

Lecture 15

Syntax as First-Order Logic

We are now operating under the assumption that syntax can be adequately described in terms of regular tree languages. They are sufficiently expressive to capture advanced linguistic concepts like headedness, projection, subcategorization, agreement, adjunction, and instances of displacement like *wh*-movement and topicalization in English. At the same time, they can be automatically converted into CFGs, which makes them suitable for a variety of well-understood parsing techniques. Today we will look once more at what can and cannot be done with regular tree languages, and we will see that, puzzlingly, syntax is both within and outside this class, depending on what kind of object we take syntax to define.

1 Syntactic Constraints

1.1 A General Template for Constraints

The syntactic literature is full of an enormous number of constraints. Some examples include:

- the Empty Category Principle (ECP), which restricts the possible landing sites of movement,
- Binding theory, which restricts the distribution of pronouns (*him*, *her*) and reflexives (*himself*, *herself*),
- the Person Case Constraint (PCC), which blocks certain combinations of pronouns in a sentence,
- the Shortest Derivation Principle (SDP), which favors shorter derivations over longer ones.

This is obviously not a complete list, and linguists keep coming up with new constraints or modify old ones to better fit the data. How could we possibly say that all constraints in natural language syntax fall within the class of regular tree languages?

As with any other empirical science, there is of course the possibility that a new discovery completely contradicts our current theories and requires new, more powerful machinery. But with any new piece of data we discover that matches current proposals, this scenario becomes less and less likely. After 50 years of generative linguistics with many competing formalisms, an abstract template for the formulation of constraints

has crystallized. Constraints that are stated with respect to a single tree usually involves four components:

- a finite number of nodes between which the dependency holds (e.g. the target site and the source of a moving phrase),
- a tree-geometric relation that picks out these nodes (e.g. c-command)
- a locality domain within which the dependency must be satisfied,
- a logical control mechanism that triggers the dependency (e.g. “if X is in position Y, then Z must be satisfied”)

Comparative constraints like the SDP, where the well-formedness of a tree cannot be decided without looking at other trees, do not obviously fit this template, and we will not discuss them here. However, [Graf \(2013\)](#) shows that they, too, can be broken down into templates of this form (using ideas we encountered in the proof that certain variants of OT generate regular string languages).

The existence of such a template for constraints is crucial because if all four factors stay within the realm of regular tree languages, then all constraints that obey this template are regular, too. In the early 90s it was realized that constraints obeying the template can be easily expressed as statements of a formal description language that can be automatically translated into refined strictly 2-local tree grammars, thereby establishing their regularity. This insight forms the backbone of what is now known as *model-theoretic syntax* ([Blackburn et al. 1993](#); [Backofen et al. 1995](#); [Kracht 1997](#); [Rogers 1998](#); [Potts and Pullum 2002](#); [Pullum 2007](#)).

1.2 Constraints, Logics, and Model Theory

If one abstracts away from all matters of implementation, a constraint c over trees is simply a method for defining the largest set of trees that satisfy c , or equivalently, do not violate c . So we can equate every constraint with a (possibly infinite) set of trees.

Something remarkably similar can be found in mathematical logic. There are many different logics, e.g. propositional logic, first-order logic, or modal logic. What makes each one of them a logic is that they are formal systems with a well-defined syntax, which defines what strings are well-formed formulas of the logic, and a semantics that assigns a specific interpretation to each formula.

For first-order logic, the syntax can be defined in a way that’s very close to a context-free grammar. First, we have to fix a vocabulary, also called a *signature*. It includes a finite number of relational symbols R_i^n of arity n , and a set O of logical operands:

\wedge	and
\vee	or
\rightarrow	implies
\leftrightarrow	if and only if
\neg	not
\forall	for every
\exists	there exists

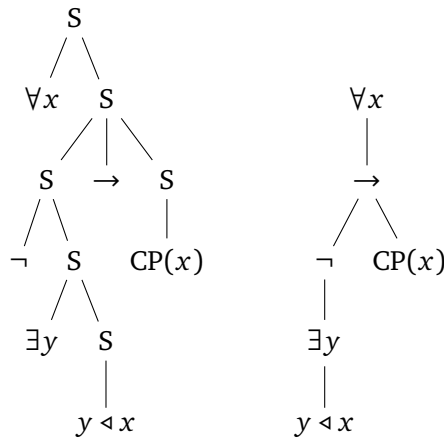
In addition, there is an infinite set of variables x_1, \dots, x_n, \dots , and the bracketing symbols “(” and “)”. The set of well-formed formulas is given by the following rules:

$$\begin{aligned}
S &\rightarrow (S \wedge S) \\
S &\rightarrow (S \vee S) \\
S &\rightarrow (S \rightarrow S) \\
S &\rightarrow (S \leftrightarrow S) \\
S &\rightarrow (\neg S) \\
S &\rightarrow (\forall x_i S) \\
S &\rightarrow (\exists x_i S) \\
S &\rightarrow R_i^n(x_1, \dots, x_n)
\end{aligned}$$

