence that uses either a forms-of-be verb or an action verb. We also describe the necessary knowledge needed by act role, the role of the grammar term action verb, to prepare a thought from a declarative sentence that uses an action verb. This learned knowledge serves as a bridge between two bodies of knowledge: grammar and action. Finally, we describe how to understand the resulting thought, i.e., recognizing the effects of the action on the appropriate objects. These effects are then carried out by simply using the existing function calls of the learning program. The knowledge required for the understanding and execution of a declarative sentence that uses a forms-of-be verb can be found in [22].

The rest of the paper is organized as follows. Section 2 gives a brief description of the English grammar. Section 3 describes how we learn the body of knowledge on various actions. Section 4 describes what needs to be learned by act role and shows how it prepares a thought from a given sentence. Section 5 describes how the effects of a declarative sentence involving a specific action can be recognized and produced by the system. Section 6 discusses some issues that we encountered and potential future work and concludes the paper.

II. The English Grammar

In our program, the learning of the English grammar is done in an incremental manner. The learned grammar is then used to parse and understand English sentences. In the future, it will use the learned grammar to generate English sentences. Our program first learns the various grammar terms in English: such as sentence, complete subject, verb, noun phrase, and preposition. The four major components for a grammar term introduced in [18] are structure, role, kind, and rule. Not all components are required for every grammar term, which means that a specific grammar term may be defined by a combination of some of these components. The structure of a grammar term defines exactly the grammatical format of the term and the type of knowledge that it contains. The format may be either a sequence or an alternative, and the type of knowledge may be grammar term or the different kinds of knowledge known by the learning program such as action, category, concept, and data type. The role of a grammar term defines the intention of the term. The roles played by the various grammar terms in a sentence allow the program to understand the exact purpose of the sentence. They serve as the bridge between the grammar knowledge world and the knowledge world that the sentence is trying to express. A term may have multiple kinds,

which are subsets that must have different roles but may share the same structure. For example, an interrogative sentence is one kind of the sentence grammar term. Finally, a rule specifies a condition that must be satisfied. Rules may be applied directly to the grammar term or to one of its structures or kinds. Sufficient condition allows us to choose the correct term or kind. However, necessary conditions allow us to eliminate exceptions especially when perfect sufficient conditions are normally not available.

Our parsing algorithm consists of a syntactic stage followed by a semantic stage. The syntactic stage deals with the analysis of the individual words in the sentence. The syntactic stage in [18] simply stems words into its root form and passes the re-constructed sentence to the semantic stage. Substantial extensions on morphology and multiple word terms have just been added in the syntactic stage [23]. The semantic stage deals with recognizing all the subparts of a given sentence, identifying the knowledge referred to in that sentence and producing an appropriate thought associated with the sentence. The semantic stage is based on a high-level template parser that makes use of the individual structures' unique internal parsers. It is a depth-first top-down parser whose execution consists of processing in two major manners: top-down and bottom-up. The top-down processing starts parsing using the structure of the highest grammar term, the sentence, and works its way down to lower level terms. It is responsible for recognizing the terms of the sentence, and identifying the knowledge mentioned in the sentence. After recognizing a term of the sentence by the knowledge involved, the algorithm verifies that the appropriate rules unique to the grammar term or its structure are satisfied. Based on the recognized grammar term with all its identified sub-roles, the top-down parsing process finishes by choosing the associated role of the term and including that role in the parse result. The role identifies the high-level purpose of the grammar term. However, the content of the role needs to be organized in order to understand the detailed purpose of the term. The processing in the bottom-up manner satisfies the role by organizing all the identified sub-roles appropriately, and the resulting role is returned to a higher-level term.

There are two possible cases in satisfying a role. First, when parsing a grammar term that has an alternative structure, since the parsing is completed by only one alternative, the grammar term can have at most one satisfied sub-role. If the alternative is at the lowest level, the role is satisfied by the knowledge identified in the sentence, e.g., the knowledge representing *John Smith* or *buy*. Otherwise, this role can easily be satisfied by the knowledge stored in the role of the involved alternative. On the other hand, after parsing a grammar term that has a sequence structure, the role of each term in the sequence has been satisfied, so there are several sub-roles available to satisfy the role associated with the grammar term. There are two tasks involved. The first task is to identify the correct sub-role to carry out the satisfaction process. We introduce the idea of control to identify the correct sub-role to carry out the duty of satisfying a role. The second task is on how to carry out the process once the correct role has been identified. It has to prepare its content in such a way that the intention can then be understood and executed easily by the appropriate knowledge object. By the time the sentence grammar term finishes parsing, the result is a thought that reflects what the sentence wants to express, and the identified knowledge is stored as the appropriate parts of this thought.

