Introduction to Natural Language Syntax and Parsing
Lecture 6: Combinatory Categorial Grammar

Stephen Clark

October 17, 2015

Long-Range Dependencies An interesting feature of natural languages is that they have syntactic constructions which allow unbounded amounts of intervening material between items which belong together in the semantic predicate-argument structure. If the job of a parser is to return such predicate-argument structure, then it needs an analysis of these constructions.

The obvious example for English is the (object) relative clause construction. In the examples on the slide, the direct object *a woman* of the verb *likes* has been extracted from the canonical object position, to the front of the noun phrase before the relative pronoun. The point of the examples is to demonstrate that, at least in principle, there is no limit to the number of words that can appear between the verb and the extracted object.

The Relative Clause Construction In CCG, the lexical category assigned to a transitive verb is always $(S\backslash NP)/NP$, irrespective of the syntactic environment it finds itself in. (This is not true of other linguistic formalisms, for example TAG.) Hence the question regarding the relative clause construction is: what is the appropriate lexical category for *whom*?

I like to think of this question as akin to a jigsaw puzzle, where we start to fill in parts of the analysis and see what's left. In this case, the type of *whom Warren likes* needs to be $NP\backslash NP$, so that it can combine with the extracted object NP to the left, and return an NP for the whole noun phrase. So it looks as though the category for *whom* has to be $(NP\backslash NP)/X$, for some X to be determined.

"Non-Constituents" in CCG If *whom* has the type $(NP\backslash NP)/X$, then *Warren likes* is a constituent with type X. Can a subject-verb combination be a constituent? Most linguistic theories would say no, but at least one of the tests for constituenthood — the coordination test — suggests that it can be (since *Warren likes* can be coordinated with other similar phrases such as *Dexter detests*).

A natural type for *Warren likes* is S/NP: a sentence missing an object NP to the right (analagous to a verb phrase, which is a sentence missing a subject NP to the left); in which case the type for *whom* is $(NP\backslash NP)/(S/NP)$. The fact that a subject-verb combination is not typically considered a constituent is the reason we have *non-constituents* in the slide title, and the scare quotes are there to suggest that perhaps these "non-constituents" should be considered constituents after all.

Deriving "Non-Constituents" In order to derive a type for *Warren likes*, we somehow need to combine an NP to the left with $(S\backslash NP)/NP$ to the right. The bracketing in $(S\backslash NP)/NP$ means this can't happen with function application (since the object NP to the right needs cancelling first with application). Two new rules, which take us beyond classical categorial grammar, will allow the combination: type-raising and composition.

Type-Raising Type-raising arises from the question: why should the verb be the function, and not the subject noun phrase? Assuming a subject NP can be a function, what would it naturally look for? The answer is a verb phrase $(S\backslash NP)$. Hence the type-raised category for a subject NP becomes a sentence missing a verb phrase to its right: $S/(S\backslash NP)$.

More generally, type-raising is represented by a unary rule *schema*: $NP \Rightarrow T/(T\backslash NP)$, where the variable T gets instantiated in a rule instance, in the current example with S. The way I like to describe type-raising is as follows: the type-raised NP looks to the right for a category looking to the left for it $(T\backslash NP)$, and when it's found that category it returns the category which the category to the right would have returned, if the category to the right had found it (i.e. T). Got it?

This type-raising rule is known as *forward* type-raising, since the resulting category looks to the right for its argument. Later we'll also encounter backward type-raising, where it looks to the left.

Forward Composition Type-raising has created a category which is looking to the right for a verb phrase $(S\backslash NP)$, and there is a verb phrase to the right, but the problem is that it's embedded in the transitive verb category – $(S\backslash NP)/NP$. The object NP, which has been extracted to the front of the noun phrase, has not yet been cancelled and hence is "getting in the way".

