Boolean Semantics

Topics: cross-categorial expressions, boolean algebras, the boolean hypothesis on natural languages, boolean types

In the previous talks we have often used sentences with predicative constructions involving the expressions *and, or, not* and *neither...nor* as illustrations for entailments that motivate a modeltheoretic analysis. The accounts we gave of such entailments, however, have systematically missed an obvious generalization: coordination and negation are of course not confined to simple intransitive predicates. These are general mechanisms in natural languages that can apply to different categories.

Not only does the syntax of these constructions involve systematic regularities across categories, also their semantics shows interesting entailment patterns. For instance, entailments we discussed like (1a)⇒(1c-d) or (1c-1d)⇒ (1b), with predicate coordinations, have natural correspondences with coordinations of other categories as in (2)-(4).

(1)
  a. Tina is tall and thin.
  b. Tina is tall or thin.
  c. Tina is tall.
  d. Tina is thin.

(2)
  a. Tina is tall and Mary is thin.
  b. Tina is tall or Mary is thin.
  c. Tina is tall.
  d. Mary is thin.

(3)
  a. Tina and Mary are tall.
  b. Tina or Mary is tall.
  c. Tina is tall.

d. Mary is tall.

(4) a. Tina kissed and hugged Mary.

b. Tina kissed or hugged Mary.

c. Tina kissed Mary.

d. Tina hugged Mary.

Not surprisingly, cross-categorial entailment relations appear also between *neither...nor* and *not*, as exemplified by the (a)⇒(b)/(c) entailments in (5)-(8).

(5) a. Tina is neither tall nor thin.

b. Tina is not tall.

c. Tina is not thin.

(6) a. Neither Tina is tall nor is Mary.

b. Tina is not tall.

c. Mary is not tall.

(7) a. Neither Tina nor Mary is tall.

b. Tina is not tall.

c. Mary is not tall.

(8) a. Tina neither kissed nor hugged Mary.

b. Tina did not kiss Mary.

c. Tina did not hug Mary.

Warning: Beware of the hasty conclusion that these patterns are general rules. For instance, in (1), replacing *Tina* by the noun phrase *no girl* would make the entailment (1a)⇒(1c) disappear: the sentence *no girl is tall and thin* does *not* entail *no girl is tall*. Semantic differences between noun phrases that are responsible for such differences will be explained in the next lecture.

In view of the appearance of coordinators (and to a lesser extent, of negation particles) with different categories, a natural expectation from any syntactic theory is to provide an account of what is common to different structures in which these expressions partake. Most existing syntactic theories of coordination start from a syntactic meta-rule parallel to (9), for all categories X.

(9) X $\rightarrow$ X coordinator X

A unified syntactic-semantic theory should explain how rules such as this get a semantic interpretation: How does the denotation of the coordinator combine with the denotations of different categories? How should we account for the cross-categorial semantic regularities we have observed?

Boolean Semantics is a framework of modeltheoretic semantics that attempts to answer these questions. Given the model structure we have outlined in the previous lecture, there is a large group of domains in the model that share a particular mathematical structure, known as a *Boolean Algebra*. It turns out that once this fact is observed, coordinators and negation particles can be given an adequate semantics that holds for different domains. Moreover, because of the similar structure of these domains, conjunction, disjunction and negation behave the same across categories. In order to see how this comes about, we have to say more on what is common to different domains: to explicate a bit the notion of boolean algebras.

We will not go deeply here into the mathematical theory of boolean algebras. Rather, let us informally illustrate how observing the common, so-called "boolean" structure of different domains can help us to account for the cross-categorial semantic behavior of coordination and negation. To demonstrate the general idea, let us start with the familiar domain of intransitive predicates. As we have said, this domain corresponds to all the subsets of the E domain of individuals: the power set of E, for which we use the notation $\wp(E)$. For instance, when $E = \{a, b, c\}$ we have:

$$\wp(E) = \wp(\{a, b, c\}) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\}$$

There are some interesting properties to this set:

1. It is partially ordered by set inclusion, as illustrated in figure 1.

2. It includes a "smallest set" with respect to inclusion: the empty set $\emptyset$.

3. It includes a "largest set" with respect to inclusion: the whole domain $E = \{a, b, c\}$.

4. For every two sets $A, B \in \wp(E)$, their intersection $A \cap B$ is in $\wp(E)$.

5. For every two sets $A, B \in \wp(E)$, their union $A \cup B$ is in $\wp(E)$.

6. For every set A in $\wp(E)$, its complement $\overline{A} = E \setminus A$ is in $\wp(E)$.

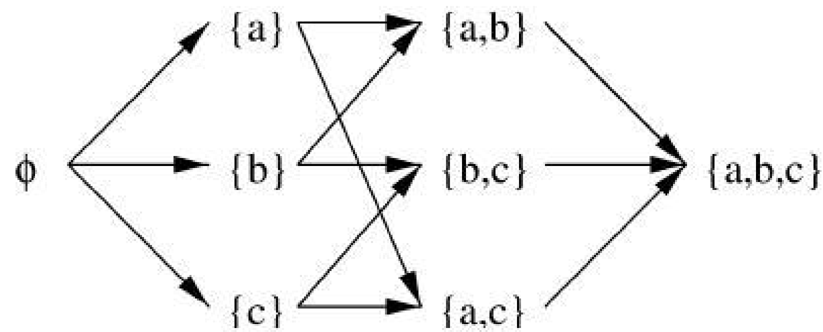

Figure 1: boolean order of $\wp(\{a, b, c\})$

To sum up, $\wp(E)$ "comes" with the tuple $\langle \subseteq, \emptyset, E, \cap, \cup, ^{-} \rangle$ that defines special relations between members of $\wp(E)$. The mathematical beauty of this structure comes from the intimate relations between these items and operations. Here are only a few of the most central relations:

- $A \cap B$ is the (unique) $X \in \wp(E)$ s.t. $X \subseteq A$, $X \subseteq B$ and for every $Y \in \wp(E)$ that satisfies $Y \subseteq A$ and $Y \subseteq B$: if $X \subseteq Y$ then $X = Y$.
  For this reason $A \cap B$ is called the *greatest lower bound* of A and B with respect to inclusion: it is the "largest" set that is a subset of both A and B.

- $A \cup B$ is the (unique) $X \in \wp(E)$ s.t. $A \subseteq X$, $B \subseteq X$ and for every $Y \in \wp(E)$ that satisfies $A \subseteq Y$ and $B \subseteq Y$: if $Y \subseteq X$ then $X = Y$.
  For this reason $A \cup B$ is called the *least upper bound* of A and B with respect to inclusion: it is the "smallest" set that is a superset of both A and B.

Due to these to properties, intersection and union can be defined using inclusion and vice versa. Other important identities are:

- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
  $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
  These two identities are called the *distributive law*.

- $A \cap \overline{A} = \emptyset \quad A \cup \overline{A} = E$ (the complement law)

These relations together characterize the domain of intransitive predicates $\wp(E)$, together with the tuple $\langle \subseteq, \emptyset, E, \cap, \cup, ^{-} \rangle$ as a boolean algebra.

Important note: Although any power set with the mentioned operations is a boolean algebra, the converse does not hold. The notion of boolean algebra is more abstract and general than the power set construction. Keep this point in mind and refer to exercises for the definition of type-theoretical domains as boolean algebras. These are examples for boolean algebras that are not power sets, although they are intimately related to power sets through the sets that their function members characterize.

Many useful domains besides the domain of intransitive predicates are boolean algebras too. For instance, the domain of truth-values $\{0, 1\}$ can be easily presented as boolean structure. One way to do that is to replace the arbitrary names 0 and 1 by the empty set $\emptyset$ for *false* and some singleton set $\{x\}$ for *true*. Obviously, the domain $\{0, 1\} = \{\emptyset, \{x\}\}$ is the power set $\wp(\{x\})$ of the singleton set $\{x\}$, since it includes all subsets of $\{x\}$: the empty set and $\{x\}$ itself. This is not the only way to define a boolean structure over the D_t domain, but it is straightforward as it lets D_t be a power set that has the same properties with respect to inclusion, intersection, union and complementation that we considered above. Note that unlike the case of the predicate domain, now we have in the algebra *only* the smallest member $\emptyset$ and the largest member $\{x\}$ and no "intermediate" members inbetween. Indeed, D_t has the structure of the smallest non-trivial boolean algebra.

Let *and* and *or* have in the domain of truth values the same function they had in the et domain: intersection and union respectively. The sentences in (2) get the denotations in (2') below.

(2') a. $\mathbf{tall}'(\mathbf{tina}') \cap \mathbf{thin}'(\mathbf{mary}')$

b. $\mathbf{tall}'(\mathbf{tina}') \cup \mathbf{thin}'(\mathbf{mary}')$

c. $\mathbf{tall}'(\mathbf{tina}')$

d. $\mathbf{thin}'(\mathbf{mary}')$

The entailments in (2) are now easily accounted for: it is clear that whenever (2'a) is $\{x\}$ (*true*) so are (2'c) and (2'd). Otherwise, suppose (2'c) for instance could be $\emptyset$ (*false*). Then (2'a) would be $\emptyset \cap \mathbf{thin}'(\mathbf{mary}') = \emptyset$, in contradiction to our assumption that (2'a) is $\{x\}$. Similarly, whenever (2'c)/(2'd) is $\{x\}$, so is (2'b). Thus, we account for the entailments observed in (2).