We describe in this paragraph the grammar knowledge of the English language necessary to express an action in a sentence. Some grammar terms involved in the action must be present in a given sentence in order for the sentence to be complete, while some grammar terms may be optional. These structural requirements are common to all actions and are learned as part of the grammar. The noun phrase corresponding to the compulsory grammar term complete subject is learned to assume the subject role. If an English sentence uses an action verb in a sentence, the complement must be an object complement. The structure of an object complement is composed of a sequence of an optional noun phrase followed by a compulsory noun phrase. The compulsory noun phrase is learned to assume the role of direct object, while the optional noun phrase has the role of indirect object, if given in a sentence. In addition, for the purpose of understanding this paper, we would want to point out the following learned knowledge of the English language. The grammar term declarative sentence is learned to have the declaration role, which is one kind of thought. The control for the declaration role is the verb role. The grammar term verb has the role of verb, and one of its alternative structures is the main verb. Two alternatives of main verb are forms-of-be verb (an alternative of linking verb) and action verb. The action verb is learned to use the action knowledge during parsing to recognize any learned action name in a given sentence. The role for the grammar term action verb is the act role, while the role for the forms-of-be verb is the define role.

III. The Action Knowledge

The knowledge required by every action includes its name, various objects involved in the action, and the effects of the action on these objects. The objects involved in an action include the actor, the act-on object, the act-for (beneficiary) object, and any other objects that may be introduced by a preposition in a sentence. The names of these objects show their logical meanings with respect to the involved action. These logical names will later be used to introduce the effects of the action, and may be different for different actions. For each action object, our program first learns about the action name, and the logical names for the compulsory actor and the act-on object. If the specific action allows the optional act-for object in a sentence, then its logical name is learned. On the other hand, for an action that does not allow an act-for object in a sentence, this fact is communicated to the program using the special keyword 'none'. Next, a sequence of possible prepositions, each followed by an appropriate logical object name, is learned. The actual knowledge objects corresponding to these logical objects are introduced in a sentence by optional prepositional phrases. For example, for the action with the name *give*, the actor may be taught as the *giver* and the act-on object is taught as *possession*. The optional act-for object is taught as the *receiver*, which can be explicitly given in the sentence or introduced by the preposition 'to'. On the other hand, for the action with the name *receive*, the actor is taught as the *receiver*, the act-on object is taught as *possession*, the act-for object as 'none', since it cannot be explicitly given in a sentence, and a *giver* object is introduced by the preposition 'from'.

Note that objects such as *receiver* and *giver* are knowledge local to the action, but the act-on object of *possession* is not a knowledge object local to either the action *give* or *receive*. It is a knowledge known by the ALPS program and it represents any knowledge object that can be possessed. This knowledge, *possession*, has to be taught to the ALPS program first, and can be set up either by a hierarchy [20] or as a definition [21]. For simplicity, we have currently set this up as a hierarchy. For example, *apple*, *orange*, *house* and *car* may all be taught as children categories of *possession*. A more important observation of the act-on object is that the actual action and the corresponding effects are different for different categories of act-on object, even with the same action name. Take the action *send* as an example, if the act-on object is a *communication-object* such as an *email* or a *letter*, the list of effects would be "receiver increase communication-object" and "receiver know message", where message is what the *communication-object* is meant for (we assume the receiver will read the email or letter). On the other hand, if the act-on object is *human*, the effect would be that the location of the *human* is changed. As another example, consider the action *receive*, the effects are the same as those of action *send* if the act-on object is a *communication-object*. However if the act-on object is *possession*, the list of effects is different, and the effects are "giver decrease possession" and "receiver increase possession". As a result, each action contains a list of alternative internal action objects; each is about a different category of act-on object with its own list of logical objects and effects. Figure 1 shows the object of the action *receive* together with its list of internal action objects, assuming that it has two act-on objects: *possession* and *communication-object*. Note that objects that are taught with the keyword 'none' represent objects that cannot be explicitly given in a sentence. As a result there is no need to store these objects. We use a dashed border to represent this fact logically in our figures.

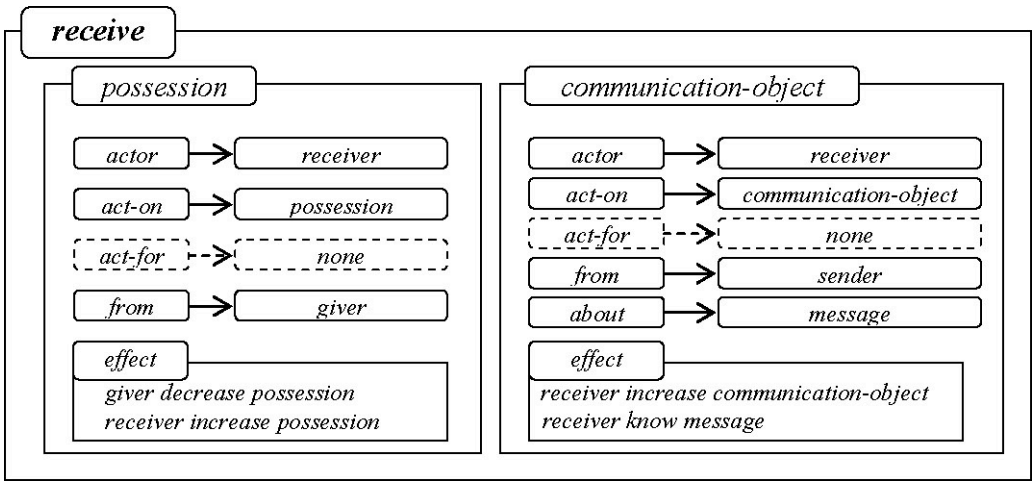
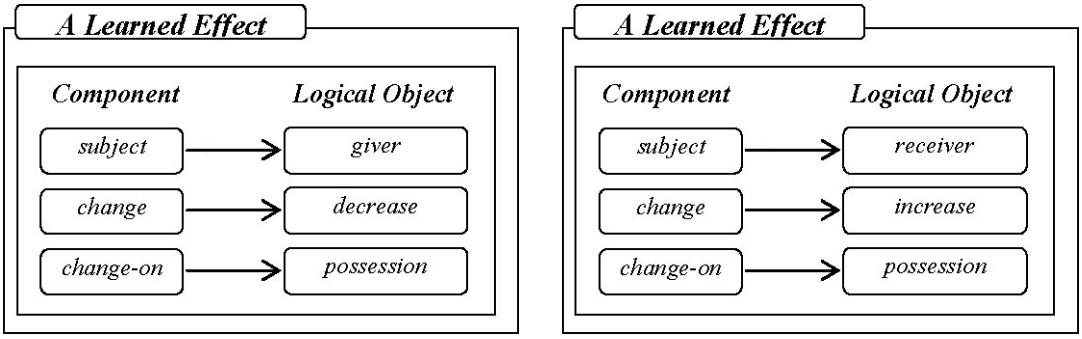


