What Should be a Proper Semantics for Dependency Structures

A. Dikovsky

LINA, UMR CNRS 6241, Nantes University
2, rue de la Houssinière
BP 92208, 44322 Nantes Cedex 3 France

Abstract

We outline a dynamic formal semantics which perfectly fits the Meaning⇔Text theory. This semantics, called Descriptive, interprets Deep dependency structures by semantic expressions defining semantic objects' extensions over dynamic finite structures. This interpretation is based on a strict correspondence between syntactic dependency types and semantic object types and is rule-to-rule compositional. The focus is made on the Syntax-Semantic interface and not on the extension definition.

Keywords

Formal semantics, compositionality, dynamicity, dependency structure

To Igor' Mel'čuk, my beloved Teacher in linguistics and my friend

1 Introduction

Sixty years after the beginning of the era of formal modeling of syntactic dependencies, the question in the title is as relevant as before. Why? Comparing the semantic theories based on dependency and on constituent and similar syntactic structures, one may observe a striking contrast. On the dependency side one may find informal semantic notations (as that of the Semantic Representations of the Meaning⇔Text theory [18, 19]) or straightforward interpretations of dependency structures themselves as semantic expressions (cf. the use of semantic roles and frame semantics [11] in statistical semantic parsing). At the same time, on the constituent side there is an abundance of formal semantics (cf. [1, 17, 26]). To understand this strange phenomenon, we should go into some detail of contemporary semantic theories.

Since the early eighties, two major breakthroughs came about influencing semantic theories: ***Montague grammar*** and ***Dynamic semantics***.

Montagovian semantics. The first breakthrough was brought about by publication of the semantic work of R. Montague [20]. The heart of the Montague grammar is the so called ***rule-to-rule*** architecture, i.e. a correspondence between the rules of a generating grammar and the composition rules of semantic expressions. Due to this correspondence, every derivation of a sentence in the grammar is rule-to-rule translated into the corresponding semantic expression. Moreover, the semantic rules are based on functional types in the sense of Russel: $(u \to v)$, where u is ***argument type*** and v is ***value type*** [1]. When a syntactic rule of syntactic type $T = (U_1 \dots U_n \to V)$

[1] $t = (u_1 \to \dots (u_n \to v) \dots)$ is abbreviated by $t = (u_1 \dots u_n \to v)$. When argument or value

constructs a constituent C of type V from n constituents C_i of types U_i, the corresponding semantic rule of the corresponding semantic type $t = (u_1 \dots u_n \ \rightarrow \ v)$ composes a semantic expression of type v corresponding to V from semantic expressions of types u_i corresponding to U_i. So both constructions may be united in a single construction rule in which all syntactic components are coupled with the corresponding semantic expressions. Such kind of semantics is called ***compositional***. The semantic expressions of Montague grammar are formulas of higher order logic. We explain its basic ideas using the sentence *All semantics leak*.

Example 1. *In this sentence, the semantics of the verb* leak *is a property of entities, i.e. a function from entities (having primitive type e) to truth values (of primitive type t). This property is defined by semantic expression:* $\lceil \mathsf{leak} \rceil^{(e \ \rightarrow \ t)} = \lambda y^e.(LEAK' \ y)$ *of semantic type* $(e \ \rightarrow \ t)$ [2]*, in which $LEAK'$ is a constant of type $(e \ \rightarrow \ t)$.*
Similarly, the semantics of the noun semantics *is the property*
$\lceil \mathsf{semantics} \rceil^{(e \ \rightarrow \ t)} = \lambda z^e.(SEMANTICS' \ z)$ *of type* $(e \ \rightarrow \ t)$,
whereas the determiners all, every *have a more complex semantics:*
$\lceil \mathsf{all} \rceil^{((e \ \rightarrow \ t) \ \rightarrow \ ((e \ \rightarrow \ t) \ \rightarrow \ t))} = \lambda P^{(e \ \rightarrow \ t)} G^{(e \ \rightarrow \ t)}.\forall x((P \ x) \ \rightarrow \ (G \ x))$
(a binary relation on properties), cf. with the semantics of determiner a*:*
$\lceil \mathsf{a} \rceil^{((e \ \rightarrow \ t) \ \rightarrow \ ((e \ \rightarrow \ t) \ \rightarrow \ t))} = \lambda P^{(e \ \rightarrow \ t)} G^{(e \ \rightarrow \ t)}.\exists x((P \ x) \ \wedge \ (G \ x))$.
So the phrase all semantics *is translated into:*
$\lceil all \ semantics \rceil = (\lceil all \rceil \ \lceil semantics \rceil) =$
$(\lambda P^{(e \ \rightarrow \ t)} G^{(e \ \rightarrow \ t)}.\forall x((P \ x) \ \rightarrow \ (G \ x)) \ SEMANTICS') =$
$\lambda G^{(e \ \rightarrow \ t)}.\forall x((SEMANTICS' \ x) \ \rightarrow \ (G \ x))$ *of type* $((e \ \rightarrow \ t) \ \rightarrow \ t)$
and then the phrase (all semantics) leak *is translated into:*
$\lceil (all \ semantics) \ leak \rceil = (\lceil all \ semantics \rceil \ \lceil leak \rceil) =$
$(\lambda G^{(e \ \rightarrow \ t)}.\forall x((SEMANTICS' \ x) \ \rightarrow \ (G \ x)) \ LEAK') =$
$\forall x((SEMANTICS' \ x) \ \rightarrow \ (LEAK' \ x))$ *of type t.*

In fact, the types in Montague grammar are more complex because the logics he uses is ***intensional*** in the sense of Carnap, i.e. all types are relativized with respect to ***worlds***. The relativization serves to express modalities, beliefs and other context dependent meaning aspects.