Assuming that our only relational symbols are binary \triangleleft and unary CP, the rules above tell us that the following is a well-formed formula:

$$(\forall x((\neg(\exists y(\triangleleft(y, x)))) \rightarrow \text{CP}(x)))$$

Assuming certain standard conventions about operator scope switching to infix notation for binary relations, we can drop some of the brackets and simplify the formula to $\forall x(\neg \exists y(y \triangleleft x) \rightarrow \text{CP}(x))$, and we can also represent it in terms of a tree (the left format shows the phrase structure, the right one the dependency structure).



Defining the set of well-formed formulas gives us only one half of a logic, the much more important half is to endow each well-formed formula with a specific meaning. This is achieved by the notion of a *model*. A mathematical structure M is a model of formula ϕ (written $M \models \phi$) iff ϕ holds in M under a fixed interpretation. This interpretation is defined in a piece-wise fashion, where $\phi[x \leftarrow u]$ is the result of replacing every free instance of x (= not c-commanded/dominated by $\forall x$ or $\exists x$ in the phrase structure/dependency tree) in ϕ by u .

$$\begin{aligned}
M \models \phi \wedge \psi &\quad \text{iff both } M \models \phi \text{ and } M \models \psi \\
M \models \phi \vee \psi &\quad \text{iff } M \models \phi \text{ or } M \models \psi \\
M \models \phi \rightarrow \psi &\quad \text{iff } M \models \phi \text{ implies } M \models \psi \\
M \models \phi \leftrightarrow \psi &\quad \text{iff } M \models \phi \text{ implies } M \models \psi \text{ and the other way round} \\
M \models \neg \phi &\quad \text{iff } M \models \phi \text{ does not hold} \\
M \models \forall x \phi &\quad \text{iff } M \models \phi[x \leftarrow u] \text{ for all } u \text{ of } M \\
M \models \exists x \phi &\quad \text{iff } M \models \phi[x \leftarrow u] \text{ for some } u \text{ of } M
\end{aligned}$$

In addition, one must also define an interpretation for all relational symbols. For example, if we assume that our class of models is the set of all trees, we may take \triangleleft to denote the mother-of relation, whereas $\text{CP}(x)$ means that node x has the label CP. In

that case, the formula above holds of all trees whose root is labeled CP. Or the other way round, every tree whose root is labeled CP is a model for this formula.

Just like every constraint can be identified with the set of trees that satisfy it, every formula can be identified with its set of models. But this means that we can actually think of constraints in terms of logical formulas: a constraint c is a logical formula ϕ such that the set of structures obeying c is exactly the set of models of ϕ .

Example 15.1 A Formula for Trace Licensing

The ECP states that every trace must be c-commanded by the moving phrase that left behind this trace. We will formalize a simplified variant that ensures the presence of a c-commanding phrase that could be the original mover. To this end, we use first-order logic with a signature that contains \triangleleft^+ for proper dominance, \approx for equivalence, and a unary predicate for every label in our intended tree alphabet. Note that we do not have to add the mother-of relation to the signature because it can be defined in terms of proper dominance:

$$x \triangleleft y \iff x \triangleleft^+ y \wedge \neg \exists z [x \triangleleft^+ z \wedge z \triangleleft^+ y]$$

The same is true of c-command:

$$\text{c-com}(x, y) \iff \neg(x \triangleleft^* y) \wedge \neg(y \triangleleft^* x) \wedge \forall z [z \triangleleft^+ x \rightarrow z \triangleleft^+ y]$$

Here \triangleleft^* denotes reflexive dominance, which is also definable from proper dominance and equivalence.

$$x \triangleleft^* y \iff x \approx y \vee x \triangleleft^+ y$$

So now we have all the predicates we need to talk about the tree geometric configuration between traces and movers, but we still need a way to identify potential movers.

Clearly a phrase is a mover if it occupies a position where it cannot have been selected as an argument. According to the standard theory of movement, movers can only occur in specifiers, which are usually reserved for the second argument of a head. So it suffices to look at the head of the phrase and check whether it selects a second argument (for the sake of simplicity, we will assume that no head has an optional second argument, that phrases obey a strict X' -template where every phrase has at most one specifier, and we also ignore adjuncts, which would be identified as potential movers with this approach).

