Title

A Model of Syntactic Parsing Based on Domain-General Cognitive Mechanisms

Authors

Cassimatis, Nicholas
Murugesan, Arthi

Peer reviewed

A Model of Syntactic Parsing Based on Domain-General Cognitive Mechanisms

Arthi Murugesan (muruga@rpi.edu)
Nicholas Cassimatis (cassin@rpi.edu)
Rensselaer Polytechnic Institute
Department of Cognitive Science, 110 8th Street
Troy, NY 12180 USA

Abstract

The relationship between linguistic and nonlinguistic cognition is an important area of study including the questions of language modularity and learnability. We believe that insights to this relationship can be obtained by implementing a precise computational model of sentence understanding within a general cognitive architecture. In this paper we have represented a wide-coverage grammar (Head-driven Phrase Structure Grammar) using non-linguistic representation. We have shown how grammar-specific representations like specifier, complement, modifier and gap map to domain-general event representations such as subgoals and temporal constraints. The paper thus demonstrates that domain-general cognitive mechanisms are sufficient and adequate for syntactic parsing.

Keywords: Syntactic Parsing; domain-general mechanisms; HPSG; Polyscheme; modularity of language.

Introduction

The relationship between linguistic and nonlinguistic cognition is an important question in cognitive science. One way of studying this topic is to ask if language is an independent module or integrated with the rest of cognition. Both approaches have problems explaining the phenomenon.

The modularity of language has been argued for by Fodor (1983) and Chomsky (1988). Both in computational linguistics and in cognitive modeling of sentence processing, modular approaches have trouble explaining the interaction of people's syntax-specific processing mechanisms and general inference mechanisms about the world. A number of sentences that we parse in daily life require visual information, social awareness, higher reasoning and other complementary cognitive processes to obtain the subtle meanings, for example:

I saw a star with a telescope.
I saw an astronomer with a telescope.

In these sentences, the telescope could be the instrument used to view the object (star or astronomer) or the telescope could be seen near (with) the object. However, typically people infer the star to be viewed through the telescope and the astronomer to be near the telescope. Disambiguation can be attributed to both the visual perception of the telescope and the reasoning that an astronomer uses the instrument frequently while watching an astronomer through the telescope is improbable. This reasoning also requires knowledge about astronomers, their instruments and the action of

seeing. Therefore, the mechanisms used to understand a seemingly simple sentence include language, vision, reasoning and semantic knowledge. The modular theory doesn't address the question raised by this phenomenon: if the mechanisms of language and other kinds of inferences are so different, how do they interact so tightly and so well?

On the other hand, the non-modular approach claims that language is a part of and interacts freely with the greater cognitive system (Newell & Simon 1972; Marslen-Wilson and Tyler, 1989). The striking problem here, perhaps most visible in syntax, is that concepts involved in language (e.g., argument structure, head projection or agreement) and concepts involved in physical reasoning (e.g., gravity, time or place) seem to be very dissimilar and unrelated. How could we use, for example, the same mechanism to solve an equation, plan a shortest route to a restaurant and determine the binding properties of a given pronoun? The lack of a detailed answer to these questions makes the modularity hypothesis more plausible.

Cassimatis (2004) addressed this disparity by showing that aspects of language syntax such as categories, word order and constituency can be mapped onto domain-general cognitive structures. Although this work demonstrated that broad linguistic concepts have duals in domain-general cognition, it did not demonstrate that these high-level dualities were sufficient to map a sophisticated grammatical theory onto domain-general cognitive structures.

Addressing Modularity with Cognitive Models of Sentence Understanding

We believe that designing a sentence parsing system within a cognitive architecture will aid us to get a better insight of language integration with cognition. Our approach is driven by the hypotheses described in this section.

Language processing uses the same mechanisms as the other forms of cognition. We accept the evidence that language is highly integrated with the rest of cognition. In addition to the commonalities between syntactic and physical structure outlined in Cassimatis (2004), there are several reasons for this hypothesis in other parts of language. For example, Bloom (2000) and Tomasello, Kruger, and Ratner (1993) provide experimental results that show the general theory of mind learning also aids in learning words. Jackendoff (1990) demonstrates how concepts from physical reasoning can be used to formalize semantics of verbs in many fields. Clark (1996) analyzes many aspects of language use as a species of social interaction. All this

work makes it at least plausible that the syntactic structure of human languages can be accounted for using domain-general cognitive notions.