Fig. 1. The action knowledge: *receive*.

Currently, we use two different mechanisms to teach the list of effects of an action. The first mechanism is used by a subclass of action called simple action. Simple actions are those actions that are fundamental, i.e., each simple action consists of no other actions. Examples of simple actions are *give* and *receive*. For simple action, it uses a special interface to learn the effects about the specific action. It simply asks for the three key components for each effect: subject, change, and change-on object. The subject is the knowledge object experiencing the effect. The change is the result of the effect, e.g., an *increase* or a *decrease*. The change-on object is the category of the object related to the subject that has been changed. For example, one of the effects learned for the action *give* on the act-on object *possession* is "giver decrease possession", where *giver*, *decrease*, and *possession* are the subject, the change, and the change-on object of the effect, respectively. As a result, given the sentence "John gives 2 apples to Jack." and based on the above learned effect, the program will understand that the possession of John will decrease by two apples. Similarly, the effect that the possession of Jack increases by two apples can also be inferred based on the other learned effect of the same action. Figure 2 shows how the two effects of the simple action *give* for the act-on object *possession* are stored. These are the same two effects of action *receive* for the act-on object *possession* where they have been shown logically in Figure 1.



(a) Giver decrease possession (b) Receiver increase possession

Fig. 2. Two effects of the simple action *give* for the act-on object possession.

The second mechanism to learn an effect is used by a different subclass of action called composite action. Each composite action involves a sequence of actions. This sequence of actions is taught as the effect list for this action using the logical objects introduced in the action. The learning of the logical objects and how they can appear in a sentence is exactly the same as in learning a simple action. However, the major difference is that the list of effects is learned as a sequence of English sentences, and hence handled differently. Take learning about the action *buy* as an example, the actor object is identified as the *buyer*, the act-on object as *possession*, the act-for object as the *receiver*, the *seller* object is introduced by the preposition 'from', and the *money* object is introduced by the preposition 'for'. The list of effects is a sequence of three declarative sentences each involving an action. The first effect is "The buyer gives the money to the seller." The second effect is "The buyer receives the possession from the seller.", and the last effect is "The buyer gives the receiver the possession." After learning each effect sentence, we parse the effect sentence and store the parsed result, which we call a final declaration thought. Notice that an effect sentence may use both local logical knowledge and global knowledge objects. Our parsing algorithm recognizes knowledge by searching for local objects before trying to find any global knowledge objects. Figure 3 shows the action *buy* with one internal action object for act-on object *possession*, where each effect is shown logically. Figure 4 gives the final declaration thought for the parsed effect: "The buyer receives possession from the seller." When given an English sentence using *buy*, these parsed effects are used in our program to produce declarative thoughts and then sentences about the actual effects. For example, based on the learned effects of *buy*, the sentence "John buys Jack 2 apples from Mary for 4 dollars." will result initially with three English sentences as its effects. They are "John gives Mary 4 dollars.", "John receives 2 apples from Mary.", and "John gives 2 apples to Jack." We will describe how these effects are produced in Section 5. By parsing, understanding, and executing these three effect sentences, a final sequence of six effects results from understanding the original sentence. They are "John decrease 4 dollars", "Mary increase 4 dollars", "Mary decrease 2 apples", "John increase 2 apples", "John decrease 2 apples", and "Jack increase 2 apples".

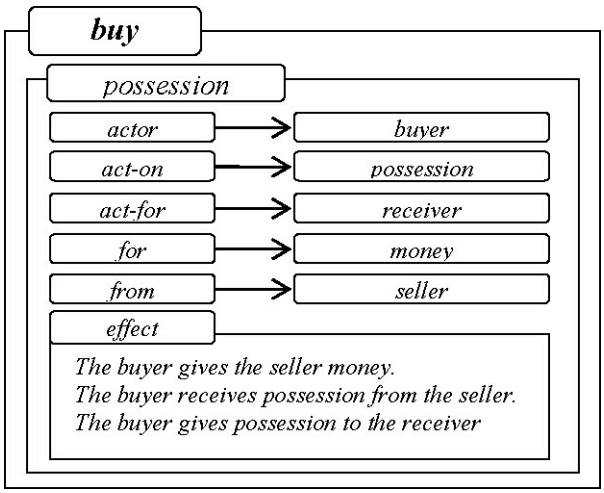


Fig. 3. The action knowledge: *buy*.