First we need to define a predicate that identifies whether a node is a specifier. A phrase is a specifier iff it is the sibling of a node labeled X' . Rather than assuming that the trees include this information in their interior node labels, we define a predicate for identifying the X' -node projected by a lexical item, which is then referenced in the definition of the predicate for specifiers.

$$\begin{aligned}\text{bar}(x, y) &\iff \exists z[x \triangleleft z \wedge z \triangleleft y \wedge \bigvee_{l \text{ a lexical item}} l(y)] \\ \text{sibling}(x, y) &\iff \exists z[z \triangleleft x \wedge z \triangleleft y] \\ \text{spec}(x, y) &\iff \exists z[\text{sibling}(x, z) \wedge \text{bar}(z, y)]\end{aligned}$$

Now we can finally define a predicate to identify moving phrases as specifiers of a head that takes no second argument.

$$\text{mover}(x) \iff \exists y[\text{spec}(x, y) \wedge \bigvee_{l \text{ takes at most one argument}} l(y)]$$

In the last step we require every trace to be c-commanded by a phrase.

$$\forall x[t(x) \rightarrow \exists y[\text{mover}(y) \wedge \text{c-com}(y, x)]]$$

Note that this is an actual formula, whereas all the previous formulas are just definitions that define predicates as a shorthand for specific subformulas that we use in definition of the ECP constraint.

Our implementation of the ECP leaves a lot to be desired, of course (a more adequate implementation is given in [Rogers 1998](#), a logical formalization of GB that takes up over 150 pages). One glaring omission is that a mover cannot have left a specific trace behind if its category feature does not fit the position in which the trace occurs, for in that case the mover cannot have originated there. This can be added to the formula above by defining a predicate that checks what type of category is required for the position occupied by a trace and checks that the mover has this category — a rather simple exercise. A more interesting aspect is that movement is usually assumed to be locally bounded, which means that the licenser of a trace must occur within a specific domain, e.g. the smallest CP containing the trace. Such domain restrictions are very common across linguistic constraints, but fortunately they pose no challenge to first-order logic.

$$\begin{aligned}\text{smallest}(x, y, ZP) &\iff ZP(y) \wedge y \triangleleft^+ x \wedge \neg \exists z[ZP(z) \wedge y \triangleleft^+ z \wedge z \triangleleft^+ x] \\ \text{local}(x, y, ZP) &\iff \exists z[\text{smallest}(x, z, ZP) \wedge \text{smallest}(y, z, ZP)] \\ \forall x[t(x) &\rightarrow \exists y[\text{mover}(y) \wedge \text{c-com}(y, x) \wedge \text{local}(x, y, \text{CP})]]\end{aligned}$$

While the example above implements a specific constraint, it is easy to see that the tricks and techniques work for virtually all linguistic constraints. Lexicalization and projection means that all local information can be inferred from the head of a phrase, and we can keep track of this information via newly defined predicates. More complex tree geometric notions like c-command or m-command are easily defined

in terms of dominance, and the same holds for the notions of closeness and locality domain that are ubiquitous in linguistics. With a little bit of ingenuity, pretty much all syntactic constraints can be expressed in first-order logic.

First-order logic is a fragment of monadic second-order logic (MSO), which enriches first-order logic with the option to quantify over sets of nodes. It turns out that MSO is exactly as powerful as refined strictly 2-local tree grammars, so one can freely translate between grammars and MSO formulas (Büchi 1960). The construction is fairly involved (see Morawietz 2003 for details), and the computational complexity of the translation is unbounded — it doesn't get any less tractable than that. Nonetheless the conversion is reasonably fast in practice, and the fact that linguistic constraints can be stated without set quantification reduces the complexity a lot. Overall, then, it is a viable strategy to formalize linguistic theories in terms of first-order formulas, which are then automatically translated into refined tree grammars or possibly even CFGs.

Conjecture 15.1 (FO-Syntax). *All aspects of natural language syntax can be expressed in terms of first-order formulas over trees without interior node labels.*

2 Pushing the Limits

2.1 Polarity Items

There are a few phenomena that push the limits of what can be expressed in first-order logic. One of them is the distribution of so-called *polarity items*, or at least their predicted distribution according to standard generalizations. Polarity items can only occur in a context of a specific polarity. Negative polarity items (NPIs) are restricted to sentences where they appear in the scope of some negative element, though it is difficult to characterize what counts as such a negative element, as the examples below illustrate for the idiomatic NPI *lift a finger*:

- (1) a. Nobody did so much as lift a finger.
- b. * Everybody did so much as lift a finger.
- c. You didn't so much as lift a finger.
- d. * You did so much as lift a finger.
- e. You never did so much as lift a finger.
- f. You hardly did so much as lift a finger.
- g. * You always did so much as lift a finger.