The rule of forward composition allows a category to "get inside" an argument category, and hence effectively bypass the object NP. The general schema is $X/Y\ Y/Z \Rightarrow X/Z$. Intuitively the Ys in the centre are cancelling. The way I like to describe forward composition is as follows: we can return an X if only we can find a Y to the right; we have a Y to the right, but only if we can find a Z further to the right. So let's just look for a Z to the right and immediately return an X, ignoring the Y.

CCG Derivation for a Relative Clause Once type-raising ($> \mathbf{T}$) and composition ($>\mathbf{B}$)[1] have created the S/NP constituent, then the derivation is straightforward. The category for the relative pronoun — $(NP\backslash NP)/(S/NP)$ — effectively "knows" that it's in the object extraction scenario, so it's looking to the right for a category which fits that scenario (one where the object NP hasn't yet been cancelled).

"Spurious" Ambiguity The use of type-raising and composition does not have to be confined to sentences with long-range dependencies. In practice, a parser will use whatever combinatory operations it has at its disposal. Hence, type-raising and composition can be used even for simple subject-verb-object sentences, as in the example on the slide, leading to additional syntactic ambiguity. This ambiguity has often been referred to as "spurious" ambiguity, since the resulting *semantic* interpretation remains the same. For example, if a logical form for the sentence were built using the combinatory operations, applying the techniques described in the other half of the course, the result would be the same for the derivation on the slide and the canonical derivation using only function application.

Generalised Forward Composition In the example on the slide, the type of *offered* is $((S\backslash NP)/PP)/NP$. It needs to be coordinated with *may give*, in which case *may give* also has to have this type. The types of *may* — $(S\backslash NP)/(S\backslash NP)$ — and *give* — $((S\backslash NP)/PP)/NP$ — look as though they ought to combine by forward composition, except that the $(S\backslash NP)$ which needs cancelling in the type for *give* is too far embedded into the category for forward composition to work.

The solution is to allow *generalised* forward composition, which allows the category on the left to get further "inside" the category on the right. The intuition is the same as for vanilla forward composition — the category in the middle is cancelling — but this time we're ignoring an extra set of brackets.

The combinatory rule described above is referred to as $> \mathbf{B}^2$, where the 2 denotes the level of embedding of the argument category (in this case $S\backslash NP$). The rule can be generalised further to $>\mathbf{B}^n$, where n is greater than 2, so that the category on the left is able to penetrate further into categories on the right with more recursive structure.

Argument Cluster Coordination The example on the slide is an example of what is often referred to as "non-constituent coordination". Since CCG has such a flexible notion of constituenthood, it turns out that even the required conjuncts in this example — *a teacher an apple* and *a policeman a flower* — can be built using combinatory rules.

[1]The use of **B** for composition follows Steedman's notation, who followed Curry (as in Curry and Feys combinatory logic).

Forward and Backward Type-Raising Backward type-raising is analagous to the forward case. If we again instantiate the T variable with S, then applying backward type-raising to an (object) NP results in $S\backslash(S/NP)$. Using a similar explanation to before, a (type-raised) NP can look to the left for a category looking for it to the right, and when it finds that category the result is whatever would have resulted if the category to the left had found the NP (in this case an S).[2]

Argument Cluster Coordination In the example on the slide, backward type-raising has been applied to all four arguments, with the T variable in the rule schema being instantiated in two different ways. (Exercise for the reader: determine T in the two cases.) Now we need a rule to combine the complicated-looking categories.

The derivation is much easier to understand if we use the abbreviations on the slide, replacing $S\backslash NP$ with VP (verb phrase), $(S\backslash NP)/NP$ with TV (transitive verb), and $((S\backslash NP)/NP)/NP$ with DTV (ditransitive verb). Now it looks as though the categories could combine with some form of composition, and a new rule of *backward* composition does the job. The general schema is $Y\backslash Z\ X\backslash Y \Rightarrow X\backslash Z$. Using a similar explanation to before, we can return an X if only we can find a Y to the left, and we have a Y to the left, if only we can find a Z further to the left; so let's just look for a Z to the left and return an X, ignoring the Y.