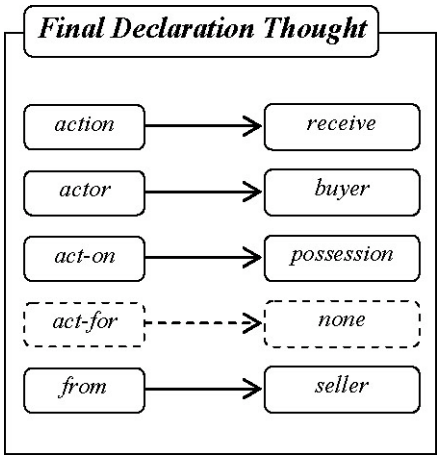


Fig. 4. The parsed effect: "The buyer receives possession from the seller."

IV. The Act Role

The role of a grammar term is to define the intention of the term. Each role is a knowledge object responsible for linking a portion of the grammar with the knowledge that it is expressing. Specifically, the act role serves as the bridge between action verbs and the knowledge about actions. It contains knowledge on how to express an action, and consequently has been used in this paper to understand a correct sentence. The knowledge associated with the act role is learned in the following sequence of steps. First, act role is learned to make use of the knowledge action. Once this is known, act role can request from action a list of knowledge that is common to every action, which includes the actor, the act-on object, and an optional act-for object. Lastly, it finds out which role in the grammar corresponds to each of those knowledge objects in the action world. Specifically, the actor is learned to be associated with the subject role in the grammar, while the act-on and act-for objects are associated with the direct object and the indirect object roles, respectively. Once these are learned, both indices and their inverted indices are set up. The indices link the grammar to the action knowledge, while the inverted indices link in the opposite direction. For example when given subject, the index is used to identify actor. On the other hand, the inverted index will allow us to locate subject when given actor. Figure 5 shows the relationships that have been learned and set up in the act role.

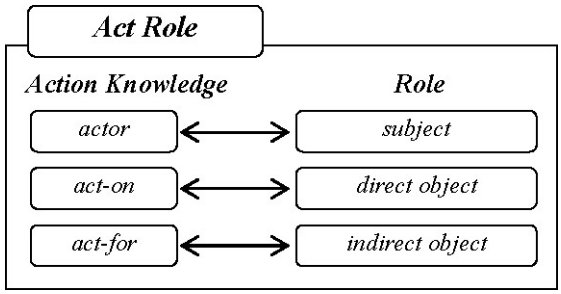
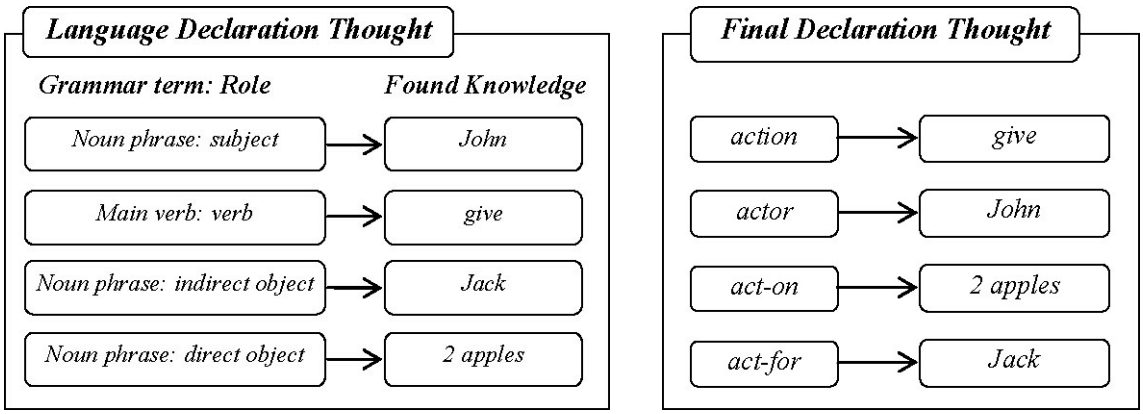


Fig. 5. The learned bridge knowledge of the act role.

Assume that all the above three major bodies of knowledge have been learned, we will first present briefly how the act role is chosen if the given declarative sentence uses an action verb. Then we will describe how the act role prepares the final declaration thought, which will be used by the action object identified in the given sentence. Finally, in the next section, we will describe how the action object recognizes and creates the list of effects. Now, when our system is given an English sentence, the parsing algorithm will use the learned grammar to parse the sentence. For each successfully parsed grammar term or structure, its associated role will be identified and called to satisfy the detail contents. For example, suppose our given sentence is a declarative sentence ended by the punctuation '.'. By the time it is decided to be a declarative sentence, its associated role, a declaration thought is

called to satisfy, and we call this a language declaration thought. Since there are several roles in the declarative sentence such as the subject and the verb roles, the control is needed to decide which one to use. Recall that the control for a declaration is the verb role, so our program uses the verb role to carry out the satisfaction process. If the English sentence uses the form-of-be verb, the verb role identified by the parsing is the define role, therefore the define role is chosen. This scenario can be found in [22], along with details on the solution for other role objects found in a sentence. Of particular relevance to this paper is about how noun phrases associate knowledge referenced in the sentence with roles such as subject and direct object. In addition, knowledge objects associated with prepositions and the constant associated with the determinate are also handled by noun phrase.

