# From Graphs to Logic for representing the semantics of natural languages

Guy Perrier

# FROM GRAPHS TO LOGIC FOR REPRESENTING THE SEMANTICS OF NATURAL LANGUAGES

GUY PERRIER

## 1. The principles

(1) Linguistics is an experimental science: the theory must be confronted with the reality to be validated and corpora are the privileged material for making experiments.
(2) The confrontation between the theory and the reality through corpora needs large scale linguistic resources and software able to deal with such resources.
(3) From this point of view, the simplicity and the readability of theoretical models is decisive.
(4) The base for representing the sense of utterances extracted from corpora is to model the dependencies between semantemes that directly come from the syntax. The way of doing this is to use directed acyclic graphs (DAGs)[1]. We call them *semantic graphs*.
(5) Not all semantic data can be deduced from the syntax (quantifier scopes, anaphoric links, ...), which produces a certain form of semantic ambiguity. Semantic graphs must take it into account under the form of additional information. In the present work, we are only interested in *scope constraints*.
(6) From semantic graphs and scope constraints, using general and specific rules, we automatically deduce a first order logic formula which represents one interpretation of the source utterance. Other interpretations come from a change either in semantic graphs or in scope constraints.

## 2. The form of semantic graphs

(1) A semantic graph is a DAG with vertices and edges respectively representing *semantemes* and *semantic roles*.
(2) A vertex $N$ is labelled with a *name*, denoted *name(N)*, and every name has a *type* chosen among three types:
    - *P for predicates:* a predicate may take from 0 to n arguments to become a proposition which can be true or false; a vertex labelled with a predicate is called a P-vertex. The arguments of a P-vertex $N$ are represented by its direct

[1]Verify that cycles are not useful in the representation of semantics.

successors[2] and their set is denoted *arg(N)*. To distinguish their roles in the predicate, their incident edges are labelled with positive integers: 1, 2, 3 ...

- *SP for scopal predicates:* a scopal predicate is a predicate with exactly one argument and its specificity is that quantifier scopes can be inserted between the predicate and its argument. A vertex labelled with a scopal predicate is called a SP-vertex. A SP-vertex $N$ is a source vertex[3] with exactly one direct successor representing its argument. This argument is a P-vertex or a SP-vertex, and it is denoted *mod(N)*.
- *Q for quantifiers:* a quantifier needs two properties as arguments, its *restriction* and its *field*[4]. The restriction defines a set of individuals on which the field applies via a specific quantification relationship. A vertex labelled with a quantifiers is called a Q-vertex. A Q-vertex $N$ has exactly one direct successor *restrict-kern(N)*, which is a P-vertex and which represents the *kernel of its restriction*. The whole restriction of Q-vertex is computed deterministically. The field is also computed but there is a part of non determinism in this computation.

The name "?" is reserved to represent a non specified predicate. For any node $N$, the type of its name is denoted *type(N)*. A vertex that is either a Q-vertex or a SP-vertex is called a S-vertex.

(3) Well-formedness constraints :
- Every P-vertex has one argument at most that is P-vertex without arguments.
- Every P-vertex without arguments has exactly one Q-vertex as a direct predecessor.
- Every P-vertex with arguments has one Q-vertex as a direct predecessor at most[5].

## 3. Scope of quantifiers and scopal predicates

(1) The *restriction* of a Q-vertex $N$, denoted *restrict(N)*, is the subgraph constituted of the vertices and edges belonging to chains that start from *restrict-kern(N)* and that do not cross Q-vertices taking their adjacent edges in the opposite direction[6].

[2]When an argument of a P-vertex $N_{pred}$ represents a quantified expression, we choose the Q-vertex $N_q$ as the argument and not its restriction kernel *restrict-ker($N_q$)*. We choose *restrict-ker($N_q$)* as the argument of $N_{pred}$ to express that $N_{pred}$ is used in the restriction of the quantifier. Such choice differs from the classical representation of quantifiers in dependency grammars.

[3]A source vertex in a graph is a vertex without incident edge.

[4]In most works, this second property is called the *scope* of the quantifier, but we reserve the word *scope* to its classical meaning in first order logical formulas.