Wide-coverage of linguistic mechanisms is required. We hope to show how the various aspects of sentence processing relate to and rely on general cognitive processes. More specifically in modeling terms, the challenge is to show how a cognitive architecture with domain-general mechanisms can be used to model syntactic parsing. The design of such a model can be approached either by (1) detailed modeling of the components of the process (e.g. part of speech disambiguation, past tense retrieval (Taatgen & Anderson 2002)) followed by integration or (2) through comprehensive modeling of the entire process from the beginning. Though careful accurate component modeling can give us tight fits to human data, these highly focused models have fewer constraints leading to higher risk of overfitting exclusively to the data set studied. On the other hand in a comprehensive model the need for the various subparts to interact with each other naturally, provides a stronger and more restrictive list of constraints. Though we realize that even a comprehensive model is just a possible explanation, the chances of overfitting are greatly reduced with the increase in the complexity of the task being modeled. Furthermore, limiting the study to a specific component may also lead to situations where the hardest part of the problem lies outside the focus of study. These convictions lead us to build a model of the entire sentence parsing system instead of focused components.

Base model on wide-coverage grammatical theory. To be confident that our model can potentially account for the wide array of syntactic phenomena, we decided to base our model on a wide-coverage theory of grammar. Among the various candidate grammatical theories, Head-driven Phrase Structure Grammar (HPSG) has wide acceptance and several computational implementations. The flexibility and broad coverage offered by few general rules make HPSG a powerful tool. Though we are primarily concerned with the syntactic aspect of sentences, HPSG can also be used as a tool to bridge syntax with semantics (Dorna & Emele, 1996). These features of HPSG and the general trend in the field towards HPSG have led us to base our model on HPSG as proposed by Sag, Wasow and Bender (2003).

Parsing integrates multiple cognitive structures and processes. For example, a common trend in HPSG and other parsing and grammatical literature has been to use statistics to fine tune the bare HPSG Parser (Torisawa, Nishida, Miyao, & Tsujii, 2000). The various aspects in which statistics are used include skewing the success of a phrase in the lexicalization step, handling word sense disambiguation, prioritizing the selection of possible parse tree path and assigning weight to HPSG rules themselves. Certain techniques like learning rules based on induction are best captured by connectionist models. Thus to be able to implement a complex system like a parser, we require a framework that provides a seamless way of integrating various computa-

tional mechanisms. Because the Polyscheme cognitive framework is designed to model the integration of multiple reasoning and learning mechanisms, we developed our model within it.

Polyscheme

Polyscheme is a cognitive framework which has been used to model a wide range of phenomena from fundamental physics laws to theory of mind based intention reading (Cassimatis 2005). The driving principle behind Polyscheme is that different high-level tasks use the same underlying set of common functions and that these common functions can be implemented using multiple computational approaches. Each module implementing a common function is called a specialist and a specialist shares opinions with the other specialists through altering the truth values of a common substrate of propositions. As the specialists are integrated through a common substrate, the internal working of each specialist is abstracted, enabling the specialist to use any computational method required. Some of the mechanisms used for implementing the parser include rules, spreading of activation and constraint satisfaction.

The various specialists used for the parser include difference specialist, temporal specialist and rule specialist. The difference specialist checks for uniqueness and identity of objects, while the temporal specialist handles sequencing of events. The rule specialist by itself enables a simple and powerful production system. The rules stored in the rule specialist consist of a set of antecedent propositions, the set of resulting propositions and a strength of the rule in terms of confidence level. For the sake of familiarity an example of a proposition is

```
PropositionName object1 object2 Time World.
```

A proposition can take any number of objects. Note that all propositions have the time, for which they hold true, associated with them. A proposition can exist with varying truth values in different worlds and this world concept can be thought of as namespaces within the common substrate.

A Model of Syntactic Parsing

The major insight our model is based on is that representations normally used to capture nonlinguistic relations can be used to represent linguistic relations. Once we show how to represent these nonlinguistic relations in Polyscheme, then its existing domain-general reasoning mechanisms will infer the structure of a sentence with no special inference machinery just for syntax.