Now, suppose our given sentence uses an action verb, the identified verb role is the act role, so the act role is chosen to prepare the final declaration thought. Since parsing is done using grammar knowledge, knowledge objects referred to in the sentence are recognized by the grammar terms, and consequently are associated with the corresponding roles such as the subject and the direct object in the language declaration thought. Figure 6a shows the content for some of the major roles for the parsed sentence: "John gives Jack 2 apples." Since act role has learned that it uses the knowledge action, it knows that the specified action object will eventually be used to understand its effect. But every action needs to know about the actor, the act-on object, etc., so the act role will prepare the final declaration thought to contain knowledge indexed by those terms. The relationships learned by the act role can now be used. For example, to add the actor object, we first find that it corresponds to the subject, and then search the language declaration thought for the knowledge having the subject role. Once the knowledge object corresponding to the subject role has been located, it can be stored in the final declaration thought under the index actor. The same can be done for the act-on, and the act-for objects. Figure 6b shows how knowledge parsed from the given sentence is indexed in this final declaration thought. Note that this has the same indexing scheme as in Figure 4, which is the parse result for an effect given by a declarative sentence using internal knowledge objects with logical names. Figure 7 shows the simplified high-level process on how a given declarative sentence that uses an action verb is parsed to produce this final declaration thought. In our figures, objects used and created by other objects appear in ellipses. Grammar knowledge objects first parse a given English sentence to produce the language declaration thought, which in turn will be used to create the final declaration thought by act role. The completely satisfied final declaration thought is the result of the parsing algorithm, and knowledge referred to in the sentence has been indexed based on the action knowledge. Now the understanding and the execution of this final declaration thought can be carried out.



(a) Language Declaration Thought (b) Final Declaration Thought

Fig. 6. The two declaration thoughts for the sentence: "John gives Jack 2 apples."

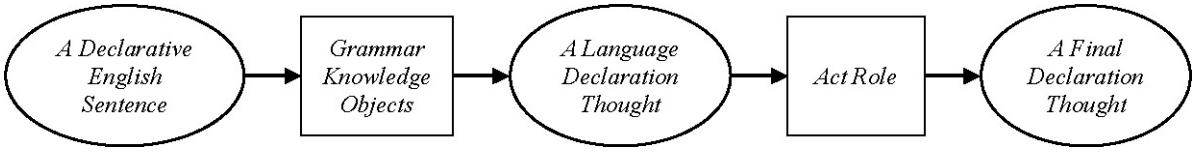


Fig. 7. The process of parsing a given declarative sentence that uses an action verb.

V. The Understanding of a Sentence using an Action Verb

Given the final declaration thought, the act role will once again be chosen based on the knowledge that verb is the control of a declarative sentence. The knowledge that act role uses action leads to the action object stored in the final declaration thought. Since the action object may be used on multiple categories of act-on objects, our program has to find out which category the actual act-on object belongs to in order to locate the correct internal action object. Once the correct internal action is found, it is used to produce all the correct effects of that action. First, a link object is created linking each logical object of the action with the actual knowledge objects from the declaration thought. Using this linked information and the learned effects of the action, the correct sequence of final effects can now be produced one effect at a time. Figure 8 shows the process of creating a final effect given a final declaration thought for a parsed English sentence.

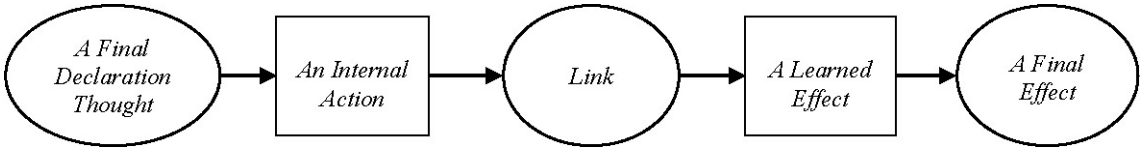


Fig. 8. The process of creating a final effect.

First, we describe the process of setting up the link, which is the same for both simple and composite actions. Recall that effects are learned using local logical names, so for each logical object in the effect, our solution has to find out the corresponding knowledge in the given sentence. Suppose the logical object corresponds to the actor of the action, we can use the term actor to identify the actual knowledge stored inside the final declaration thought. This is easily done since the final thought created by act role is indexed by action knowledge names such as actor and act-on (Figure 6b). The same mechanism is used to find the actual act-on object, the act-for object, and other objects associated with the various prepositions. The link establishes the relationships between the knowledge objects from *action* and those from the final declaration thought. For example, given the sentence "John gives an apple to Jack." , the created link matches *giver* with *John*, *possession* with *1 apple*, and *receiver* with *Jack*. After the link has been created, the final effect can now be formulated identifying the components of a learned effect by the actual knowledge in the given sentence. The process of creating the final effect is different for effects of simple and composite actions. For simple actions, this is relatively straightforward using the established link. Continuing with the above example and the learned effect "giver decrease possession", the final effect contains *John* as the subject, *decrease* as the change, and *1 apple* as the change-on object. Figure 9 shows the detailed contents of the involved objects in the process. Finally, these final effects are carried out, when needed, by simply using the existing function calls of the learning program.