Negation and negative quantifiers license this NPI, but so do adverbials like *never* and *hardly*. The licensing element can also occur in a higher clause, in which case it may be a verb like *doubt*.

- (2) a. Nobody believes you did so much as lift a finger.
- b. * I believe you did so much as lift a finger.
- c. I doubt you did so much as lift a finger.

The standard explanation is in term of semantic entailment patterns, with NPIs like *lift a finger* and *ever* restricted to downward entailing patterns, while positive polarity items (PPIs) like *somewhat* are restricted to contexts that are not downward entailing. The specifics of this proposal are not all that relevant to us except for one crucial property: negation switches downward entailing contexts into non-downward

entailing ones, and *vice versa*. So if an NPI is licensed in configuration C , it should not be licensed in $\neg C$, but be licensed again in $\neg\neg C$. If this is indeed the correct analysis, then the distribution of NPIs and PPIs cannot be defined in first-order logic. That's because we are dealing with a *modulo* counting pattern, similar to what we saw for the string language $(aa)^+$, and these patterns require the full power of regular languages. Since first-order logic is a fragment of MSO, it does not have access to this additional power and thus cannot express the semantic generalization.

But whenever one is faced with generalizations that need more power than expected, the first step of action is to verify that I) the generalization is empirically correct, and II) there isn't a structurally simpler generalization that covers the same data. In the case of NPI and PPI licensing, the predicted pattern of alternating licensing contexts simply does not obtain.

- (3) a. I doubt you did so much as lift a finger. (predicted grammatical)
- b. ? I doubt you didn't do so much as lift a finger. (predicted ungrammatical)
- c. ? Nobody doubts you did so much as lift a finger. (predicted ungrammatical)
- d. ?? Nobody doubts you didn't do so much as lift a finger. (predicted grammatical)

In addition, NPIs are frequently accepted in contexts that look downward entailing but actually aren't.

- (4) Every politician that no news paper reported on in great detail has ever made it past the first round.

These effects are usually attributed to processing errors, but in combination with the other facts they cast further doubt on the canonical analysis. Perhaps the distribution of NPIs and PPIs is subject to a first-order definable constraint after all, a constraint that looks quite a bit different from the semantic generalization simply due to the fact that first-order logic is incapable of expressing such a generalization.

2.2 Binding Theory

Binding theory regulates the distribution of elements whose reference depends on some other element in the sentence. These usually take the form of reflexives (*himself*, *herself*, *itself*), pronouns (*her*, *him*, *it*), and reciprocals (*each other*). The involvement of reference and meaning turns binding theory into a very wide and vaguely circumscribed phenomenon that spans syntax, semantics, and pragmatics. How exactly it can and should be factored across these components is difficult to decide, and as we will see next, the FO-syntax conjecture hinges on what kind of factorization one assumes.

The canonical view of binding theory is that DPs and referentially dependent elements are assigned referential indices as in the examples below.

- (5) a. John_i hit himself_i.
- b. John_i threatened Bill_j to hit him_j.
- c. * John_i hit him_i.
- d. * Mary_i hit himself_j.

Sentences are then filtered out according to whether these indexations satisfy certain principles. The exact nature of said principles vary across languages, but we can identify to macro-parameters (cf. [Kiparsky 2002, 2012](#)).

± **s-bound** Does the referentially dependent element need a *syntactic* antecedent within a certain locality domain? (rather than just a previously established discourse referent)?

± **obviative** Must the index of the referentially dependent element be unique within a certain syntactic locality domain?

English *himself* for instance, must be s-bound within the smallest TP containing a subject, but it is not obviative. English *him*, on the other hand, is more ambiguous.

In general, *him* needs no syntactic antecedent, but it does presuppose a previously established discourse referent. This requirement is lifted when the pronoun is used *deictically*, e.g. by pointing at somebody in the room. Obviation is a little trickier to determine. The sentence *Gatsby_i is much older than him_i* seems perfectly fine when the speaker, mistaken about Gatsby's appearance, is unknowingly pointing at Gatsby. At the same, it is unclear that this sentence actually involves two identical indices, rather than two distinct indices (= distinct people in the speaker's mind) that happen to map to the same person in the real world. Excluding these special cases, though, *him* is obviative with the smallest TP containing a subject, which is witnessed by the fact that *John_i hit him_i* cannot mean the same thing as *John hit himself*.