[5]In this it represents a reified predicate: the predicate is taken as a class of individuals.

[6]The connected component of *restrict-kern(N)* in the graph without the edge from $N$ to *restrict-kern(N)* is not a correct definition. See the counter-example of *every farmer who owns a donkey beats it*. In the trip for exploring the restriction, a forbidden step from a Q-vertex to a predecessor P-vertex means that a reference of the restriction is re-used outside this restriction. A forbidden step from a P-vertex to a predecessor Q-vertex means that two quantifiers share the same restriction: the semantic graph is ill-formed.

The *source of the restriction* is the set of the vertices that are considered as sources in the graph of the restriction.

(2) *Exclusivity of restrictions* : in a well-formed semantic graph, any subgraph is the restriction of one Q-vertex at most. This property can be verified by computing the restriction of Q-vertices : when we build a chain from the restriction kernel of a Q-vertex, if we reach another restriction kernel, then we can conclude that the restriction is shared by two Q-vertices, which is incorrect.

(3) The *minimal field* of a Q-vertex $N$, denoted *min-field(N)* is the subgraph constituted of all vertices and edges reachable from the direct predecessors of $N$ without crossing the edge from $N$ to *restrict-kern(N)*. It is augmented with the restrictions of the reachable Q-vertices. If the minimal field is considered as an own graph, its source vertices constitute the *source* of the minimal field.

(4) For a Q-vertex $N$, the union of *restrict(N)*, *field(N)* and the edge from $N$ to *restrict-kern(N)* constitutes its *minimal scope* and it is denoted *min-scope(N)*.

(5) The *minimal scope* of a SP-vertex $N$ is constituted of all vertices and edges that are reachable from $N$. It is augmented with the restrictions of the reachable Q-vertices and it is denoted *min-scope(N)*.

(6) In a semantic graph, there are three kinds indeterminacy:
- in the definition of the SP-vertex scopes,
- in the definition of the Q-vertex fields,
- in the relationship between different scopes.

Indeterminacy is removed by adding *scope constraints*: the first one is solved by fixing the SP-vertex scopes and the two last ones by defining a partial order between S-vertices.

(7) For every SP-vertex $N$, its *scope*, denoted *scope(N)*, is defined from the choice of its *source*. The source is constituted of *mod(N)* and some vertices chosen among the sources of the whole graph with the following property: *mod(N)* is reachable from each of them without crossing any Q-vertex. Then, *scope(N)* is the subgraph constituted of all vertices and edges reachable from vertices of the source. It is augmented with the restrictions of the reachable Q-vertices.

(8) We define a partial order of *government* between S-vertices that respect the following conditions :
- if a SP-vertex governs another node, then the second one must belong to its scope;
- if the scopes of two S-vertices are not disjoint[7], one of the vertices must govern the other.

(9) From the government relation, we define the *field*, denoted *field(N)*, of every Q-vertex $N$ recursively according the following rules:
- If $N$ directly governs a SP-vertex that belongs to its restriction, then *field(N)* is equal to *min-field(N)*.

[7]If the vertex is a Q-vertex,the scope that is considered here is the minimal scope.

- If $N$ directly governs a SP-vertex that does not belong to its restriction, then *field(N)* is the union of its minimal field and the scope of the SP-vertex that it governs.
- If $N$ directly governs a Q-vertex $M$, then *field(N)* is the union of *min-field(N)* and the part of *field(M)* that is outside *restrict(N)*.

The *source* of *field(N)* is defined as the set of the vertices that are sources of *field(N)* considered as a graph. The *scope* of $N$, denoted *scope(N)*, is the subgraph that is the union of *restrict(N)*, *field(N)* and the edge from $N$ to *restrict-kern(N)*.