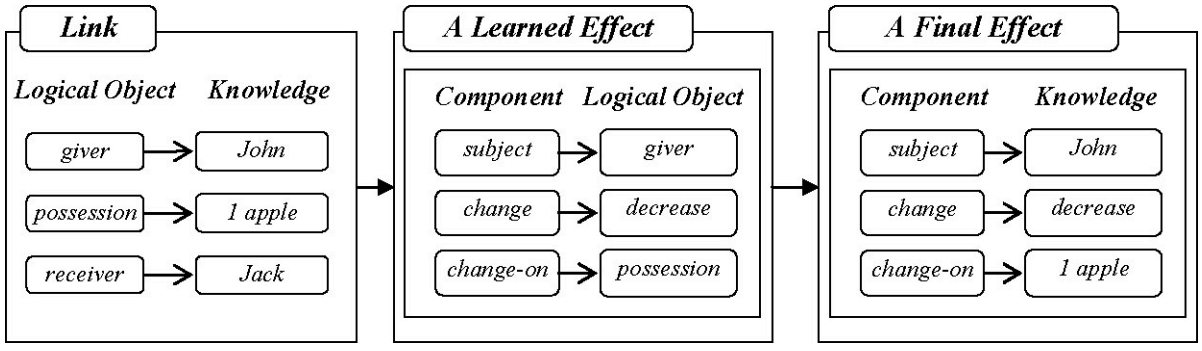


Fig. 9. The detailed process of producing a final effect from the link.

For effects learned for a composite action, recall that each effect is learned as an English sentence expressed in terms of local logical objects' names and a learned action, and the stored effect is the parsed result of the effect sentence. We call this learned effect the effect thought. Whereas, we call the parse result of the given sentence as the sentence thought. Our final goal is to produce a final effect as an English sentence. We divide the process into two steps. The first step is to use the effect thought as a template to produce a final effect thought substituting local logical objects by objects obtained from the sentence thought. The second step is to use the learned grammar to

produce the final effect sentence given the final effect thought. At the time an effect thought is asked to produce the final effect thought in the first step, since it is a declaration thought, it will create a declaration thought. Once again, since the control of a declaration is its verb and it is an action, act role is used. Now, using the same link information and the same process as described earlier, the actual knowledge object of the sentence thought can be located and indexed by action knowledge such as actor and act-on. However, since the final effect thought is going to be used by the grammar knowledge to produce an English sentence, the final effect thought has to be prepared in such a manner that knowledge objects are indexed by the language terms such as subject and direct object instead of action terms. By using the inverted index set up in the act role, we can store the actual knowledge of the given sentence using subject instead of actor as its index. Here, we have illustrated how the act role acts as a bridge in the opposite direction to create the final effect thought. The rest of the compulsory language terms in a sentence, such as direct object, can be put into the final effect thought in a similar manner. Figure 10 shows this process on one of the effects, "the buyer gives the seller money", of the action *buy* using the given sentence "John buys Mary 2 apples from Jack for 5 dollars." As for the second step, we are currently working on composing an English sentence given this final effect thought using the learned grammar. In the meantime, since the effects are completely understood, we produce these final effect sentences in a brute-force manner. Our program then parses these final effect sentences and executes them. Notice that if all objects involved in the effect thought were provided in the given sentence, they can all be created successfully. However, it is quite possible that certain logical objects are not included in the given sentence because they are optional such as objects associated with prepositional phrases. Our current solution will use a temporary object called unknown and compose the final effects using these objects. Since sentences without these optional objects are legal sentences, if an effect involves some unknown object, its effect is assumed to be fine, but the execution of that effect is skipped. For example, in the sentence "John buys a car.", one of the effect sentences is "John receives a car from unknown.", and this effect in turn will create two effects "John increase 1 car" and "unknown decrease 1 car". Only the first effect can be carried out, but the second effect cannot be carried out because it only indicates logically that the possession of an unknown object decreases by one car. In the future, if a context has already been established, some unknowns may be resolved by using this context.

