СИНТАКСИЧЕСКИЙ И СЕМАНТИЧЕСКИЙ ПАРСЕР, ОСНОВАННЫЙ НА ЛИНГВИСТИЧЕСКИХ ТЕХНОЛОГИЯХ ABBYY COMPRENO

Анисимович К. В. (Konstantin_An@abbyy.com),
Дружкин К. Ю. (Konstantin_D@abbyy.com),
Зуев К. А. (Konstantin_Z@abbyy.com),
Минлос Ф. Р. (f.minlos@gmail.com),
Петрова М. А. (Maria_P@abbyy.com),
Селегей В. П. (Vladimir_S@abbyy.com)

Компания Abbyy, Москва, Россия

Ключевые слова: синтаксис, семантика, автоматический анализ, парсер

SYNTACTIC AND SEMANTIC PARSER BASED ON ABBYY COMPRENO LINGUISTIC TECHNOLOGIES

Anisimovich K. V. (Konstantin_An@abbyy.com),
Druzhkin K. Ju. (Konstantin_D@abbyy.com),
Minlos F. R. (f.minlos@gmail.com),
Petrova M. A. (Maria_P@abbyy.com),
Selegey V. P. (Vladimir_S@abbyy.com),
Zuev K. A. (Konstantin_Z@abbyy.com)

ABBYY, Moscow, Russia

The paper presents ABBYY Syntactic and Semantic Parser that was a participant of the Dialog 2012 Syntactic Parsers Testing Forum. We will refer to the parser technology (both parsing algorithms and linguistic model) as Compreno technology. We do not touch on any evaluation issues, as they are tackled by the Forum panel. Instead, the paper makes public some underlying principles of the parser. What we want to communicate directly concerning the testing are the features of the project which are both relevant to the comparison of our results with the "gold standard" adopted by the panel and, at the same time, important for the whole architecture of our technology.

Key words: syntax, semantics, natural language processing, parser

Introduction

Essentially, what a parser strives to extract from the sentence is *who did what to whom* (*when* and *where*). The question is what level of representation is aimed at. In Compreno project, the ultimate goal is to achieve not only the syntactic disambiguation, but the semantic one as well. Semantic and syntactic representations are viewed rather as two facets of the same structure (much as in the mainstream G&B / P&P approach), than as two distinct types of structure (as, for instance, in LFG). Another (interrelated) feature of the Compreno parsing technology is that syntactic and semantic disambiguation are processed in parallel from the very start (in contrast to the architecture more usual for the NLP systems — the semantic analysis follows the syntactic one). This Compreno peculiarity makes it difficult to adapt analysis structures for the surface syntax testing requirements.

The objective of immediate semantic interpretation for a syntactic structure determine some features of the syntactic structures in question. The most evident one is the heavy use of null elements. The model case of null element is phonetically null subject of non-finite clause. For example, in equi-predicates and raising predicates, the null subject of infinitive is coreferential with an argument of the matrix predicate.

Another huge class of mismatches also comes from different perspectives, taken by the Compreno project and the Golden Standard (GS). Within the purely syntactic framework of the GS, functional words (such as prepositions, conjunctions, and complementizers) are treated as heads, while in Compreno syntax, they are dependent nodes. The same goes for a little more complicated **более чем**-construction (*в течение более чем сорока лет*; *подавать документы в более чем пять вузов*). The GS picks up *более* as the head of the construction (a natural decision, within the syntacticocentric approach), for Compreno, the semantic link to the noun is more important (for example, we immediately obtain the collocation *подавать документы в вуз(ы)* ‘*submit documents to institutes*’, without any intervening nodes).

A final example of discrepancy to present here is clause attachment in examples like *Не вызывает сомнения, что…* ‘No doubt…’. Compreno parser attaches the *что*-clause to the noun licensing it; thus we can count statistics concerning the frequency of *что*-clauses with specific nouns.

The task of the normalization of Compreno syntactic structures for the parser competition was all but trivial, taking into account striking difference between the two syntactic representation (when two or more discrepancies occurred in the same clause, the situation could became even more complicated).