The fragments of English grammar used in the original Montague works were rudimentary. Nevertheless, his grammar included an important idea of type translation: every phrase p should have a ***syntactic type*** T and the ***semantic type*** t of the corresponding semantic expression $\lceil p \rceil^t$ should be computed from T through a function ρ: $t = \rho(T)$.

Example 2. *For instance, if in our example:*
semantics *has (syntactic) type CN,*
leak *has type* $(NP \ \rightarrow \ S)$,
meet *has type* $(NP \ NP \ \rightarrow \ S) = (NP \ \rightarrow \ (NP \ \rightarrow \ S))$ *(no order on argument types !)*
and all *has type* $(CN \ \rightarrow \ NP)$, *then defining*
$\rho(S) = t$,
$\rho(CN) = (e \ \rightarrow \ t)$,

subtypes are functional themselves, t is called ***higher order***.

[2]Simplifying a bit, $F = \lambda X^u.\Phi^v$ is a function of argument X of type u with values of type v. For an argument expression A of type u, application $(F \ A)$ of F to A gives the value expression resulting from Φ by substituting A for all occurrences of variable X. If Φ is a constant of type $(u \ \rightarrow \ v)$, then the expression $(F \ A) = (\Phi \ A)$ has type v. E.g. $LEAK'^{(e \ \rightarrow \ t)}$ is a constant of type $(e \ \rightarrow \ t)$). Applying it to an expression E^e of type e one obtains the value expresion $(LEAK' \ E)$ of type t.

$\rho(NP) = ((e \ \rightarrow \ t) \ \rightarrow \ t)$ and
$\rho((A \ \rightarrow \ B)) = (\rho(A) \ \rightarrow \ \rho(B))$ for all A and B,
one may compute the semantic types (cf. Example 1):
$\mathsf{semantics}^{\rho(CN)} = \mathsf{semantics}^{(e \ \rightarrow \ t)}$,
$\mathsf{leak}^{\rho((NP \ \rightarrow \ S))} = \mathsf{leak}^{(\rho(NP) \ \rightarrow \ \rho(S))} = \mathsf{leak}^{((e \ \rightarrow \ t) \ \rightarrow \ t) \ \rightarrow \ t)}$, *similarly,*
$\mathsf{meet}^{\rho((NP \ NP \ \rightarrow \ S))} = \mathsf{meet}^{(((e \ \rightarrow \ t) \ \rightarrow \ t) \ \rightarrow \ (((e \ \rightarrow \ t) \ \rightarrow \ t) \ \rightarrow \ t))}$ *and*
$\mathsf{all}^{\rho(CN \ \rightarrow \ NP))} = \mathsf{all}^{(\rho(N) \ \rightarrow \ \rho(NP))} = \mathsf{all}^{((e \ \rightarrow \ t) \ \rightarrow \ ((e \ \rightarrow \ t) \ \rightarrow \ t))}$.

So Montague semantics is compositional in a very strong sense: syntactic formation rules and the corresponding semantic expression composition rules are defined as operations with the same number of arguments, both are typed and the semantic operation type is calculated from the syntactic type through a uniform translation. This semantics is defined for many so called type logical grammars (cf. [22]), higher order intuitionistic logic deductive systems, such as Lambek grammar [15, 16], intuitionistic subsets of linear logic [12], etc., which support this strong compositionality. The principal benefit of using such kind of grammatical calculi is that due to the so called ***Curry-Howard isomorphism***, every semantic expression interpreting a sentence is isomorphic to a proof in the calculus of correctness of syntactic type assignment to words in the sentence.

Dynamic semantics. The second breakthrough in semantic theories was brought about by the dynamic semantics of the Discourse Represetation Theory (DRT) of H. Kamp ([13, 14]). Two main notions of DRT are *reference* and its *scope*. Simplifying matters, one may think of DRT semantic expressions (also called DRT) as of first order formulas in which in the place of quantifiers are used free object variables (references) [3]. The subformula where a reference x is declared is the scope of x. Proper and common nouns, as well as adjectives are expressed as properties of the references: e.g., $John(x)$, $student(x)$, $good(x)$. Co-reference is expressed through reference sharing: $student(x) \ \wedge \ good(x)$ or explicitly: $x = y$. Verbs are represented by predicates, their ***actants*** (the subject and the complements) are represented by the references of the corresponding noun phrases: $\{x, y : John(x) \ \wedge \ speaker(y) \ \wedge \ write(x, y)\}$ corresponds to *John writes to me*. This semantics is ***dynamic*** in the sense that every new sentence in the discourse changes the context adding to it new references. This is due to a particular interpretation of indefinite noun phrases: for every such phrase a new reference is created: e.g., $\{x, y : John(x) \wedge student(y) \wedge good(y) \wedge has(x, y)\}$ corresponds to *John has a good student*. This use of reference y corresponds to the existential quantification in the classical first order logic. Universal quantifiers are, as usual, simulated through implication. For instance, the sentence in Example 1 is expressed by DRT $\{x : semantics(x) \ \rightarrow \ leak(x)\}$.

Why these simple and elegant schemes do not directly apply to dependency structures? Let us analyze the reasons point by point.