The HPSG grammar is based on words and phrases modeled as feature structures. The feature structures interact with each other through the constraints of the Grammar rules to form larger phrases and eventually a sentence. Hence the essential components of modeling a HPSG parser are word and phrase representations, feature structures, grammar rules and lexical rules. Here we show how these basic HPSG entities can be represented in terms of domain general concepts.

Words and phrases as events

The utterance of a word is treated in our model like the occurrence of any other event that can be perceived. In this case the perceived event is the `Utterance` event. Some of the properties required to distinctly identify and describe an `Utterance` event (for example the occurrence of word "dog" in the sentence "The dog barked") are the start time, end time and phonology of the word.

```
ISA dogphr UtteranceEvent E R.
StartTime dogphr t2 E R.
EndTime dogphr t3 E R.
Phonology dogphr 'dog' E R.

ISA theDogPhr UtteranceEvent E R.
StartTime theDogPhr t1 E R.
EndTime the DogPhr t3 E R.
Phonology theDogPhr 'the dog' E R.
```

A phrase which is formed by combining neighboring words is also considered a `Utterance` event as the phrase shares the same properties of unique start time, end time and phonology. For example, the noun phrase "the dog" which is generated by combining `thephr` of time t1 to t2 and `dogphr` of time t2 to t3 has the structure shown above. `ISA` is the Polyscheme proposition denoting object category.

Feature Structure as Event Structure

Feature Structures of words and phrases form the basic building blocks of parse trees and sentences in HPSG. The typical features of a phrase or word are `Head, Part of Speech, Agreement, Number, Person, Specifier, Complement, Phonology` and `Gap`. In Polyscheme, the utterance of a word is an event and the features of this event are considered as objects. For example, the Head feature of the event `dogphrase` is an object, say `dogphrHead`. A characteristic property of all these features is that each feature can take only one object as its value. In Polyscheme an object taking a single value is termed as an attribute. Declaring the features as attributes, lets us leverage on a number of built in Polyscheme functionalities.

$$\begin{bmatrix} Head \begin{bmatrix} POS \text{ Noun} \\ AGR \text{ [NUM Singular]} \\ COUNT \text{ Countable} \end{bmatrix} \\ PHONOLOGY \langle 'dog' \rangle \end{bmatrix}$$

Polyscheme representation of the above feature structure is

```
UtteranceEvent dogphr E R.
  Head dogphr dogphrHead E R.
    POS dogphrHead Noun E R.
    AGR dogphrHead dogphrAgr E R.
        NUM dogphrAgr Singular E R.
    COUNT dogphrHead Countable E R.
  PHONOLOGY dogphr 'dog' E R.
```

Note that the nested features are written as propositions relating the internal contained object to the subsuming object as shown with `Agreement` and `Num`. The `Gap` and `Comple-`

`ment` features take a list instead of one object. In Polyscheme this is captured by creating a list-object which in turn is associated with all the objects that it contains.

Tags In HPSG besides features, tags are used widely to represent relations between feature structures. Tags with the same label indicate that the feature structures are identical with essentially the same reference and not just copies holding similar values. For e.g. phrase1 and phrase3 are constrained to be the same by tag ||1||.

$$||1|| \begin{bmatrix} phrase1 \\ Head[POS \text{ Noun}] \end{bmatrix} \begin{bmatrix} phrase2 \\ Head[POS \text{ Verb}] \end{bmatrix} ||1|| \begin{bmatrix} phrase3 \\ Head[POS \text{ Noun}] \\ PHONOLOGY \text{ 'bat'} \end{bmatrix}$$

This is captured in Polyscheme by ensuring that the two objects are one and the same instance using the `Same` proposition as in `Same phrase1 phrase3 E R.`

Grammar Rules

An important part of HPSG is a small number of very general grammar rules. With just Head-Specifier, Head-Complement, Head-Modifier and Head-Filler Rule, an effective parsing system can be built.

Head-Specifier Rule: The Head Specifier Rule states that a phrase selecting a preceding or "specifier" phrase, when preceded by the required specifier combine to form a larger phrase. A typical example of Head Specifier rule instance is a verb requiring a subject as the specifier phrase. For example, in the sentence "Dogs barked", "barked" is the verb requiring the subject "dogs" as the specifier. Another common example of Head Specifier phrase is a noun requiring a determiner, as in "dog" requiring "a" or "the", to precede it.