The Compreno Linguistic Model

1. Basic syntax

1.1. Dependency links and constituent structure

The syntactic structure of a sentence is represented as a syntactic tree, augmented with non-tree links. Tree links encode syntactic dominance; non-tree links capture conjunction, anaphora, distant agreement, and other non-local dependencies between nodes.

The syntactic trees modeled within the Compreno framework may be seen either as projective dependency trees or, alternatively, as constituent trees, where every non-terminal node has one terminal child (its **lexical head**, or **core**) and zero or more non-terminal children.

Projective dependency trees are such that their subtrees correspond to contiguous chunks of text (which means that if A and C are descendants of X, and B is linearly between A and C, then B is also a descendant of X). This property allows us to speak of subtrees and constituents interchangeably[1].

Children fill certain syntactic slots within the parent (e. g. articles fill the “$Article” slot within noun phrases; the link between a verb and its subject is labeled "$Subject” and so on). Most syntactic links also get a semantic interpretation, with the exception of function words (such as auxiliary verbs, conjunctions, or prepositions), and displaced elements (see below).

We treat conjuncts as sisters, i. e. daughters of a common parent.

1.2. Linear order

As in HPSG, the description of possible linear orders is kept separate from the description of possible constituent structures. Suppose, `{Slot _ 1 ... Slot _ N}` is a set of syntactic slots, available inside a noun phrase. E.g. $Article is in this set,

1 Projective dependency trees are obviously isomorphic to constituent structures with lexical heads: “words” correspond to “lexical cores”, “subtrees” to “constituents”, “dependency links” to the relations between the (cores of) constituents and the (cores of) their child constituents. Note that, as function words and auxiliary verbs are treated as dependent elements, many syntactic constructions become projective, which would otherwise be non-projective. Consider the case of approximative inversion (i) and object fronting (ii):
(i) [года [через] [два]]
(ii) [Обед] [будет] готовить [Маша].
(An alternative analysis would be: через → года → два; будет → готовить → обед.)

and also $Modifier_Attributive, $OfPostmodifier and many others. A **linear order template** would look like this:

(9) `Slot _ 1 [Slot _ 2 Slot _ 3] Core Slot _ 4 (Slot _ 5 | Slot _ 6) Slot _ 2`

Not all slots must be filled, but if they are filled, then the fillers must be in this order with respect to the core and to each other. Square brackets mean that any order is allowed. Vertical bar means that only one of the alternatives is allowed.

Constituents have types that roughly correspond to parts of speech of their lexical cores (e.g. adjective phrases with adjectival cores). For every constituent type there is a linear order description. A linear order description is a set of variants. A variant is a pair of:

- a grammar expression that is matched against the grammar value of a constituent[2],
- a linear order template.

Variants are needed because linear orders may vary (e. g. the position of English subjects is different in declarative and interrogative sentences).

A number of inferences can be made from such descriptions. For example, we may know that some slot is never allowed before the core in noun phrases. Such inferences are automatically extracted, stored away and later used by the parser.

1.3. Morphological and syntactic categories

A morphological paradigm is a multi-dimensional table, where dimensions are the morphological categories (= attributes), and columns or rows are morphological grammemes (= values). Consider the following fragment of a traditional morphological paradigm (categories: Case × Number):

	Singular	**Plural**
Nominative	*слон*	*слоны*
Genitive	*слона*	*слонов*
…	…	…

Ideally, all cells in the table must be filled with single word-forms. But how should we deal with analytical verb forms (e. g. futures or passives that require auxiliary verbs)? If we put them in the table, we leave the realm of morphology proper. Auxiliary verbs fit into the internal structure of verb phrases (in syntax), not into the internal structure of verbs (in morphology). But if we leave them out, the paradigm

2 See section 1.6.

table will have gaps, and may lose its symmetry. Here is a typical example (categories: Tense × Aspect; Passive, 3Sg, Plural):

	Past	**Present**	**Future**
Imperfective	*создавались*	*создаются*	*будут создаваться*
Perfective	*были созданы*	*созданы*	*будут созданы*

The inclusion of analytical forms into verbal paradigms was initially motivated by symmetry. But this approach can be generalized to other grammatical elements as well. If auxiliary verbs are part of verbal paradigm, then prepositions are part of nominal paradigm. So in addition to the morphological category of Case with grammemes <Nominative | Genitive | Dative | …>, we now have the syntactic category of ExtendedCase with grammemes <ECabout | ECafter | … | ECwithout>. Here is a syntactic paradigm with grammatical elements (Definiteness × ExtendedCase; Plural):

