Towards a Tight Integration of Syntactic Parsing with Semantic Disambiguation by means of Declarative Programming[*]

Yuliya Lierler
University of Nebraska at Omaha
ylierler@unomaha.edu

Peter Schüller
Sabancı University
peterschueller@sabanciuniv.edu

Abstract

We propose and advocate the use of an advanced declarative programming paradigm – answer set programming – as a uniform platform for integrated approach towards syntax-semantic processing in natural language. We illustrate that (a) the parsing technology based on answer set programming implementation reaches performance sufficient for being a useful NLP tool, and (b) the proposed method for incorporating semantic information from FRAMENET into syntactic parsing may prove to be useful in allowing semantic-based disambiguation of syntactic structures.

1 Introduction

Typical natural language processing (NLP) system consists of at least several components including syntactic and semantic analyzers. A common assumption in the design of an NLP system is that these components are separate and independent. On one hand, this allows researchers an abstraction necessary to promote substantial steps forward in each task, plus such a separation permits for more convenient, modular software development. On the other hand, constraints from "higher level" processes are frequently needed to disambiguate "lower level" processes. For example, consider the syntactically ambiguous sentence

I eat spaghetti with chopsticks. (1)

Its verb phrase allows for two syntactic structures:

$$\cfrac{\cfrac{\cfrac{eat}{(VP/PP)/NP} \quad \cfrac{spaghetti}{NP}}{VP/PP} \qquad \cfrac{with\ chopsticks}{PP}}{VP} \qquad \cfrac{\cfrac{eat}{VP/NP} \qquad \cfrac{\cfrac{spaghetti}{NP} \quad \cfrac{with\ chopsticks}{NP\backslash NP}}{NP}}{VP} \tag{2}$$

In the former, the prepositional phrase "*with chopsticks*" modifies the verbal phrase "*eat spaghetti*", and in the latter, it modifies the noun phrase "*spaghetti*". The sentence

I eat spaghetti with meatballs (3)

is syntactically ambiguous in a similar manner. In order to assign the proper syntactic structure to each of these sentences one has to take into account *selectional restrictions*, i.e., the semantic restrictions that a word imposes on the environment in which it occurs. For instance, in (1) the fact that a chopstick is an instrument suggests that "*with chopsticks*" modifies "*eat spaghetti*" as a tool for eating. Thus, an approach that integrates syntactic and semantic processing is essential for proper analysis of such sentences. Modern statistical methods, dominant in the field of syntactic analysis, take into account

[*] We would like to thank Vladimir Lifschitz and Ray Mooney for useful discussions and comments related to the topic of this work. Peter Schüller was supported by a TUBITAK scholarship.

selectional restrictions *implicitly* by assigning most probable syntactic structure based on observed co-occurrences of words and structures in corpora. Yet, this is often not sufficient. Sentences (1) and (3) illustrate this point, as the advanced parsers, including Stanford and Berkeley systems, do not produce proper syntactic representations for these sentences: instead they favor the same structure for both of them.[1] Similarly, semantic role labelers (joint syntactic-semantic parsers) such as SEMAFOR (Das et al., 2010) and LTH (Johansson and Nugues, 2007) display the same issue. The FRAMENET project (Baker et al., 1998) provides information that can disambiguate sentences (1) and (3). For instance, the frame *food* corresponds to the word "*spaghetti*". This frame contains information that *food* only takes other *food* as constituents. Thus modifying "*spaghetti*" with "*chopsticks*" in a parse tree for (1) yields a forbidden situation.

In this paper we present preliminary work on a system for natural language parsing that targets a tight integration of syntax and semantics. We illustrate its ability to take into account both quantitative and qualitative data available for processing natural language, where the former stems from statistical information available for natural language and the latter stems from lexical and commonsense knowledge available in lexical datasets such as FRAMENET.