Though the constraints of Head Specifier rule seem to be specific to words and phrases, we can see its core idea as an instance of a general rule, relating events with other events that must precede them. This concept of an event requiring another event to come before and qualify the event is common in many domains like planning where the preconditions (Fikes & Nilsson, 1971) of an event are explicitly stated. We can also find analogies of preceding events in common physical reasoning scenarios. For example, in the action of opening a locked door, the event of unlocking and opening the door is the main or head event. However, the `unlocking-door-event` is incomplete without the `retrieval-of-keys` event preceding it. Hence the `unlocking-door-event` corresponds to an example of a Head Specifier rule where the head event is unlocking and opening the door and the specifier event is taking out the keys. Examples of physical world preceding events are:

(Take out the keys) (Opening a locked door)
(Loading a gun) (Firing the shot)

Thus the *specifier* as defined in HPSG can be thought of as *a preceding event in domain-general terms*. Hence the specifier attribute of a HPSG feature structure is captured through the Preceding proposition in Polyscheme.

The HPSG head specifier rule is defined as follows in terms of feature structures and tags.

$$\begin{bmatrix} \text{phrase} \\ \text{SPR} \quad \langle \ \rangle \end{bmatrix} \rightarrow \|1\| \quad H \begin{bmatrix} \text{phrase} \\ \text{SPR} \quad \langle \|1\| \rangle \\ \text{COMPS} \ \langle \ \rangle \end{bmatrix}$$

There are three constraints posed by the Head Specifier rule. The first constraint is that the Head phrase must immediately follow the Specifier (SPR) phrase in time. Secondly, the Complement (COMPS) feature of the Head phrase must already be satisfied and made empty. The final constraint ensures the identity of the event that comes before the phrase and the specifier of the phrase through the tag ||1||.

The domain-general Polyscheme rule that captures the Head Specifier rule is in terms of preceding event and following event corresponding to the specifier and complement attributes of HPSG.

```
Meets ?precedingevt ?headevent E ?w +
Preceding ?headevent ?precededby E ?w +
Following ?headevent empty E ?w +
Same ?precedingevt ?precededby E ?w
=> ,
Preceding ?Phrase- empty E ?w +
PercolatePrinciples ?Phrase- ?headevent E ?w
```

The immediate sequencing of the specifier and Head phrase is an essential constraint to be modeled. In Polyscheme, the `Temporal Specialist` tracks the sequencing of events and the `Meets` proposition specifically indicates the ending time of the first event coinciding with the beginning time of the second event. The next constraint enforced by the Head Specifier rule is on the `Following`-event attribute of the Head Phrase to be empty. The final constraint of the Head Specifier rule, imposed by the tag ||1|| is captured though the `Same` proposition.

Though `Same(precedingevt,precededby)` is easy to capture, Polyscheme does not initially assume that `precedingEvt` and `precededBy` phrases could be the same as each word occurs independently of the other words. The new object created for `Preceding` event attribute of the head phrase is hence independent of the phrase, `precedingevt`. To capture the identity between `precedingevt` and `precededBy`, a pre-head Specifier rule is introduced in Polyscheme. The pre-head Specifier states that if all other constraints of a Head Specifier Rule are satisfied then the phrases `precedingevt` and `precededby` are likely to be the same.

```
Meets ?precedingevt ?headevent E ?w +
PrecedingEvent ?headevent ?precededby E ?w+
FollowingEvent ?headevent empty E ?w
   ~~> , Same ?precedingevt ?precdedby E ?w
```
The difference specialist verifies that no attributes in the two phrases contradict and automatically falsifies the likelihood of `Same` proposition in the case of a contradiction. Hence the domain general Head-Specifier rule proceeds on

the likelihood of `Same (precedingevt, precededby)` being conserved by the difference specialist. This process of rejection by difference specialist is the essential constraint that filters invalid phrases like "the go" from valid head specifier phrases like "the dog".

Let us consider the formation of the phrase "the dog". The table below compares the feature structures of 'the' and 'dog' phrases.