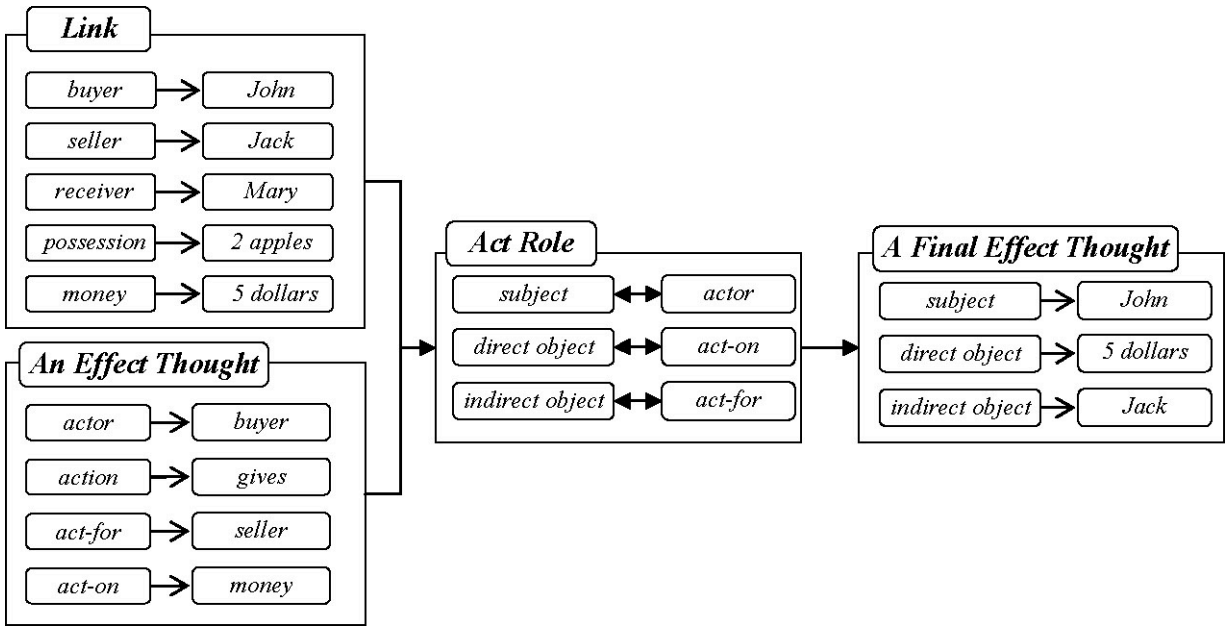


Fig. 10. An example of producing a final effect thought.

VI. Discussion and Conclusion

There are several problems involving knowledge objects introduced by prepositions. The first two problems involve the indirect object, and we handle them differently. The first of these problems is when an indirect object is mistakenly used when it is not allowed for a specific action verb, as with the sentence "He received a friend a package." The system can take one of three alternatives: reject the sentence as an error, infer the indirect object as

some object introduced by a preposition, or ignore the error but accept the rest of the sentence. The first alternative of rejecting the sentence seems to be an extreme reaction to a minor problem. The second alternative is to try to correct the error. The system could infer that the indirect object is reflective of the 'from' object, i.e., the original intention of the sentence may be "He received a package from a friend." However, inferring a preposition can lead to unintended results. For instance, the original intention could be that the package is for a friend rather than from a friend. As a result, our system takes the third alternative, which is to ignore the extra complement but still accept the rest of the sentence and give a warning message. The second problem occurs when an action object is reflected in both the indirect object and in a prepositional phrase, resulting in a direct conflict. For example, given the sentence "John gave Jack 3 apples to Mary", we have a conflict when trying to understand the statement. It is unclear to the program which object, *Jack* or *Mary*, is the act-for object to receive the possession. We consider this a more serious problem, and so we treat it as an error and reject the given sentence. The next problem is that some prepositions may be used for multiple purposes, and that both may be presented in the same sentence. Currently, actions are taught to associate one classification per preposition. For example, the preposition 'for' is taught to precede a knowledge object of the category money, as in "John buys a book for 10 dollars." However, this would need to be adjusted so that an action can recognize multiple classifications for the same preposition and learn how to handle them in the future. For instance, if the previous sentence is replaced by "John buys a book for Mary.", the intention is no longer the price paid, but rather, the receiver of the purchase. In addition, both forms of the preposition can be used in the same sentence: "John buys a book for Mary for 10 dollars." Furthermore, there are a whole host of other problems that we will have to deal with in the future such as compound subjects, compound predicates, pronouns, and gerunds.

Understanding the intention of a sentence, especially a sentence involving cause and effects, is an important first step to perform logical reasoning. Currently, we are working on testing the pre-conditions in order for the effects of an action to be carried out successfully. For the effects of an action, if the pre-conditions of an effect are not satisfied, the effect cannot be realized. For example, a person cannot give away a house unless he/she owns that house. On the other hand, some pre-conditions are considered inconsequential. An example of this would be a person giving away a piece of paper, which can be considered inexpensive and that the giver should normally have enough to give out. The problem of satisfying pre-conditions may also happen when a sequence of effects of an action has dependency requirements, such as "Jack bought his son a Lamborghini from a dealer for half a million dollars." Before Jack can give his son the car, he must give the money to the dealer. In order to do that, he must have enough money or credit. So if he cannot afford it, the effect that the dealer receives the money fails, and thus Jack will be unable to give his son the car. In situations like this, an effect cannot be carried out if a prior effect has failed. Another problem that may involve reasoning is to recognize the equivalence of two actions. For pairs of actions such as give and receive, or buy and sell, each pair of actions is actually the same action when both are acting on the same category of objects (such as possession). Both action names may be used to express the same intention except that the roles of some objects are interchanged. For example, the sentence "John gave Mary a book." is the same as "Mary received a book from John." The reasoning involved in how to recognize equivalent actions requires a lot of other knowledge and involves the development of the logic and reasoning capability of the learning program. The reasoning capability of the learning program using cause-and-effect knowledge will be the next major focus of the ALPS project.