See Heim (1998) for a detailed discussion of such sentences.

Other languages allow for different combinations. The Icelandic reflexive *sig*, for example, mirrors *himself* in that it is not obviative, but at the same time its locality domain for s-binding is much larger. Swedish *sig* and Marathi *aapan* have a similar s-binding requirement within a large locality domain but in addition they are obviative within the smallest TP containing a subject.

Given such a tremendous amount of cross-linguistic variation, one might be rightfully worried about the feasibility of a first-order formalization of binding theory. These worries are justified, but the culprit isn't variation; it is indexation. By assumption, languages are infinite and the length of sentences is unbounded. Consequently, there is no upper limit on how many referents may be mentioned in a sentence, which in turn implies that there is no finite bound on the number of indices we need — for every sentence with one more DP we may have to add one more index. This is obviously a problem given our assumption that alphabets must be finite, but there are ways to encode indices without putting them in the alphabet — for example, if all indices are natural numbers then *John_n* could be represented as a node *John* whose sister is the root of a unary branching tree of depth *n*. But Rogers (1998) shows that no matter how indices are represented, the free-indexation system at the core of binding theory increases the power of the system so much that it becomes *undecidable*. That is to say, it is no longer possible to determine for arbitrary, freely indexed trees whether they are well-formed or not. So binding theory as commonly envisioned isn't just a problem for first-order logic, it is an insurmountable task for all computational machinery.

Of course this formal result seems at odds with our intuition that speakers do not have a hard time determining referents, so either all real-world problems simply happen to be the most trivial instances of an enormously complex system, or free indexation isn't the right way to think about binding theory — or possibly both. Index-free theories of binding theory have been developed in the meantime (Bonato 2005; Kobele 2006), defusing the problem to some extent. This is no guarantee, though, that binding theory is first-order definable. In order to show that, we have to go one step further.

As mentioned at the outset of this section, binding theory combines three different mechanisms:

- a syntactic mechanism that regulates whether specific lexical items may occur in a given context,
- a semantic mechanism that links the interpretation of referentially dependent elements to antecedents in the structure or the discourse,
- a discourse mechanism that keeps track of referents and uses pragmatic principles to help in discourse resolution.

The FO-conjecture is about the complexity of syntax, not semantics or pragmatics. So only the syntactic part of binding theory can furnish a licit counterexample. This syntactic part does not have to pay attention to specific meanings, since those are established by semantics. Instead, it only has to ensure that the distributional requirements are satisfied, which is the case as long as there is at least one possible reading. In other words, syntax only has to rule out sentences where the referentially dependent elements are distributed in such a fashion that it is impossible to assign a meaning according to the laws of binding theory.

With this shift in perspective, binding theory becomes a lot simpler at the syntactic level. Determining that s-binding is satisfied isn't all that different from our constraint earlier on that every trace needs to be c-commanded by a mover. The definition of locality differs quite a bit, and what counts as a valid antecedent is more restricted (*Mary* cannot be an antecedent of *himself* due to gender mismatch), still it all fits comfortably within the bounds of first-order logic. Obviation is a lot harder. Due to the unbounded nature of various constructions, it is theoretically possible to have an unbounded number of obviate elements within the same obviation domain. Now it necessarily holds that if there are n obviate elements within the same obviation domain, then there must be at least n possible antecedents outside the obviation domain. You might remember that this kind of unbounded counting cannot be achieved with finite-state methods, which are equivalent to MSO, which subsumes first-order logic. So first-order logic cannot even handle syntactic binding theory unless the following additional restriction holds in every natural language:

This is a necessary condition, not a sufficient one. We ignore the extra conditions here for the sake of exposition. See [Graf and Abner \(2012\)](#) for a full discussion.

Limited Obviation There is an upper bound k such that if an obviation domain contains $n > k$ obviate elements, k antecedents suffice to furnish a grammatical reading.

Limited Obviation thus posits that some obviate elements within the same obviation domain can have the same referent, and that only a fixed number of antecedents are ever needed thanks to this potential overlap.

Limited Obviation is not the kind of condition linguists would come up with, it follows strictly from our search for a restriction that renders binding theory first-order definable. This makes it even more surprising that Limited Obviation seems to be an actual language universal, as is argued by [Graf and Abner \(2012\)](#) based on a careful examination of a variety of unbounded constructions. So even though binding theory is a tight fit, it does stay within the bounds of first-order logic.

2.3 Copy Movement

3 Towards a Context-Free Solution