The phrase	Dog phrase
Start Time t0	Start Time t1
End Time t1	End Time t2
POS determiner	POS Noun
Preceding empty	Preceding detPhrase
Following empty	Following empty

The matching rule instance of the pre-head specifier rule is

```
Meets ThePhr DogPhr E ?w +
Preceding DogPhr detPhrase E ?w +
Following DogPhr empty E ?w
   ~~> , Same ThePhr detPhrase E ?w
```

The difference specialist gives opinion on `Same (ThePhr, detPhrase)` by comparing the attributes of the two phrases. The attributes of ThePhr are listed above and the feature structure of a generic detPhrase in HPSG is given as:

$$\begin{bmatrix} Head\begin{bmatrix} POS \ \text{determiner}\end{bmatrix} \\ \text{SPR} \ \langle \text{empty} \rangle \\ \text{COMPS} \ \langle \text{empty} \rangle \\ MOD \ \langle \text{empty} \rangle \\ GAP \ \langle \text{empty} \rangle \end{bmatrix}$$

As the attributes defined for ThePhr - POS, Preceding and Following are determiner, empty and empty showing consistency with the feature structure of detPhrase, the difference specialist allows the claim `Same(precEvt, precBy)` to exist in likely confidence.

However, considering the phrase "the go"

The phrase	Go phrase
Start Time t0	Start Time t1
End Time t1	End Time t2
POS determiner	POS Verb
Preceding empty	Preceding nounPhrase
Following empty	Following empty

Matching rule instance:

```
Meets ThePhr GoPhr E ?w +
Preceding GoPhr nounPhrase E ?w +
Following GoPhr empty E ?w
   ~~> , Same ThePhr nounPhrase E ?w
```

The HPSG feature structure of a generic noun phrase is similar to the `detPhrase` except that the value of POS is Noun. The difference specialist falsifies `Same(ThePhr, nounPhrase)` as the POS attribute of `ThePhr` and noun-

`Phrase` are different. This falsification prevents the Head Specifier rule of 'the go' phrase from firing.

In the cases where the difference specialist preserves the likelihood of the same proposition, all the conditions required for the domain general Head Specifier rule are met. The result of Head Specifier rule is the complete feature structure of the combined event. The Percolation principles aid in the creation of the new phrase.

Percolation Principles:
When feature structures interact through Grammar Rules, only the specific feature corresponding to the rule applied is defined for the new phrase. For example a phrase formed by the Head Specifier Rule has only the preceding attribute defined (here empty). Hence in HPSG there are principles that fill in the other attribute values of a new phrase given that the feature structure of the composing phrases including the Head Phrase are known. The Gap Percolation rule concatenates the list of all the composing phrases' Gap objects in the order of occurrence. While the Head Feature Principle and Valence Principle state that the Head Feature and the Valence Feature of the new created phrase are the same as that of the Head Phrase. The applicability of Percolation rules in a domain general scenario is also intuitive. In the example of a gun firing event, the Head event is firing the shot. The features of this head event would include the type and features of the gun say Remington Model 68, 6.22 mm shotgun. These features are essential in the bigger gun firing event created by combining preceding and following terms to the head shot fired event.

Head-Complement Rule[1]: The HPSG Head Complement Rule is similar to Head-Specifier Rule in that a head phrase needs another qualifying phrase to build a new and complete phrase. However, the Head-Complement rule differs on the two accounts of the qualifier phrase following the head and the head phrase taking more than one qualifying phrase.

A typical example of a head complement rule is a verb taking objects. A strictly transitive verb like 'devour' takes a single complement after it like in the sentence 'he devoured the food'. On the other hand, a ditransitive verb like 'handed' takes a two object complement list as in 'he handed (me) (the pen)'. Prepositions, like in and on, also take complements typically noun phrases to create complete phrases like 'on the roof' and 'in the room'.

Even in real world events like dining in a restaurant need complement actions to complete the event. In this example, the process of dining would be the head event and paying the bill event which follows the head would be the complement. Considering an event of starting a car, the complements would be essentially the events that immediately follow it like engine starting and making a sound.

Hence the *domain-general `following` event* can effectively capture the *Complement attribute of HPSG features*.