## 4. The automated generation of the logical form from semantic graphs

(1) Every vertex $N$ is associated with its logical interpretation $F_N$, which is a function computing a logical formula from specific arguments[8].
(2) Every Q-vertex $N$ is associated with a unique variable denoted $x_N$. The variable represents any instance of its restriction [9].
(3) Then, we build a logical formula for every vertex $N$. This formula represents the *logical form* of $N$ and it is denoted *logic-form(N)*. We build *logic-form(N)* recursively, following the direction opposite to the edges of the graph and applying the following rules:
   - If $N$ is a P-vertex with its arguments $N_1, \ldots, N_p$, then we define its *primitive form* as the logical formula that interprets its when we forget that $N$ is in the source of the field of a possible Q-vertex. We denote it *primitive-form(N)*. If $N$ is not quantified, that is not the direct successor of a Q-vertex, then *primitive-form(N)* $= F_N(t_1, \ldots, t_p)$, where the terms $t_i$ are defined as follows:
     - if $N_i$ is a Q-vertex, then $t_i = x_{N_i}$.
     - if $N_i$ is quantified, that is the direct successor of a Q-vertex $M_i$, then $t_i = x_{M_i}$.
     - otherwise, $t_i =$ *logic-form*$(N_i)$.

     If $N$ is quantified, that is the direct successor of a Q-vertex $M$, then *primitive-form(N)* $= F_N(x_M, t_1, \ldots, t_p)$, with the same definition for $t_i$ as in the previous case.
   - If $N$ is a P-vertex, which is not in the source of the field of some Q-vertex, then *logic-form(N) = primitive-form(N)*.
   - If $N$ is a P-vertex, which is in the source of the field of some Q-vertex, then *logic-form(N) = logic-form(M)*, $M$ being the Q-vertex that has $N$ in the source of its field and that governs all Q-vertices with the same property.

[8]Usually, $F_N$ depends only on *name(N)*. In this case, we also write the function $F_{name(N)}$. Consider for instance indefinites, which are interpreted with the quantifier *a*. In most cases, $F_a = \lambda(P, Q)\ .\lambda x.P\ x \wedge Q\ x$, but in some cases, such as donkey sentences, the function depends on the context of the vertex $N$. The interest of using semantic graphs is to facilitate the computation of $F_N$. Most times, $F_{name(N)}$ reduces to a constant that identifies to *name(N)*, but sometimes, it is defined in a more sophisticated way (for instance the adjective *last*).

[9]If the restriction kernel itself is a predicate with arguments, the variable is used to reify the argument predicate.

- If $N$ is a SP-vertex, and if $N_1$, ..., $N_p$ constitute the source of its scope, then *logic-form(N)* = $F_N$ *(*$\bigwedge_1^p$ *logic-form(*$N_i$*))*.
- If $N$ is a Q-vertex, then *logic-form(N)* = $F_N$ *(*$Form_{restrict}$*,* $Form_{field}$*)*, where $Form_{restrict}$ and $Form_{field}$ are defined as follows[10]:
  - We assume that the source of *restrict(N)* is constituted of $N_1$, ..., $N_p$, then two cases have to considered:
    - if there is no SP-vertex among $N_1$, ..., $N_p$, then $Form_{restrict}$ = *logic-form(restric-kern(N))* $\wedge \bigwedge_{i=1}^p$ *logic-form(*$N_i$*)*.
    - if there is a SP-vertex among $N_1$, ..., $N_p$, then $Form_{restrict} = \bigwedge_{i=1}^p$ *logic-form(*$N_i$*)*.
  - In order to compute $Form_{field}$, two cases have to be considered:
    - *N directly governs no other Q-vertex:* If the source of *field(N)* is constituted of $N_1$, ..., $N_p$, then $Form_{field} = \bigwedge_{i=1}^p$ *primitive-form(*$N_i$*)*.
    - *N directly governs another Q-vertex* M*:* if $N_1$, ..., $N_p$ are the elements of the source of *field(N)* that do not belong to *field(M)*, then $Form_{field}$ = *logic-form(M)* $\wedge \bigwedge_{i=1}^p$ *primitive-form(*$N_i$*)*

(4) Finally, the logical formula that interprets the whole semantic graph is the conjunction of the logical forms attached to all sources of the graph that are not in restrictions of quantifiers.

[10]The rules for computing $Form_{restrict}$ and $Form_{field}$ presented in the article fail if a source element of the field of a Q-vertex $N$ belongs to the restriction of another Q-vertex $M$ and if $M$ governs $N$ (see donkey sentences).