Lierler and Schüller (2012) developed a Combinatory Categorial Grammar (CCG) parser ASPCCGTK[2]. A distinguishing feature of ASPCCGTK is that its design allows for synergy of both quantitative and qualitative information. First, it relies on the C&C part-of-speech *supertagger* (Clark and Curran, 2007) – built using latest statistical and machine learning advances in NLP. Second, its implementation is based on a prominent knowledge representation and reasoning formalism — answer set programming (ASP), see Brewka et al. (2011). ASP constitutes a convenient framework for representing constraints posed by selectional restrictions *explicitly*; thus we can augment implicit information available from statistical part-of-speech tagging with qualitative reasoning. We believe that the ASPCCGTK parser is a strong ground for designing a systematic, elaboration tolerant, knowledge intensive approach towards an integrated syntax-semantics analysis tool. Performance results on ASPCCGTK reported in (Lierler and Schüller, 2012) suggest that the "planning" approach adopted for parsing in the system scales to sentences of length up to 15 words. It may be sufficient for a number of applications: for example, 6.87 is the average number of words in sentences in the GEOQUERY corpus (Zelle and Mooney, 1996). But, in order for ASPCCGTK to become a viable NLP technology it is important to address the issue of its scalability.

The two contributions of this paper are as follows. First we demonstrate how use of the Cocke-Younger-Kasami (CYK) algorithm (Kasami, 1965) enhances the performance of ASPCCGTK. We evaluate the new approach implemented in ASPCCGTK on the CCGbank corpus (Hockenmaier and Steedman, 2007) and report the results. Second we propose and illustrate the method on how (a) FRAMENET can be used for properly disambiguating sentences (1) and (3), and (b) how this information is incorporated into the ASPCCGTK system. As a result we are able to use the ASPCCGTK parser to generate only the expected syntactic structures for the sentences in question.

In the future, we will automate a process of extracting selection restriction constraints from the data available in FRAMENET, by building an interface between ASPCCGTK and FRAMENET. CCGbank will provide us with extensive real world data for evaluating our approach. Once successful, we will look into expanding the approach to the use of other semantic annotations datasets for lexical items such as VERBNET, PROPBANK, NOMBANK and others for more complete sets of lexical constraints.

2 Extending ASPCCGTK for parsing CCG with CYK in ASP

Combinatory Categorial Grammar (Steedman, 2000) is a formalism that uses a small set of combinatory rules and a rich set of categories. Categories are either atomic such as NP, or complex such as $S\backslash NP$, which is a category for English intransitive verbs. The category $S\backslash NP$ states that an NP to the left of the

[1] Thanks to Nathan Schulte for carrying out the research supporting this claim by investigating the behavior of nine state-of-the-art parsers listed at `http://nlp.stanford.edu/software/stanford-dependencies.shtml`.

[2] `http://www.kr.tuwien.ac.at/staff/former_staff/ps/aspccgtk`

word will result in a sentence S. Given a sentence and a lexicon containing a set of word-category pairs, we replace words by appropriate categories and then apply combinators. For example, in the former derivation in (2), "*eat*" has category $(VP/PP)/NP$ and "*spaghetti*" has category NP. The combinator used in this derivation is *forward application* (*fa*)

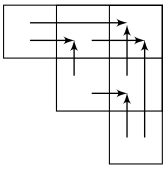
$$\frac{A/B \quad B}{A}\,fa \tag{4}$$

where A and B are variables that can be substituted for CCG categories. Applying forward application to "*eat*" and "*spaghetti*" substitutes A with VP/PP and B with NP and yields VP/PP. An input sentence is part of a grammar if some sequence of applying combinators results in the category S at the root of the parse tree.

The implementation of AspCcgTk is based on answer set programming – a declarative logic programming paradigm. ASP roots in answer set semantics of logic programs (Gelfond and Lifschitz, 1988). The idea of ASP is to represent a problem by a program whose answer sets correspond to solutions. For example, for parsing we encode the grammar and the input sentence in a way that each answer set corresponds to a valid parse tree of the input. Unlike in an imperative style of programming, in declarative programming we describe a specification of the problem, which expresses what the program should accomplish rather than prescribing how to do it. Answer set solvers use this specification to efficiently navigate through a search space and find solutions to the problem. For a more detailed and yet brief introduction of CCG and ASP we refer the reader to (Lierler and Schüller, 2012).