The HPSG rule in terms of feature structures is

$$\begin{bmatrix} \text{phrase} \\ \text{COMPS} \ \langle \ \rangle \end{bmatrix} \rightarrow \text{H} \begin{bmatrix} \text{phrase} \\ \text{COMPS} \ \langle ||1||, .., ||n|| \rangle \end{bmatrix} ||1||, .., ||n||$$

The domain general Polyscheme rule enforcing the head complement rule is

```
Following ?headEvent ?actionList E ?w +
NOT Same ?actionList empty E ?w +
First ?actionList ?firstComplement E ?w+
Rest ?actionList ?otherComplements E ?w+
Meets ?headEvent ?followingEvent E ?w +
Same ?firstComplement ?followingEvent E ?w
 ==> ,
COMPS ?Phrase- ?otherComplements E ?w +
PercolatePrinciples ?Phrase- ?headEvent E ?w
```

In Polyscheme, the Head-Complement rule is implemented as a recursive rule that creates phrases by combining the Head with the first complement phrase. The `Following` attribute of the new phrase created has all but the first element of the Head event's `Following`, while all the other attributes of the new phrase are filled in through the percolation principles.

Head-Modifier Rule: Head modifier rule states that a phrase can modify a head even though it is not specifically selected by that head. For example, "The dog in the park barked" is a sentence in which the phrase "the dog" is further described using the modifier "in the park". Note that the phrase "The dog barked" would still make a valid sentence even without the modifier "in the park". Hence the essence of the head modifier relation is that the head can exist independently of the modifying event. However, the modifying phrase supplements the effect of the main event.

As with previous rules, we can generalize head-modifier rule to beyond language. This depends on seeing modifiers as events that contribute to alter another event without being required by it.

The Head Modifier rule given by HPSG and the corresponding domain general Polyscheme rule are shown below.
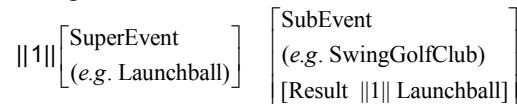
$$[\text{phrase}] \rightarrow \text{H} ||1|| \begin{bmatrix} \text{COMPS} \ \langle \ \rangle \end{bmatrix} \begin{bmatrix} \text{COMPS} \langle \ \rangle \\ \text{MOD} \ \langle ||1|| \rangle \end{bmatrix}$$

```
Meets ?headEvent ?ModifyingEvent E ?w +
Following ?headEvent empty E ?w +
Modifies ?ModifyingEvent ?BossEvent E ?w +
Following ?ModifyingEvent empty E ?w +
Same ?headEvent ?BossEvent E ?w +
    ==> ,
PercolatePrinciples ?Phrase- ?headEvent E ?w
```
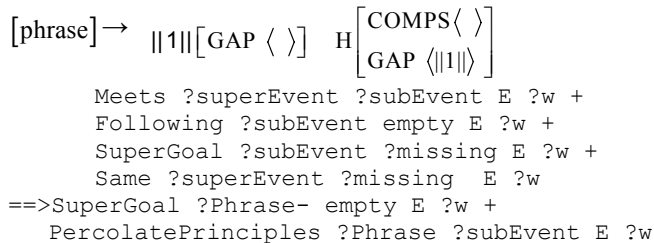
Head-Filler Rule: Head Filler Rule is HPSG's rule for dealing with long distance dependencies. An example of a sentence with long distance dependency is "The table that

Jim uses is old". In this sentence, although intuitively we know the object of the verb 'uses' to be 'the table', 'the table' is not spoken after the verb 'uses'. Hence the table though stated only once is essentially used in two places as in 'Jim uses the table' and 'the table is old'. Such hidden reference to a previously used term as in the phrase 'Jim uses' alluding to the object 'the table', is termed as a long distance dependency.

The concept of long distance dependencies can be thought of as a subordinate event modifying a super event. In this example "the table" is the super event while the phrase 'Jim uses the table' is the subordinate event or subgoal adding more information to the super event or super goal. Note that 'the table' or the super goal is itself a part of the subgoal 'Jim uses the table'. Real world analogies to an event with a subgoal include the scenario of launching a ball to break a window. The main event or super goal in this scenario is launching the ball. However, a possible subgoal is to swing a golf club for launching the ball. The swing-golf-club event provides more information to the launch-ball event while in itself taking the launch-ball event as a resulting action.