1. Different principles of syntactic typing. First of all, historically, the logical type semantics was applied to English and followed the chomskian syntactic tradition. In its type system, the primitive types correspond to the traditional grammatical and syntactic categories, such as S (sentence) and CN (common noun) and composite types define the argument structure of constituent formation rules. Constituent heads are not explicitly represented in these types. By contrast, the dependency syntax tradition is based on syntactic roles of the heads. So in a type $T = (U_1 \ldots U_n \ \rightarrow \ V)$, the value subtype V

[3]In fact, DRT are represented graphically: formulas-scopes are represented by boxes, subformulas by embedded boxes.

should correspond to the syntactic role of the head word (i.e. to the dependency through which it is subordinated to its governor) and the argument subtypes $U_1, \ldots, U_n$ should be the dependencies through which the head word governs its subordinate words. Such types (we will call them ***dependency types*** or ***D-types***) define argument structure of formation rules for constituents with a fixed syntactic role of the head word. As a consequence, the constituent based types are more flexible and the D-types are more numerous (with respect to the same dictionary words). For instance, using the definition of the determiner every in Example 1, one may compute the conventional semantical expression for the sentence *Every girl meets a guy*:

$\lceil Every\ girl\ meets\ a\ guy\rceil = \lceil((meets\ (a\ guy))\ (Every\ girl))\rceil =$
$\quad \forall w((GIRL'\ w)\ \rightarrow\ \exists m((GUY'\ m)\ \wedge\ ((MEET'\ w)\ m)))$

subsequently applying the functional semantic expression $\lceil \mathsf{meet}\rceil$ in Example 2 to the expressions $\lceil a\ guy\rceil$ and $\lceil every\ girl\rceil$. But one also may use another definition of determiner a, different from that in Example 1:

$\lceil \mathsf{a}\rceil^{((e\ \rightarrow\ t)\ \rightarrow\ (e\ \rightarrow\ ((e\ \rightarrow\ t)\ \rightarrow\ t))} =$
$\quad \lambda P^{(e\ \rightarrow\ t)} G^{(e\ \rightarrow\ ((e\ \rightarrow\ t)\ \rightarrow\ t))}.\exists x((P\ x)\ \wedge\ ((\lambda y.(G\ y))\ x))$

to compute a different reading [4] semantic expression:

$\lceil Every\ girl\ meets\ a\ guy\rceil = \lceil((Every\ girl)\ (meets\ (a\ guy)))\rceil =$
$\quad \exists m((GUY'\ m)\ \wedge\ \forall w((GIRL'\ w)\ \rightarrow\ ((MEET'\ w)\ m)))$.

The first analysis is compatible with the natural dependency typing, in which meets is the head of both constituents containing it. In this typing, meets should have type $(SUBJ\ DOBJ\ \rightarrow\ S)$, where $SUBJ$ is the head type of girl and $DOBJ$ is the head type of guy. The second typing corresponds to no natural dependency structure because the type of girl contradicts the syntactic role of *subject*, so the difference between the two is *purely semantical*.

This fundamental difference explains various more specific ones:

- in dependency typing, determiners are subordinated to names, whereas in the logical (constituent) typing, on the contrary, a common noun phrase is the argument of its determiner and functionally depends on it,
- in dependency typing, ad-nominal modifiers depend on the modified nouns and adverbial modifiers depend on modified verbs, whereas in logical typing it is the other way round,
- copulas and light verbs have as simple D-types as regular verbs, whereas their logical types are quite special.

2. Models of types. For all logical type systems and grammars there are logical interpretations (model-truth, algebraic, sometimes intuitionistic). On the contrary, to present day, no workable [5] formal interpretation is found for dependency types [6] (even though there exist D-type calculi).

3. Discontinuity. Formal definition of discontinuous constructions is the most important and complex problem faced by formal grammars. Because of flexibility of logical types, there are various means of logical typing for many discontinuous constructions, such as topicalization, relativization, extraction, pied-piping, islands, etc. (cf. [22, 23]). At the same time, till recently, there were no D-type calculi for discontinuous dependencies. The first calculus was proposed for Categorial Dependency Grammars (CDG)

[4]So called ***de dicto*** meaning in ***quantifying-in*** constructions.

[5]Semantics designed in [7, 9] are unfortunately excessively complex.

[6]This may explain the saying that *Everybody needs dependencies even through nobody understands what is that.*

in [8] and evolved to a very simple polynomial time decidable calculus in [6]. Now it is successfully used for parsing of a rather complete dependency grammar of French [10, 2].
4. Heads. Dependency types, by nature, define dependencies from heads to their subordinates. By contrast, in order to represent the heads in logical constituent types, one needs to use rather expressive logical means (cf. the ***modalities*** in [21, 22]) which make type calculi very complex or even undecidable.

The fundamental difference in treating determiners is the most problematic in this comparative analysis because it concerns reference, scoping and binding. In fact, in order to design a dependency semantics, one should understand how to formally treat determiners as noun subordinates (and, by the way, the place and role of logics in this treatment).

2 Contours of Dependency Semantics

Here a linguist might be perplexed: where is the *real semantics* in all that: lexical meanings (e.g. paraphrases, lexical functions), rules of their combination (e.g. for determining composed paraphrases' arguments), communicative structure (say, semantic diathetic shifts), etc.? Yes, we should make it clear that logical semantics studies **universals**. It is a formal language and systems of schematic rules in it properly interpreting syntactic structures by semantic expressions. "Properly" usualy means two things:
- basing on a type correspondence and strictly compositionaly (cf. rule-to-rule) and
- correctly with respect to reference in models (e.g. correctly interpreting quantifiers).