The CYK (Cocke, Younger, and Kasami) algorithm for context-free-grammars was initially published by Kasami (1965). It can be extended to CCG using ideas from Lange and Leiß (2009). Given an input of n words, CYK operates on an $n \times n$ triangular chart. Words in the input are associated with categories in the diagonal of the chart. Combinatory rules combine categories from two chart cells into a single category in another "corresponding" cell. We illustrate these intuitions using a 3×3 chart:

$$\begin{tikzcd} A/B \arrow[r] & A \\ & B \arrow[u] \end{tikzcd}$$



$$\begin{tikzcd} \texttt{Pleft,Pj} \arrow[r] & \texttt{Pi,Pj} \\ & \texttt{Pi,Pdown} \arrow[u] \end{tikzcd}$$

1,1	2,1	3,1
	2,2	3,2
		3,3

(5)

An input is recognized as part of the grammar if the top right chart cell contains the category `S` after successive application of combinators to the chart. A realization of CYK for recognition of context-free grammars in ASP was described by Drescher and Walsh (2011). First, we adapt their approach to CCG. Second, we extend it to the task of generating parse trees as we are not only interested in recognizing grammatical inputs but also in producing appropriate parses.

We now show parts of our realization of CYK in the ASP formalism. We represent a chart using a predicate `grid(Pi,Pj,Cat)` and initialize the diagonal using the rule

```
grid(P,P,C) :- category_at(C,P).
```

where the `category_at` predicate is obtained from tagging the input with the C&C supertagger.

Forward application is realized as follows (grid variables are shown in (5), X and Y are variables that stand for CCG categories):

```
applicable(fa,Pj,Pi,Pleft,Pdown,X,rfunc(X,Y),Y) :-
  grid(Pleft,Pi,rfunc(X,Y)), grid(Pj,Pdown,Y).
grid(Pj,Pi,X) :- applicable(_,Pj,Pi,_,_,X,_,_).
```

where `rfunc(X,Y)` encodes complex category of the form X/Y. The first rule defines where the *fa* combinator can be applied to within the chart. The second rule defines which categories this application creates. For obtaining parse trees, we "guess" for each instance of `applicable` if that combinator is actually applied (`SrcLeft`, `SrcDown`, and `Result` stand for CCG categories in the CYK grid):

```
{ applied(Comb,Pj,Pi,Pleft,Pdown,Result,SrcLeft,SrcDown) } :-
  applicable(Comb,Pj,Pi,Pleft,Pdown,Result,SrcLeft,SrcDown).
```

The curly bracket construct in the head of this rule is what expresses the guess as we can intuitively read this rule as follows: an expression in the head *may* hold in case if the body of the rule holds.

To obtain only valid parse trees, we furthermore (i) add rules that constraint the selection of multiple `applied` combinators in one cell, (ii) define reachability of diagonal chart cells from the S category in the top right cell, and (iii) add rules that require all diagonal cells to be reachable.

We believe that the possibility of explicitly representing alternatives and then restricting them by expressing appropriate conditions using declarative means makes ASP a powerful tool for working with ambiguities in the field of NLP.

We conducted empirical experiments to compare the performance of the original AspCcgTk and the AspCcgTk enhanced with CYK as described here. We report average times and number of timeouts when parsing all sentences of Section 00 of the CCGbank corpus (Hockenmaier and Steedman, 2007) using a timeout of 1200 seconds. The sentences were chunked and tagged by the C&C supertagger. The benchmark results show that the CYK approach has a significant performance advantage over the old approach. Columns show average time in seconds for groups of sentences of a certain length. Number in parenthesis represents the number of timeouts.