	No preposition	**ECin**	**ECof**	**…**
Indefinite	*books*	*in books*	*of books*	…
Definite	*the books*	*in the books*	*of the books*	…

Finally, for every syntactic slot $Slot we can create a binary category SlotCategory with grammemes <SlotFilled | SlotNotFilled>. For instance, noun phrases that contain an "of"-postmodifier ("[the father [of John]]") get the <OfPostModifier> grammeme. Here is a fragment of a syntactic paradigm with lexical child phrases (presence of adjective modifiers × presence of of-modifiers in NPs; Singular, Definite):

	Without of-modifiers	**With of-modifiers**
Without adjective modifiers	*the father* *the statement*	*the father of John* *the statement of truth*
With adjective modifieers	*the late father* *the bold statement*	*the late father of John* *the bold statement of truth*

The move from purely morphological inflectional paradigms to morphosyntactic paradigms including analytical forms (of, e.g., tense) is a traditional one. What is unusual in the Compreno approach is the inclusion of the absolutely asymmetrical oppositions in the paradigm. E.g., the absence of a modifier is not likely to be perceived as a zero instantiation of some binary opposition. But in Compreno project, the grammeme is a universal means to refer to any syntactic configuration (see also the next chapter).

In this system, while establishing a syntactic link between a parent node P and a child node C, we do not have to look at other nodes around, but the syntactic context is visible in the grammar values of P and C. The presence and type of preposition is visible on the noun, the presence of negation is visible on the main verb and so on. Even the demand that C should have no children can be expressed as a restriction on the grammar value of C.

1.4. Syntactic levels and syntactic forms

At the core of the Compreno framework lies the notion of **syntactic paradigm**. The notion may at first sound odd, but it is a natural extension of the traditional notion of paradigms in morphology. Moreover, this move is not a new one. The term *syntactic paradigm* was used by Kenneth L. Pike [4][3]. The theoretical perspective of notions morphological paradigm vs. syntactic paradigm is discussed in detail in [6].

The set of categories and available grammemes for a certain part of speech is composed of multiple sources. First, there are **morphological categories** that come from the morphological dictionary (e. g. case or gender of adjectives). Second, there are **classifying categories for lexemes**; their grammemes are assigned manually to some lexemes that have non-trivial syntactic properties (e. g. Russian numerals from 2 to 4 are marked <SmallNumeral>). They can be viewed as extensions to morphology. Third, there are **classifying categories for classes (see section 3).**. Fourth, there are **syntactic categories**, to be discussed here. Finally, there are special categories, such as Capitalization; their values are supplied by the processing engine.

A pair of a lexeme and a lexical class uniquely selects **a syntactic paradigm**, which defines a full range of syntactic possibilities allowed.

A paradigm is a set of **syntactic levels**, each of which defines some aspect of syntactic structure. Logically, a paradigm is a <u>conjunction</u> of levels: the constituent has to match all of them. Some levels are universal (i. e. defined for parts of speech, e. g. for all noun phrases), some lexicalized (i. e. defined for branches of semantic hierarchy).

A syntactic level is a set of **syntactic forms**, each of which defines a specific syntactic configuration. Logically, a level is a <u>disjunction</u> of forms: the constituent has to match at least one of them.

Levels are associated with categories of syntax; forms are associated with specific grammemes. If a constituent matches a form, it gets a corresponding grammeme.

A syntactic form may specify, among other things:

- a grammar expression that is matched against the grammar value of a constituent,
- zero or more surface slots that must be filled,
- for each surface slot, a set of semantic slots, available as its semantic interpretation.

There are syntactic categories that are parallel to morphological categories. For example, verbs have morphological Tense. Bun in analytical forms, the Tense is morphologically expressed on the auxiliary verb. So we have a syntactic category

[3] The founder of tagmemic grammar; he also coined the term grammeme, heavily used in Russian linguistic, including the Compreno project, but almost uknown in the Western tradition, see [12].

SyntacticTense. When the *main* verb is finite, its SyntacticTense follows its Tense; but when the *auxiliary* verb is finite, the main verb copies its value of SyntacticTense by agreement.