Language dependent semantical elements and issues may find or not a place in these schemes. For instance, the logicians are not really concerned with lexical functions or with diathetic variants of verbs because they may be treated as typed primitives. Time/aspect relations may be of more concern for them, when included into reference, whereas the linguists treat them as concrete feature values. Because of the reference-through-truth, some issues, e.g. plurality (expression of the plural number) are hard problem for logicians, whereas they are trivial elements of meaning structure for linguists.

Returning to dependency semantics, what should be interpreted and through what? Basically, we make three decisions:
- Instead of interpreting typed surface dependencies (say, using the CDG calculus), we will rather define a rule-to-rule semantics from Deep dependency structures [7]. Doing so, we avoid many specific dependencies (e.g. those connecting the elements of analytic verb forms, of comparative constructions, etc.) and the difficulties with word order.
- We suppose that co-reference and ellipsis are resolved.
- We give up the reference-through-truth leaving it to reasoning.

2.1 Deep Dependency Structures

The ***Deep dependency structures*** (***D-structures*** for short) we use go back to the *stemmas* of Tesnière [25] and are rather close to the ***Deep syntactic structures*** of Mel'čuk [19]. A D-structure is a tree with nodes labeled by ***lexical expressions*** (***L-expressions*** for short) and arcs labeled by ***deep dependency names*** (***D-dependencies*** for short). L-expressions are of the form $E = (Ref\ X : d)\ L^t \in \mathcal{L}_{\bar{f}=\bar{v}}$, where:

[7]We leave aside the question of how surface dependency structures are transformed into deep dependency structures. There may be various techniques, e.g. combinators or finite automata on trees.

- X is a (optional) ***reference***,

- d is a (optional) ***domain determiner***, that is an element of the set $DD = \{$**distrib**(*utive*), **n**(*on*)**distrib**(*utive*), **gener**(*ic*), **mass**, **not_all**, **single**, (*pairwise*)**diff** (*erent*), **exclusive**, $\perp, \ldots\}$, the list is open,

- L^t is a lexeme of D-type t and

- $\mathcal{L}_{\bar{f}=\bar{v}}$ is a ***lexical class*** with features $\bar{f}$ having values $\bar{v}$, e.g.

$$(Ref\ a_1)\ TURN^{(SUBJ\ OBJ[1]\ \rightarrow\ SENT)} \in \mathbf{VT}_{(t/asp=pres/cont,m=ind)}.$$

We suppose that every lexical class has a particular set of features with varying values. E.g., lexemes in a verbal class may be characterized by values of time/aspect, mode, illocutive status, and other semantic features, lexemes in a nominal class may be characterized by gender, number, animacy, mass/nmass, and some other features, for instance, **relation**: relation names for noun attributes (see Fig. 1).

0 is a special ***undefined*** (feature-less) lexeme. It is used in the position of the extracted actant in relative clause D-substructures.

We use the following D-dependencies:
1. ***sentence dependency*** $SENT$;
2. ***actant dependencies***:
2.1. ***subject dependency*** $SUBJ$;
2.2. ***object dependencies*** $OBJ[I]$ $(1 \leq I \leq K)$ [8];
3. ***relative dependency*** REL;
4. ***attribute dependency*** $ATTR$;
5. ***property dependency*** $PROP$;
6. ***degree dependency*** $DEGR$;
7. ***aggregation dependencies*** $AGGR[C]$, (C is an aggregation connector, e.g., **and**, **or**, etc.) for non-distributive nominal units (cf. $\{$*John and Bill's*$\}$ *book*);
8. ***coordination dependencies*** $COOR[C]$ (C is a coordination connector, e.g., **and**, **but**, etc.).

For every reference X in a D-structure D, its ***scope*** consists of D itself and of all nodes below and on the right of X in D. This scope is naturally extended to the D-structures following D in the discourse.

Reference/Zeros Convention. We impose a special discipline of use of references and zeros in D-structures:
1) Every reference X should be in the scope of an L-expression introducing X;
2) Every reference X is introduced only once.
3) A path p in D is a ***trace*** if $p = L \xrightarrow{REL} L_1 \xrightarrow{d_1} \ldots L_r \xrightarrow{d_r} 0$ and neither of dependencies $d_1 \ldots d_r$ is REL.

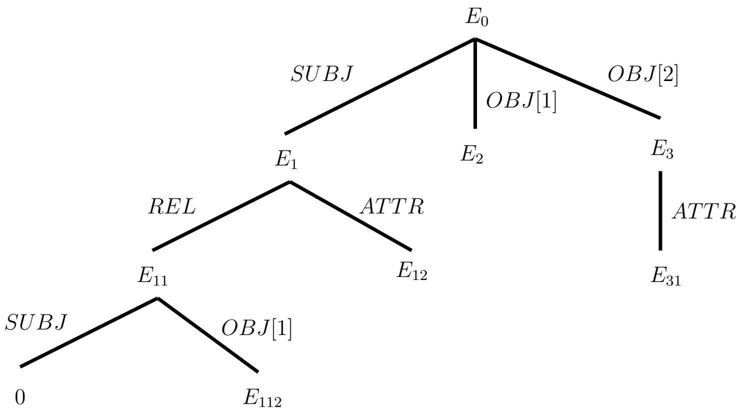
- Every occurrence of 0 in D belongs to a trace;
- Different traces cannot start in the same node;
- 0 lexeme is always a leaf.

In Figure 1 we show a D-structure of the sentence *Every farmer in the village, having tractors, shares them with neighbors.*

2.2 Language of Descriptive Semantics

Full definition of the semantic language we use is beyond the scope of this article. We will rather outline its most specific features and describe its syntax (which is necessary

[8] Usually, $K = 2$.