Number of Words	1-10	11-15	16-20	21-25	26-30	31-35	36-40	41+
Number of Sentences	195	285	345	330	287	224	118	129
AspCcgTk (CYK)	0.04	0.18	0.49	1.04	2.11	3.21	6.66	27.01(3)
AspCcgTk (original)	0.13	1.07	4.90	20.93	68.21(1)	194.59(2)	342.88(24)	497.93(75)

3 Semantic Disambiguation using FRAMENET

FRAMENET is a dataset of semantic relations based on Frame Semantics (Fillmore and Baker, 2001). Lexical items *evoke* certain *frames* that contain *frame elements*; for example, "*eat*" evokes an *ingestion* frame and everything that is of *semantic type* **food** evokes a *food* frame. Sample information available in the *ingestion* and *food* frames follow:

Frame	*Frame Element*	*Semantic Type*
ingestion	INGESTOR	**sentient**
	INGESTIBLE	**ingestible**
	INSTRUMENT	**tool**
	MANNER	**manner**

Frame	*Frame Element*	*Semantic Type*
food	CONSTITUENT	**food_constituent**

Each frame element is a slot that *may* be filled only by elements of the correct semantic type. Types are organized in a taxonomy. For instance, the following part of the taxonomy is relevant to this presentation:

$$\textbf{tool}\ is\_a\ \textbf{instrument} \qquad \textbf{food}\ is\_a\ \textbf{ingestible} \qquad \textbf{food}\ is\_a\ \textbf{food\_constituent}.$$

We propose a concept of a "semantically coherent" parse tree. Information from FRAMENET allows us to disambiguate semantically coherent and incoherent trees. We now make these ideas precise. Each node in a tree is annotated with a *tag* – either a distinguished tag $\perp$ or a pair $T||F$ where both T and F are sets consisting of semantic types. Each leaf of a tree is assigned a tag $T||F$ in accordance with FRAMENET information for a corresponding word. The set T contains the semantic types associated with the leaf-word. For instance, for word "*spaghetti*", this set T_{sp} is $\{$**food**, **food_constituent**, **ingestible**$\}$. The set F contains the semantic types associated with the frame elements of a frame evoked by a leaf-word. For instance, for word "*eat*" that evokes the *ingestion* frame, this set F_{eat} is

$$\{\textbf{sentient}, \textbf{ingestible}, \textbf{tool}, \textbf{manner}\}.$$

To define a tag for a non-leaf node of a tree we introduce the following terminology. Any non-leaf node in a CCG parse tree is a parent of two children: a *functor* and an *argument*. Depending on semantic information assigned to nodes, they act as functors or arguments. For a non-leaf node p, we define a tag $T_p||F_p$ as follows

$$T_p||F_p = \begin{cases} \bot & \text{if a tag of either } f \text{ or } a \text{ is } \bot \\ \bot & \text{if } F_f \cap T_a = \emptyset \\ T_f||(F_f \setminus \{s\}) & \text{if there is a semantic type } s \in F_f \cap T_a \end{cases}$$

where f and a stand for a functor and an argument children of p, respectively. Pairs $F_f||T_f$ and $F_a||T_a$ correspond to tags of these children. We say that a parse tree is *semantically coherent* if there is no node in the tree annotated by the $\bot$ tag.

Recall the syntactic structures (2) corresponding to the verb phrase of sentence (1). The annotated counterpart of the former structure follows[3]:

$$\dfrac{\dfrac{\dfrac{\mathit{eat}}{(VP/PP)/NP : \emptyset||F_{eat}} \quad \dfrac{\mathit{spaghetti}}{NP : T_{sp}||\{\textbf{\textsf{food\_constituent}}\}}}{VP/PP : \emptyset||\{\textbf{\textsf{sentient}}, \textbf{\textsf{tool}}, \textbf{\textsf{manner}}\}} \quad \dfrac{\mathit{with\ chopsticks}}{PP : \{\textbf{\textsf{tool}}, \textbf{\textsf{instrument}}\}||\emptyset}}{VP : \emptyset||\{\textbf{\textsf{sentient}}, \textbf{\textsf{manner}}\}}$$

This subtree is semantically coherent. On the other hand, part of the later structure in (2) constitutes semantically incoherent subtree:

$$\dfrac{\dfrac{\mathit{spaghetti}}{NP : T_{sp}||\{\textbf{\textsf{food\_constituent}}\}} \quad \dfrac{\mathit{with\ chopsticks}}{NP\backslash NP : \{\textbf{\textsf{tool}}, \textbf{\textsf{instrument}}\}||\emptyset}}{NP : \bot}$$

To implement described process within ASPCCGTK approach, we first manually specify a dictionary that contains FRAMENET information sufficient for annotating leaf nodes stemming from the words in an input sentence. We then use logic rules to (a) define annotations for non-leaf nodes of parse trees and (b) restrict the produced parse trees only to these that are semantically coherent. On the sentences (1) and (3), the ASPCCGTK parser implementing this approach is capable to enumerate only and all semantically coherent parses that correspond to syntactic structures expected for the sentences.