We see that using the boolean perspective on the domains for predicates and truth-values we can let the *and* and *or* coordinators *uniformly* denote the intersection and union functions respectively. Similar points, with some syntactic complexities, hold for negation. Boolean semantics adopts the following generalization of these observations.

The boolean hypothesis: *All domains for denotations of expressions in natural language are boolean algebras. Coordination and negation cross-categorially denote the corresponding boolean operators in each domain.*

This means that the denotation of coordinators and negation particles is not stipulated *ad hoc* for each category in which they occur. There is a uniform semantic function for these expressions across all the different domains in which they can operate. More can be said, in fact. Having stipulated that the D_t domain has the structure of a (minimal) boolean algebra, we can deduce the boolean structure of many other domains directly from their type-theoretical definition as given in the previous lecture. For instance, the boolean structure of D_{et} appears because it is defined as a set of functions into a boolean domain, D_t. In general, all domains of functions with a boolean range are naturally defined as boolean algebras. We formally characterize the boolean types as follows.

Definition 1 (boolean types) *Type t is boolean, and any type $\alpha\beta$ is boolean iff β is boolean.*

Examples:

- Boolean types: $t,\ et,\ (et)(et),\ e(et),\ (ee)t$
- Non-boolean types: $e,\ ee, (et)e$

Intuitively, all types that "end with t" are boolean and those that "end with e" are non-boolean.

Definition 2 (polymorphic boolean operators) *Let τ be a boolean type. Let $\wedge_{t(tt)}$, $\vee_{t(tt)}$, $\neg_{tt}$ and $\rightarrow_{t(tt)}$ be the standard propositional functions. Denote:*

$$\sqcap_{\tau(\tau\tau)} \quad = \begin{cases} \wedge_{t(tt)} & \text{if } \tau = t \\ \lambda X_\tau.\lambda Y_\tau.\lambda Z_{\sigma_1}.X(Z) \sqcap_{\sigma_2(\sigma_2\sigma_2)} Y(Z) & \text{if } \tau = \sigma_1\sigma_2 \end{cases}$$

$$\sqcup_{\tau(\tau\tau)} \quad = \begin{cases} \vee_{t(tt)} & \text{if } \tau = t \\ \lambda X_\tau.\lambda Y_\tau.\lambda Z_{\sigma_1}.X(Z) \sqcup_{\sigma_2(\sigma_2\sigma_2)} Y(Z) & \text{if } \tau = \sigma_1\sigma_2 \end{cases}$$

$$\neg_{\tau\tau} \quad = \begin{cases} \neg_{tt} & \text{if } \tau = t \\ \lambda X_\tau.\lambda Z_{\sigma_1}.\neg_{\sigma_2\sigma_2}(X(Z)) & \text{if } \tau = \sigma_1\sigma_2 \end{cases}$$

$$\sqsubseteq_{\tau(\tau t)} \quad = \begin{cases} \rightarrow_{t(tt)} & \text{if } \tau = t \\ \lambda X_\tau.\lambda Y_\tau.\forall Z_{\sigma_1}[X(Z) \sqsubseteq_{\sigma_2(\sigma_2 t)} Y(Z)] & \text{if } \tau = \sigma_1\sigma_2 \end{cases}$$

$0_\tau = X \sqcap \neg X$, *for arbitrary* X_τ

$1_\tau = X \sqcup \neg X$, *for arbitrary* X_τ

The notations $X \sqcap Y$*,* $X \sqcup Y$ *and* $X \sqsubseteq Y$ *are "sugarings" for* $(\sqcap(X))(Y)$*,* $(\sqcup(X))(Y)$ *and* $(\sqsubseteq(X))(Y)$ *respectively.*

Examples: apply these definitions to selected cases of (1)-(8) above.

The examples above have illustrated the relations between boolean operators in the corresponding domains, and not much more than this is required for our objectives in this course. Note however, that now it is clear why domains like D_e are not classified as boolean domains. The reason is that no boolean structure is projected on this domain from the structure we have assumed for D_t. This shows an interesting (and debatable) conclusion from the boolean hypothesis: all non-boolean types, although sometimes simple and plausible, are *not* straightforward candidates for being types of natural language expressions. Their postulation does not support the simple and unified account of conjunction, disjunction and negation that was illustrated above. Therefore, we have a strong theoretical reason to avoid them.

We seem to be running into a problem. The boolean hypothesis flatly contradicts our assumption that proper names denote e type entities. How do proper names as in (3) get interpreted and coordinated after all? In the next lecture we will see how the boolean hypothesis can be maintained in a way that not only answers this question, but also gives an insightful account of the semantics of noun phrases in general.