$E_0 = (Ref\ f_1)\ SHARE \in \mathbf{VT}^{(SUBJ\ OBJ[1]\ OBJ[2]\ \to\ SENT)}_{(t/a=pres/neutr,m=ind)}$ $E_{11} = (Ref\ f_2)\ HAVE \in \mathbf{VT}^{(SUBJ\ OBJ[1]\ \to\ REL)}_{(t/a=pres/neutr,m=ind)}$

$E_1 = (Ref\ p_1 : distrib)\ FARMER \in \mathbf{CN}^{(REL\ ATTR\ \to\ SUBJ)}_{(n=pl)}$ $E_{12} = (Ref\ n_1 : single)\ VILLAGE \in \mathbf{CN}^{(\ \to\ ATTR)}_{(n=sg,relation=loc)}$

$E_2 = (Ref\ s_1 : ndistrib)$ $E_{112} = (Ref\ s_1 : distrib)\ TRACTOR \in \mathbf{CN}^{(\ \to\ OBJ[1])}_{(n=pl)}$

$E_3 = (Ref\ p_2 : ndistrib)\ NEIGHBOUR \in \mathbf{CN}^{(ATTR\ \to\ OBJ[2])}_{(n=pl)}$ $E_{31} = (Ref\ p_1)_{(relation=appurt)}$

Figure 1: D-structure of *Every farmer in the village, having tractors, shares them with neighbors*

for rule-to-rule semantics).

The ideas underlying our semantics, we call ***Descriptive***, originate from different sources.

The first source is the Semantics of Montague, from which we inherit the Russel-Church constructive component of strong typing and of use of λ-calculus, and especially, the rule-to-rule architecture.

The second source is the ***Underspecified Semantics***, a tendency coming into play in mid-90s [4, 5, 24] as a reaction to the inherent difficulties with quantifier scoping (cf. our examples in the Introduction). Its main idea is to postpone the truth/reference analysis by separating non-compositional noun phrase scoping from compositional predication-argument analysis. It is a cutting-the-Gordian-knot technical solution. But it may also be seen as a failure of the program, going back to Frege, Carnap, Tarsky and realized by Montague, to establish linguistic semantics on the grounds of reference through truth in models. This philosophical program contradicts to the long linguistic tradition, in which the concept of truth, negation duality, quantification and reference are peripheral issues. So we make the final step in this direction and completely separate model construction (a language task) from model checking (a reasoning task). Practically, this means that our semantics extracts from D-structures facts of different nature: relations between objects, co-reference constraints and feature value constraints, without checking their consistency.

The third source is the Dynamic semantics from which we take the construction of facts through generation of new references mostly, but not exclusively, provided by the indefinite noun phrases, and also the scoping rules. At the same time, our dynamic semantics is fundamentally different.

- First of all, in Descriptive Semantics, a new fact will be created and added to the evolving structure even when it is provably incompatible with the existing facts (so this structure will not be a model). This is of course not the case in DRT.

- Secondly, the constructed structures are finite. This is a consequence of our choice to discard reference through truth in models, because the negation may be treated as any other marker without presuming the Closed World or some other non monotone assumption guaranteeing logical treatment of negation. Another (technical) consequence is that we no more need reference variables over objects, but may do with constants serving as objects. At the same time, the dynamicity in Descriptive Semantics is more complex than that of DRT because the evolving structure Σ relates to every existing object o its extension $\|o\|^\Sigma$ which is a set of existing objects. When Σ is changed, not only a new object is created, but the extension of some other objects may be updated. This extensional objectivism is really appropriate to natural language semantics: an unreal, impossible, contradictory entity or concept under discussion is treated as any other entity as soon as it is represented by an object (it is of no importance that its extension is empty).

- Finally, and importantly, we no more need the true quantifiers, e.g. the generalized quantifiers [3]. Indeed, when a noun phrase's extension is updated, we are not obliged to really compute the new subset of the cartesian product of verb arguments' extensions (though we can) because we won't check that an implication or conjunction constraint is true in it. So it is enough just to mark the corresponding objects with conditions to check. This is how determiners are treated as noun subordinates in this semantics.

The expressions of Descriptive Semantics are called ***Descriptive Semantics Representations*** (***DSR***). They define typed objects. ***Semantic types*** (***S-types***, or just ***types***) of these objects are constructed from two primitive types: e (entities) and t (statements/facts/events):

S-types:
$PT \ ::= \ e \ \mid \ t$
$T \ ::= \ PT \ \mid \ PT \to T$
Notation : $u_1 \to (\ldots \to (u_k \to u)\ldots) =_{df} u_1 \ldots u_k \to u$.

All considered sets are **finite**. In particular, $\mathcal{O}^\tau$, $\tau \in T$ will denote a **finite** set of objects of type τ, its subsets $\mathcal{C} \subset \mathcal{O}^\tau$ may keep the type, if needed: $\mathcal{C}^\tau$.