Backward Crossed Composition Other linguistic phenomena suggest the need for additional rules. The phenomena often involve coordination, as in the *buy today and cook tomorrow* example. The use of *backward-crossed* composition allows the types of *buy* and *today* to combine in the required fashion. (Explaining this one is left as an exercise for the reader.)

Another Combinatory Rule One question you may be asking at this stage is: how do we decide which combinatory rules are allowed? From a linguistic theory perspective, the approach is usually to see which linguistic phenomena an additional rule could help explain, whilst at the same time not licencing analyses for ungrammatical sentences of the language in question. One rule which we would not want to add to the English grammar is *forward*-crossed composition. However, there are constructions in Dutch which appear to require this rule.

Cross-Serial Dependencies in Dutch There are some sentences involving subordinate clauses in Dutch which appear to have some level of *crossing* dependencies. The translation of the example on the slide is *because I saw Cecilia help Henk feed the hippos.* The indices on the NP arguments are not part of the atomic symbol, but are there to indicate where the dependencies are in the

[2]Note this allows the possibility of another derivation for a subject-verb-object sentence, again resulting in the same semantic interpretation.

sentence. For example, *NP₄* is *the hippos*, which is also the thing being fed (object of *voeren*). Note that, in this Dutch construction, the arguments are listed before the verbs and, crucially, the respective orders of the arguments and verbs mean that the dependencies cross, rather than nest, as they do in the English translation. The dependencies are often referred to as *cross-serial* because the crossings have a serial quality to them, also; i.e. the noun phrases and verbs have to line up in a particular order.

It is left as an exercise to the reader to understand how the rules of forward-crossed composition and generalised forward-crossed composition can enable a derivation which captures this crossing.

Mild Context Sensitivity A long-standing question in theoretical linguistics concerns how much automata-theoretic power is required to process natural languages. Within the Chomskian paradigm, many arguments were given which purported to show that natural languages are not context-free. However, Pullum and Gazdar in 1982 [2] showed that these arguments were invalid. It was not until the mid-eighties that a number of researchers, including Stuart Shieber [3], noticed that there were phenomena in Dutch, and Swiss German, which appeared to exhibit the sort of crossing dependencies which cannot be handled with the stack-like architecture of a push-down automaton (the automata-theoretic equivalent of the context-free grammar).

The addition of the *generalised* composition rules leads to a CCG with greater than context-free power, but still much less powerful than a context-sensitive grammar.[3] Amazingly, it was shown by Weir, Vijay-Shankar and Joshi in the late 80s that a number of formalisms, including CCG and TAG, are weakly equivalent in terms of the languages they can generate. I say “amazingly” because, on the face of it, these formalisms look rather different.

These formalisms have become known as “mildly context-sensitive”. The hypothesis is that mild context-sensitivity is just the right place on the Chomsky hierarchy to be describing natural languages: high enough up that the cross-serial dependency phenomena can be handled, but low enough down that efficient polynomial-time algorithms still exist for these formalisms.

The question of generative capacity for CCG has been revived recently with the work of Kuhlmann, Koller and Satta [1].

Readings for Today’s Lecture The following is not required reading, since these notes should be enough to understand the remaining, more practically-oriented, lectures. However, for those keen to learn more about the linguistic theory, the following is an excellent exposition:

- Combinatory Categorial Grammar (2011), (With Jason Baldridge) Draft 7.0, to appear in: R. Borsley and K. Borjars (eds.) Non-Transformational Syntax, 181-224, Blackwell. Available at http://homepages.inf.ed.ac.uk/steedman/papers.html.

[3]The notion of “much less” can be made mathematically precise on the Chomsky hierarchy; see the work of Weir, Vijay-Shankar and Joshi.

References

[1] Marco Kuhlmann, Alexander Koller, and Giorgio Satta. Lexicalization and generative power in CCG. *Computational Linguistics*, 41(2), 2015.

[2] Geoffrey K. Pullum and Gerald Gazdar. Natural languages and context-free languages. *Linguistics and Philosophy*, 4, 1982.

[3] Stuart M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343, 1985.