4 Conclusions and Future Work

In this work we propose and advocate the use of advanced declarative programming paradigm – answer set programming – as a uniform platform for integrated approach towards syntax-semantic processing in NLP. We illustrate that the CCG parser ASPCCGTK based on an ASP implementation reaches performance sufficient for being a useful NLP technology by taking advantage of the data structures of the CYK algorithm. Even though ASP has a high worst-case complexity, a related declarative paradigm with the same worst-case complexity was shown to be effective for solving NLP problems: the usage of Integer Linear Programming in (Roth and Yih, 2007). We also propose a method for disambiguating syntactic parse trees using the semantic information stemming from the FRAMENET dataset and implement it within the ASPCCGTK parser. This implementation results in the first step towards a synergistic approach in syntax-semantic processing by means of technology such as ASP. There is an open question on how to automatically fetch relevant information from FRAMENET in order to make the proposed implementation widely usable. This is the subject of ongoing and future work. One reasonable direction to explore is assess the usability of FRAMENET-based semantic role labeling systems for our purposes, in particular, LTH and SEMAFOR. The CCGbank will serve us as a test bed for evaluating the effectively of proposed method and directing this research. The source code of the reported implementation is available online at the ASPCCGTK website under version 0.3.

[3] The definition of semantically coherent trees presented here is a simplified version of a more complex construct, which takes into account functors that carry no semantic type information by themselves (for example, a functor corresponding to a word "*with*") but rather inherit this information from its argument.

References

Baker, C. F., C. J. Fillmore, and J. B. Lowe (1998). The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, San Mateo, CA, pp. 86–90.

Brewka, G., I. Niemelä, and M. Truszczyński (2011). Answer set programming at a glance. *Communications of the ACM 54(12)*, 92–103.

Clark, S. and J. R. Curran (2007). Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics 33*(4), 493–552.

Das, D., N. Schneider, D. Chen, and N. A. Smith (2010). Probabilistic frame-semantic parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, Stroudsburg, PA, USA, pp. 948–956. Association for Computational Linguistics.

Drescher, C. and T. Walsh (2011). Modelling grammar constraints with answer set programming. In *International Conference on Logic Programming*, Volume 11, pp. 28–39.

Fillmore, C. J. and C. F. Baker (2001). Frame semantics for text understanding. In *WordNet and Other Lexical Resources, NAACL Workshop*.

Gelfond, M. and V. Lifschitz (1988). The stable model semantics for logic programming. In R. Kowalski and K. Bowen (Eds.), *Proceedings of International Logic Programming Conference and Symposium (ICLP'88)*, pp. 1070–1080. MIT Press.

Hockenmaier, J. and M. Steedman (2007). CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguististics 33*(3), 355–396.

Johansson, R. and P. Nugues (2007). Lth: semantic structure extraction using nonprojective dependency trees. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, Stroudsburg, PA, USA, pp. 227–230. Association for Computational Linguistics.

Kasami, T. (1965). An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, Massachusetts.

Lange, M. and H. Leiß (2009). To CNF or not to CNF? An efficient yet presentable version of the CYK algorithm. In *Informatica Didactica 8*. Universität Potsdam.

Lierler, Y. and P. Schüller (2012). Parsing combinatory categorial grammar via planning in answer set programming. In *Correct Reasoning*, Volume 7265 of *Lecture Notes in Computer Science*, pp. 436–453. Springer.

Roth, D. and W. Yih (2007). Global inference for entity and relation identification via a linear programming formulation. In L. Getoor and B. Taskar (Eds.), *Introduction to Statistical Relational Learning*. MIT Press.

Steedman, M. (2000). *The syntactic process*. London: MIT Press.

Zelle, J. M. and R. J. Mooney (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, Portland, OR, USA, pp. 1050–1055.