DSR are constructed from Object Expressions and Constraint Formulas:

Object Expressions :
$OBJ^\tau \ ::= \ \mathcal{O}^\tau \cup \mathcal{U}$, where $\mathcal{U}$ is a set of local substitution variables (bound by λ).
Constraint Formulas :
$SG \ ::= \ \varepsilon \ \mid \ \neg$ (signs)

$CF^\tau \ ::=$	$\mathcal{F}^\tau(OBJ^\tau) = \mathcal{V}^\tau$	$\mid$	(feature values)
	$SG\ \mathcal{P}^\tau(OBJ^\tau)$	$\mid$	(signed properties)
	$OBJ^\tau \equiv OBJ^\tau$	$\mid$	(co-reference statements)
	$SG\ (OBJ^\tau \in OBJ^\tau)$	$\mid$	(inclusion statements: $o_1 \in \|o_2\|$)
	$SG\ (OBJ^\tau \subseteq OBJ^\tau)$	$\mid$	(subset statements: $\|o_1\| \subseteq \|o_2\|$)
	$SG\ (OBJ^\tau \in \mathcal{C}^\tau)$	$\mid$	(class-inclusion statements)

Elementary Formulas:
$EF^\tau \ ::= \ CF^\tau \ \mid \ SG\ \mathcal{O}^{(u_1 \ldots u_k \to u)}(OBJ^{u_1} : DD, \ldots, OBJ^{u_k} : DD)$
where DD are domain determiners (see above).

Composite DSR are conjunctions of ***abstract*** DSR of the form $\lambda x.E$ and of ***normal*** (not abstract) DSR:

Descriptive Semantics Representations (DSR) :

$$\begin{array}{lcllll} F^{\tau} & ::= & EF^{\tau} & \mid\ \mathcal{O}^{\tau} \in \mathcal{C}^{\tau}\{F\} & \mid\ EF^{\tau} \wedge F^{\tau} \\ F & ::= & \bigcup\limits_{\tau} F^{\tau} & \mid\ \lambda \mathcal{U}.(\bigcup\limits_{\tau} F^{\tau}) & \end{array}$$

3 From D-structures to DSR: rule-to-rule definition

Because of space limits, we won't present the full definition of our semantics. First of all, we will completely neglect the (more technical) aspect of object extension in a structure [9]. So we will only show a part of dynamicity related to object generation. Besides this, we will show interpretation of only some of deep dependencies.

The semantic fragment we show will be defined as a function on pairs $< D, \Xi >$, where D is a D-structure and Ξ is a finite context (i.e. a finite function from D-structure references to objects), which returns a DSR $\lceil D \rceil^{\Xi}$ and a new context. The definition is recursive and bottom-up. It consists of two parts: ***lexical semantics*** Λ^{Ξ} which interprets L-expressions by DSR, and ***composition rules*** defining DSR $\lceil D(E) \rceil^{\Xi}$ of the subtree $D(E)$ with the root E from the DSR of the subordinate L-expressions (for simplicity, we don't distinguish between a node and the L-expression labeling it).

3.1 Lexical Semantics

In the definition below we use the following notation:
Ξ' denotes the next (updated) context ($\Xi'^{(k)}$ is the context after k updates; $\Xi'^{(0)} = \Xi'$).
$\rho : \mathcal{T} \rightarrow T$ is a translation from D-types to S-types.
$Lex : \mathcal{L}^t \rightarrow \mathcal{C}^{\rho(t)}$ is a function assigning object class constants to lexemes.
When o is an object, $\bar{f}(o) = \bar{v}$ denotes a system of its features value definitions: $f_i(o) = v_i$.
For an L-expression $E = (Ref\ X^t)$ **sg** $L^t \in \mathcal{L}_{\bar{f}=\bar{v}}$ in D, we will represent the type t in the most general form as:
$t = (\bar{R}\bar{A}\bar{C} \rightarrow d)$, where:
$\bar{R} = (R_1, \ldots, R_n)$, $n \geq 0$ (actant types),
$\bar{A} = (A_1, \ldots, A_k)$, $k \geq 0$ (non actant non coordination types),
$\bar{C} = (C_1, \ldots, C_m)$, $m \geq 0$ (coordination types).
For E of this type,
$L_1, \ldots, L_k$ denote the L-expressions subordinate to E through dependencies $\bar{A}$,
$\bar{Q} = (q_1, \ldots, q_k)$ denote their domain determiners,
sg is the sign of L and
$relation(L^t)$ denotes the value of the feature $relation$ in the case where E is an attribute, i.e. is subordinate through dependency $ATTR$ (cf. the attribute $VILLAGE$ in Figure 1).

Here are some lexical definitions.

3.1.1 References and zeros

Let X^t be a reference.
Case 1. $E = (Ref\ X^t)$ is a reference in Ξ (i.e. $(X^t = o) \in \Xi$ for some o). Then:
$\Lambda^{\Xi}(E) = true$ and $\Xi' = \Xi$.
Case 2. $E = (Ref\ X^t)$ and X^t is not in Ξ. Then:
$\Lambda^{\Xi}(E) = (\textbf{new}\ o \in \mathcal{C}^{\rho(t)})\{true\}$ and $\Xi' = \Xi \cup \{X^t = o\}$.

[9]In [7, 9], on the contrary, there is no syntax interface, but only a very technical extension definition.

Here and below **new** $o \in \mathcal{C}^{\rho(t)}$ stands for the call of a procedure **new** creating a new object o in the class $\mathcal{C}^{\rho(t)}$.

Zeros. $E = 0$. Then $\Lambda^{\Xi}(E) = \lambda x.x$ for a new local variable $x \in \mathcal{U}$ and $\Xi' = \Xi$.

In all cases shown below $\Xi' = \Xi \cup \{X^t = o\}$.

3.1.2 Verbal Lexemes

$E = (\mathit{Ref}\ X^t)\ \mathbf{sg}\ L^t \in \mathcal{L}^t_{\bar{f}=\bar{v}}$), $t = (\bar{R}\bar{A}\bar{C} \ \rightarrow\ d)$, where $d ::= \mathit{SENT} \mid \mathit{PROP} \mid \mathit{COOR}$, and $\rho(t) = (\alpha \ \rightarrow\ s)$ for some α.