Also, consider the syntax of Russian numerical expressions in Nominative and Inanimate Accusative. With “small numerals” (2,3,4), head nouns assume the form of <Genitive, Singular>; with “big numerals”, <Genitive, Plural>. To deal with this, we distinguish the morphological categories of Case and Number and the syntactic categories of SyntCase and SyntNumber. The former correspond to the morphological form of the head noun; the latter, to the semantic and combinatorial properties of the whole constituent. In the following examples, the morphological grammemes are glossed next to word-forms, and the syntactic grammemes, after the square brackets.

(1) [[Два] мальчика GENITIVE SINGULAR] SYNTNOMINATIVE SYNTPLURAL

(2) [[Пять] мальчиков GENITIVE PLURAL] SYNTNOMINATIVE SYNTPLURAL

1.5. Underspecification

A word-form (or a constituent) can be *underspecified* with respect to some category. For example, plural forms of Russian adjectives are underspecified for Gender.

In many cases, morphological ambiguity is resolved by syntactic context; but sometimes it must remain. For example, some Russian nouns have identical forms for Genitive and Accusative, and in some constructions both Genitive and Accusative are allowed.

(3) Девочка не заметила мальчика GENITIVE | ACCUSATIVE

If a morphological ambiguity cannot be resolved, we say that the grammar value is underspecified in some category.

Underspecification has computational benefits. As an extreme example, consider indeclinable nouns (e.g. пальто SINGULAR | PLURAL . NOMINATIVE | GENITIVE | DATIVE | ACCUSATIVE | INSTRUMENTAL). Frameworks that demand all word-forms to be fully specified, would split them into many homonymous word-forms. But at the early stages of analysis it can be better to have a single ambiguous node than 10 fully specified nodes.

Underspecification is intimately connected to three-valued logic. (Indeed, the question “Is this word-form Genitive?” now has three possible answers: “Yes”, “No” and “Maybe”). Also, it naturally lends itself to use in unification algorithms (more of which later).

1.6. Grammar expressions

The name of a grammeme can be viewed as a predicate, and such predicates can be combined by standard means of predicate logic: conjunction, disjunction, negation and bracketing. So we get **grammar expressions**, as the following[4]:

(4) `~Present | Imperfective`

(5) `~(Present, ~Imperfective)`

Grammemes in a grammatical category form a set; so negation means complementation in that set. Suppose a category `A` with grammemes `{a1, a2, a3}`. The expression <`~a1`> is equivalent to <`a2 | a3`>.

Grammar expressions are used in many places. Most importantly, they restrict the set of possible parent-child combinations in syntactic trees.

1) Syntactic slots (or, alternatively, links) are special objects with their properties. The major property of a slot is "government", which is a grammar expression describing the allowed form for the child node.

2) Syntactic slots are connected to "syntactic forms" of the parent constituent. The concept of "syntactic forms" will be explained later. What is important now is that syntactic forms have grammar expressions describing the parent node. For example, comparisons (*"than X"*) can be attached to adjectives, only when adjectives have comparative degree (*"bigger than X"*, but not *"big | biggest than X"*).

When a grammar value of a would-be child is matched against a grammar expression of the slot, the expression can return "false" or "true". "True" means "possibly true", because the grammar value can be underspecified.

Suppose we check if an indeclinable noun can be a dative object. The government of the slot demands <`Dative`>, but the noun has <`Nominative|Genitive|Dative|…`>. The check returns "possibly true", but the contradictory grammemes remain in the grammar value of the noun. But when, at some stage of parsing, we commit to having this link in the tree, the contradictory grammemes <`Nominative|Genitive|…`> are filtered from the grammar value, and only <`Dative`> remains.

[4] Those are two ways to express logical implication.

1.7. Agreement

1.8. In traditional linguistics, government and agreement are the two most prominent types of featural dependency between words. The previous section showed how grammar expressions capture the traditional notion of government; this section in turn discusses agreement rules.

An agreement rule is a set of variants. An agreement variant is a triple of:

1. a grammar expression, that is matched against the grammar value of the first node;
2. a grammar expression, that is matched against the grammar value of the second node;
3. a list of agreement categories, in which the two nodes must have the same grammar value.