In this paper, we have illustrated what needs to be learned so that the knowledge learned in English grammar can be used to express a thought about an action. We have shown how to use this bridge knowledge to prepare an internal thought when given a declarative sentence that uses an action verb. In addition, we have discussed some specific sub-components of the action knowledge, distinguishing their meaning when they are used for different categories of act-on objects, and described how they were used in formulating the final effects of the action referenced in a given sentence. Our solution demonstrates a clear separation of the three bodies of knowledge: grammar, action, and the bridge knowledge, and why each are required and how they are related to one another. Specifically, the bridge knowledge is on how to express a thought in a sentence, and it is used in this paper to understand a sentence by producing its intended thought.

REFERENCES

[1] R. Weischedel, J. Carbonell, B. Grosz, W. Lehnert, M. Marcus, R. Perrault, and R. Wilensky, "White paper on natural language processing," in *Proc. ACL Human Language Technology Conference*, 1989, pp. 481-493.

[2] D.A. Turner, "A new implementation technique for applicative languages," *Softw. Pract. Exper.* Vol 9, no. 1, pp. 31-49, 1979.

[3] P. Hudak, S.L. Peyton-Jones, P. Wadler, B. Boutel, J. Fairbairn, J.H. Fasel, M.M. Guzman, K. Hammond, J. Hughes, T. Johnsson, R.B. Kierburtz, R.S. Nikhil, W. Partain, and J. Peterson, "Report on the programming language Haskell, a non strict, purely functional language," *SIGPLAN*, vol. 27, no. 5, R1-R164, 1992.
[4] R. Leermakers, The Functional Treatment of Parsing, International Series in Engineering and Computer Science, Kluwar Academic Publishers, 1993.
[5] P. Ljunglof, "Functional pearls: Functional chart parsing of context-free grammars functional pearl," *J. Funct. Program*, vol. 14, no. 6, pp. 669-680, 2004.
[6] P.C. Callaghan, Generalized LR parsing, The Happy User Guide, Chap. 3, Simon Marlow, 2005.
[7] R. Garigliano, R. Morgan, and M.Smith, "The LOLITA system as a contents scanning tool," in *Proc. 13th International Conference on Artificial Intelligence, Expert Systems and Natural Language Processing*, Avignon, France, 1993.
[8] R.A. Frost, "W/AGE the Windsor attribute grammar programming environment," in *Proc. IEEE Symposia on Human Centric Computing Languages and Environments*, 2002, pp. 96-99.
[9] R.A. Frost, "Realization of Natural Language Interfaces Using Lazy Functional Programming," *ACM Computing Surveys*, vol. 38, no. 4, article 11, Dec. 2006.
[10] S. Russell and P. Norvig, Artificial Intelligence, A Modern Approach, 2nd Ed., Prentice Hall, 2003.
[11] A. Church, "A formulation of a simple theory of types," *Journal of Symbolic Logic*, vol. 5, pp. 56-68, 1940.
[12] D. Jurafsky and J. H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Prentice Hall, 2000.
[13] C. Shan, "Monads for natural language semantics," in *Proc. 13th European Summer School in Logic, Language and Information*, Student Session, K. Striegnitz, Ed., Helsinki, Finland, 2001, pp. 285-298.
[14] A. Voutilainen, "A syntax-based part of speech analyzer," in *Proc. Seventh Conference of the European Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, Dublin, 1995, pp. 157-164.
[15] F. Karlsson, A. Voutilainen, J. Heikkila, and A. Anttila (eds.), Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text, Mouton de Gruyter, Berlin and New York, 1995.
[16] M. Marcus, B. Santorini, and M. Marcinkiewicz, "Building a Large Natural Language Corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313-330, 1993.
[17] J. Blitzer, R. McDonald, and F. Pereira, "Domain Adaptation with Structural Correspondence Learning, Department of Computer and Information Science," U. Penn., Technical Report, 2007.
[18] W. Faris and K. Cheng, "An Object-Oriented Approach in Representing the English Grammar and Parsing," in *Proc. International Conference on Artificial Intelligence*, 2008, pp. 325-331.
[19] K. Cheng, "An Object-Oriented Approach to Machine Learning," in *Proc. WSES International Conference on Artificial Intelligence*, 2000, pp. 487-492.
[20] K. Cheng, "The Representation and Inferences of Hierarchies," in *Proc. IASTED International Conference on Advances in Computer Science and Technology*, 2006, pp. 269-273.
[21] K. Cheng, "Representing Definitions and Its Associated Knowledge in a Learning Program," in *Proc. International Conference on Artificial Intelligence*, 2007, pp. 71-77.
[22] Faris and K. Cheng, "Understanding and Executing a Declarative Sentence involving a forms-of-be verb," technical report #UH-CS-09-03, Computer Science Dept., University of Houston, February 2009.
[23] W. Faris and K. Cheng, "An Improved Syntactic Stage of a Parsing Algorithm," in preparation.