Case of non-copula verbs $\mathcal{L}^t_{\bar{f}=\bar{v}} =_{df} \mathbf{V}^t_{\bar{f}=\bar{v}}$. Then X^t is not in Ξ and:

$$\Lambda^{\Xi}(E) = \lambda \bar{x}\bar{y}\bar{z}.((\mathbf{new}\ o) \in \mathit{Lex}(L^t))\{\mathbf{sg}\ o(x_1 : q_1, \ldots, x_n : q_n) \ \wedge\ \bar{f}(o) = \bar{v} \ \wedge \bigwedge_{A_i = \mathit{ATTR}} \mathit{relation}(L_i)(o) = y_i \ \wedge \bigwedge_{A_i = \mathit{PROP} \mid \mathit{REL}} (y_i\ o) \ \wedge\ \bigwedge_{j=1}^{m} z_j\},$$

where $\bar{x} = x_1, \ldots, x_n$, $\bar{y} = y_1, \ldots, y_k$, $\bar{z} = z_1, \ldots, z_m$ are local variables in $\mathcal{U}$ corresponding to the argument types $\bar{R}, \bar{A}, \bar{C}$.

Less formally, this DSR creates and adds to the constraints the new fact $\mathbf{sg}\ o(x_1 : q_1, \ldots, x_n : q_n)$ identified by a new object o, the feature constraints applied to o, the values of its attributes, its properties and (recursively) the coordinated facts.

3.1.3 Nominal Lexemes

Simplifying the real situation, we suppose that nominal lexemes have no actants.

Common nouns $\mathcal{L}^t_{\bar{f}=\bar{v}} =_{df} \mathbf{CN}^t_{\bar{f}=\bar{v}}$.

$E = (\mathit{Ref}\ X^t)\ L^t \in \mathbf{CN}^t_{\bar{f}=\bar{v}}$, $t = (\bar{A} \ \rightarrow\ d)$, $\bar{A} = (A_1, \ldots, A_m)$ and $\rho(t) = (\alpha \ \rightarrow\ e)$.

Case of non aggregative dependency d.

$$\Lambda^{\Xi}(E) = \lambda \bar{y}.((\mathbf{new}\ o) \in \mathit{Lex}(L^t))\{\bar{f}(o) = \bar{v} \ \wedge \bigwedge_{A_i = \mathit{ATTR}} \mathit{relation}(L_i)(o) = y_i \ \wedge \bigwedge_{A_i = \mathit{PROP} \mid \mathit{REL}} (y_i\ o)\}.$$

Note that $(y_i\ o)$ may correspond to a relative clause. In this case o will be passed into its DSR through λ-abstracted local variables using the Abstraction propagation rule below.

3.1.4 Adjectival and Adverbial Lexemes

Case of property lexemes $\mathcal{L}^t_{\bar{f}=\bar{v}} =_{df} \mathbf{A}^t_{\bar{f}=\bar{v}}$, where

$t = (\bar{A} \ \rightarrow\ \mathit{PROP})$, $\bar{A} = (A_1, \ldots, A_m)$ and $\rho(t) = (\alpha \ \rightarrow\ s)$.

$E = (\mathit{Ref}\ X^t)\ L^t \in \mathbf{A}^t_{\bar{f}=\bar{v}}$,

$$\Lambda^{\Xi}(E) = \lambda x\bar{y}.((\mathbf{new}\ o) \in \mathit{Lex}(L^t))\{\mathbf{sg}\ o(x) \ \wedge\ \bar{f}(o) = \bar{v} \ \wedge \bigwedge_{A_i = \mathit{ATTR}} \mathit{relation}(L_i)(o) = y_i \ \wedge \bigwedge_{A_i = \mathit{PROP} \mid \mathit{REL}} (y_i\ o)\}.$$

3.2 Composition Rules

Suppose that $E = (\mathit{Ref}\ X^t)\ L^t \in \mathcal{L}_{\bar{f}=\bar{v}}$ and $t = (\bar{B} \ \rightarrow\ d)$, where $\bar{B} = (B_1, \ldots, B_r)$ and $E_1, \ldots, E_r$ are the L-expressions subordinate to E through $B_1, \ldots, B_r$ in D. Then:

$$\lceil D(E) \rceil^{\Xi} = \Lambda^{\Xi'^{(r)}}(E) \ \circ\ (\lceil D(E_1) \rceil^{\Xi'^{(0)}} \ldots \lceil D(E_r) \rceil^{\Xi'^{(r-1)}}),$$

where $\circ$ is the composition defined by the following rules:

Normal composition rule. If DSR $\Delta_1, \ldots, \Delta_r$ are all normal or d is REL or $PROP$, then:

$$\Delta \circ (\Delta_1, \ldots, \Delta_r) =_{df} (\Delta\ \Delta_1 \ldots \Delta_r).$$

Abstraction propagation rule. If all DSR $\Delta_1, \ldots, \Delta_r$, but one (say, Δ_1) are normal, then:

$$\Delta \circ (\Delta_1, \ldots, \Delta_r) =_{df} \lambda x.(\Delta\ (\Delta_1\ x) \ldots \Delta_r),$$

where $x \in \mathcal{U}$ is a new local variable.

Λ is defined so that in every D-structure, in any non leaf L-expression E either the first or the second rule is applied. This is a consequence of the following fact.