$$
\|1\| \begin{bmatrix} \text{SuperEvent} \\ (e.g. \text{ Launchball}) \end{bmatrix} \quad \begin{bmatrix} \text{SubEvent} \\ (e.g. \text{ SwingGolfClub}) \\ [\text{Result } \|1\| \text{ Launchball}] \end{bmatrix}
$$

Long distance dependencies are common in language usage and form a difficult problem in language understanding. In HPSG the unspecified `preceding` or `following` event in the subordinate phrase is indicated through the GAP feature. In the example sentence the object of the verb 'uses' would be a gap feature. The Head Filler rule describes how a new phrase can be formed when the super goal precedes a subgoal event with a gap in it. The corresponding Polyscheme rule is also shown below.

$$
[\text{phrase}] \rightarrow \quad \|1\| \begin{bmatrix} \text{GAP } \langle \ \rangle \end{bmatrix} \quad H \begin{bmatrix} \text{COMPS} \langle \ \rangle \\ \text{GAP } \langle \|1\| \rangle \end{bmatrix}
$$

```
    Meets ?superEvent ?subEvent E ?w +
    Following ?subEvent empty E ?w +
    SuperGoal ?subEvent ?missing E ?w +
    Same ?superEvent ?missing  E ?w
==>SuperGoal ?Phrase- empty E ?w +
  PercolatePrinciples ?Phrase ?subEvent E ?w
```

Domain-General Representations

HPSG Type	Category
SPR	Preceding Event
COMPS	Following Event
MOD	Modifies Event
GAP	Super Goal

The HPSG features are mapped to the above representations which are automatically handled by Polyscheme's common functions with no additional language specific mechanisms.

Conclusions

Our model demonstrates that domain-general cognitive mechanisms are sufficient to model syntactic parsing.

This has several implications for issues surrounding research into language and thought. First, the implausibility of non-modular approaches due to the superficial conceptual differences between linguistic and nonlinguistic cognition are reduced. Second, it suggests a possible explanation for children's ability to learn language within a short time frame (Chomsky, 1980) and with relatively small exposure to linguistic input (Pullum, 1996). Many of the processes and structures required for language may have developed in children before they begin to understand and use language. Third, if the mind represents syntactic and non-syntactic information using the same mechanisms, the puzzle of how the two forms of information can interact so seamlessly in language use is reduced.

In the future, we plan to expand on this ability of reasoning, social awareness, and meta-information to aid in the disambiguation of both word senses and parse trees. Integrating the semantics HPSG offers with the syntactical parse structures is another area to be pursued.

References

Cassimatis, N. (2005). Integrating cognitive models based on different computational methods. *Proceedings of the 10th Conference of the Cognitive Science Society*.

Cassimatis, N. L. (2004). Grammatical Processing Using the Mechanisms of Physical Inferences. *The Twentieth-Sixth Annual Conference of the Cognitive Science Society*.

Chomsky, N. (1980). *Rules and Representations*. Oxford: Basil Blackwell.

Clark, H. H. (1996). *Using Language*. Cambridge University Press, Cambridge.

Dorna, M. & Emele, M. (1996) Semantic-based Transfer, Proc. *16th Inernational Conference on Computational Linguistics* (pp. 316—321). København, Denmark.

Fodor (1983). The Modularity of Mind, Cambridge, MA

Fikes, R., & Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189-208.

Jackendoff, R. (1990). *Semantic Structures*. Cambridge, MA.

Marslen-Wilson, W., & Tyler, L. (1989). *Modularity in Knowledge Representation and Natural-Language Understanding*. Cambridge, MA: MIT Press. 37-62.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Pullum, G. (1996). Learnability, hyperlearning, and the poverty of the stimulus. *22nd Annual Meeting of the Berkeley Linguistic Society*. Berkeley, California.

Sag, I., Wasow, T., & Bender, E. (2003). *Syntactic Theory: A Formal Introduction*. (CSLI Lecture Notes No. 152).

Taatgen, N.A., & Anderson, J.R. (2002). Why do children learn to say "broke"? A model of learning the past tense without feedback. *Cognition* 86(2).

Tomasello, M., Kruger, A., and Ratner, H. (1993). Cultural Learning. *Behavioral and Brain Sciences,* 16:495-552.

Torisawa, Nishida, Miyao, & Tsujii (2000). An HPSG parser with CFG filtering. *Jounal of Natural Language Engineering* 6(1):63–80