Proposition 1. *Let D be a D-structure, E be an L-expression in D and $E_1, \ldots, E_r$ be all its subordinate L-expressions. Then among the DSR $\lceil D(E_1)\rceil^{\Xi'(0)}, \ldots, \lceil D(E_r)\rceil^{\Xi'(r-1)}$ there is at most one abstract.*

4 Conclusion

The outlined fragment of Descriptive semantics shows how one can define a compositional dynamic formal semantics for dependency structures. Basically, it should be defined from the Deep, not from the Surface, dependency structures and should leave aside reference through truth in models. Under these conditions, the semantics may be designed in a strictly compositional rule-to-rule style and may be established on a natural correspondence between dependency and semantic object types. Moreover, this semantics is object-oriented, deals with finite structures and is easily and efficiently implementable.

References

[1] *Logic, Language, and Meaning. Vol I: Introduction to Logic; Vol II: Intensional Logic and Logical Grammar.* University of Chicago Press, Chicago and London, 1991.

[2] Ramadan Alfared, Denis Béchet, and Alexander Dikovsky. Cdg lab: a toolbox for dependency grammars and dependency treebanks development. In K. Gerdes, E. Hajicova, and L. Wanner, editors, *Proc. of the 1st Intern. Conf. on Dependency Linguistics (Depling 2011)*, Barcelona, Spain, 2011. http://depling.org/proceedingsDepling2011/.

[3] J. Barwise and R. Cooper. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4:159–219, 1981.

[4] Johan Bos. Predicate logic unplugged. In P. Dekker and M. Stokhof, editors, *Proc. of the 10th Amsterdam Colloquium*, pages 133–142, 1995.

[5] Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag. Translation using minimal recursion semantics. In *Proc. of the 6th Int. Conf. on Theoretical and Methodological Issues in Machine Translation (TMI95)*, Leuven, Belgium, 1995.

[6] Michael Dekhtyar and Alexander Dikovsky. Generalized categorial dependency grammars. In *Trakhtenbrot/Festschrift*, LNCS 4800, pages 230–255. Springer, 2008.

[7] A. Dikovsky. On speaker's stance meaning of discourse. In *Proc. of the 4th International Conference "Meaning-Text Theory" (MTT 2009)*, Montreal, Canada, 2009.

[8] Alexander Dikovsky. Dependencies as categories. In *"Recent Advances in Dependency Grammars". COLING'04 Workshop*, pages 90–97, 2004.

[9] Alexander Dikovsky. On constructive semantics of natural language. In Solomon Feferman and Wilfried Sieg, editors, *Proofs, Categories and Computations: Essays in honor of Grigori Mints*. College Publications, London, UK, 2010. With the collaboration of Vladik Kreinovich, Vladimir Lifschitz, and Ruy de Queiroz.

[10] Alexander Dikovsky. Categorial dependency grammars: from theory to large scale grammars. In K. Gerdes, E. Hajicova, and L. Wanner, editors, *Proc. of the 1st Intern. Conf. on Dependency Linguistics (Depling 2011)*, Barcelona, Spain, 2011. http://depling.org/proceedingsDepling2011/.

[11] C.J. Fillmore, C.R. Johnson, and R.L. Petruck. Background to framenet. *International Journal of Lexicography*, 16:235–250, 2003.

[12] J.-Y. Girard. Linear logic. *Theoretical Computer Scienc*, 50:1–102, 1987.

[13] H. Kamp. A theory of truth and semantic representation. In J. Groenendijk, T. Janssen, and M. Stokhoff, editors, *Formal Methods in the Study of Language*. Foris, Dordrecht, 1981.

[14] Hans Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer Academic Publishers, Dordrecht, 1994.

[15] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, 65:154–169, 1958.

[16] Joachim Lambek. On the calculus of syntactic types. In R. Jacobson, editor, *Structure of Languages and its Mathematical Aspects. Proc. of the Symposia in Applied Mathematics. Vol. XII, American Mathematical Society*, pages 166–178, 1961.

[17] Shalom Lappin, editor. *The Handbook of Contemporary Semantic Theory*. Blackwell Publishing, 1996.

[18] I. Mel'čuk. Meaning-Text Models: A Recent Trend in Soviet Linguistics. *Annual Review of Antropology*, 10:27–62, 1981.

[19] Igor Mel'čuk. Vers une linguistique ≪sens–texte≫ . Leçon inagurale au collège de france. 1997.

[20] R. Montague. In R. Thomason, editor, *Formal Philosophy: Selected Papers of Richard Motague*. Yale Univ. Press, New Haven, CT, 1974.

[21] Michael Moortgat and Glyn V. Morrill. Heads and phrases. Type calculus for dependency and constituent structure. Ms OTS, Utrecht.

[22] Glyn V. Morrill. *Type Logical Grammar. Categorial Logic of Signs*. Kluwer Academic Publishers, Dordrecht, 1994.

[23] Glyn V. Morrill. Logic programming of the displacement calculus. In S. Pogodalla and J.-P. Prost, editors, *Logical Aspects of Computational Linguistics. Proc. of the 6th Intern. Conference LACL 2011*, LNAI 6736, pages 175–189. Springer Verlag, 2011.

[24] Reinhard Muskens. Order-independence and underspecification. In J. Groenendijk, editor, *Ellipsis, Underspecification, Events and More in Dynamic Semantics*. 1995. DYANA Deliverable R.2.2.C.

[25] L. Tesnière. *Éléments de syntaxe structurale*. Librairie C. Klincksiek, Paris, 1959.

[26] J. van Benthem and A. ter Meulen, editors. *Handbook of Logic and Language*. North-Holland Elsevier, The MIT Press, Amsterdam, Cambridge, 1997.