Logically, an agreement rule is a disjunction of its variants, and a variant is a conjuction of its three parts.

Agreement is checked between two nodes, connected in some way. The rules of agreement are auxiliary to the rules that create connections. Depending on the type of the main rule, the "first and second nodes in agreement" can be parent and child, or left conjunct and right conjunct, or noun phrase and anaphoric pronoun that refers to it.

Note that agreement rules are not restricted to agreement proper (when two nodes have identical values in some categories), but can capture other regularities as well. Suppose we need to say: if the child has <P>, the parent must have <Q>. We can create an agreement rule with two variants: both variants have no agreement categories; the first variant demands <P> of the child and <Q> of the parent; the second demands <~P> of the child and nothing of the parent.

Formulating the rules of noun-numeral agreement in Russian is left as an exercise to the reader.

2. Null elements

2.1. Motivation for null elements

The use of null elements in linguistic frameworks is contentious. On the one hand, null elements simplify syntactic rules, and make surface structures closer to their semantic interpretations. On the other hand, null elements remain a theoretical construct, not directly supported by the observable data, and, more importantly, they come with a high computational cost.

From the algorithmic point of view, null elements fall into two groups. Null elements that have no "real" descendants are not important in the early stages of parsing. The decision about their existence can be postponed until a syntactic tree has been built. Such unproblematic cases will be discussed in the next section. Null elements that can have "real" descendants make more trouble, leading to an undesirable proliferation of hypotheses at early stages of parsing. So it is worthwhile to consider the reasons for having null cores.

The substantivation of adjectives, as in (6), and coordinate ellipsis, as in (7), can be easily analyzed both ways:

(6) *The politicians and* ***the rich*** *did not bother.*

(7) *It improves the flow of traffic not in one direction but* ***in two****.*

From the purely syntactic point of view, it is also possible to allow adjectives and numerals in nominal syntactic positions, and to allow their combination with articles and prepositions. But postulating zero nouns (*the rich <u>people</u>, in two <u>directions</u>*) not only allows us to keep syntax simple, it also makes possible to attach lexical meaning (e. g., 'direction') to a separate syntactic node, making other computations more straightforward.

Example (8) is more difficult for us, because we take prepositions to be children of nouns:

(8) ***Privacy of*** *and access to information.*

Here again ellipsis greatly simplifies the structure (*[privacy [[of] ~~information~~]]*); but with some ingenuity we could do without it.

2.2. Ellipsis templates

An ellipsis template describes a fragment of syntactic structure. It specifies both linear and hierarchical relations between constituents. It can mention coordination links. It can check grammar value of nodes against grammar expressions. In short, a template can refer to most aspects of the final structure.

One or more nodes in the template bear the label "new". Those are the null elements. If we can insert them in the sentence, so that the resulting structure fits the template, then null elements are created. (Their existence remains hypothetical; they can make their way into the final structure, or they may not.)

We will not describe the syntax of ellipsis templates here, but the basic idea of a linear-and-hierarchical template is certainly familiar to the reader.[5]

A word of caution is in order. Ellipsis is good for people, as it keeps the grammar simple and intuitive; but the real computational gain comes from templates, not from the use of ellipsis per se. While most other grammar rules are binary and local,

[5] Querying and transforming XML, for instance, requires similar techniques, since the structure of an XML document is both linear and hierarchical, with some non-local references between nodes.

templates are holistic, and offer a rich view of the context. If the context is wrong, the template can be rejected very fast. An immediate-constituent grammar designed to handle ellipsis would create many false hypotheses which would then die slowly and painfully, littering the syntactic graph for a while.

2.3. Movement rules

Many models of dependency syntax (most notably, the Meaning-Text Model by Mel'chuk) allow non-projective links to capture long-distance dependencies. Consider, for example, the following sentences:

(9) *Куда ты* ***хочешь****, чтобы я пошёл?*

(10) *Одну сумку ему* ***разрешили*** *оставить.*

Our model handles such cases thus:

- The displaced element is attached to the linearly suitable "adoptive parent", which must be an ancestor of its "true" parent,
- It is attached in a special slot that has no semantic interpretation.
- This special slot triggers a rule, which searches the subtree of its "adoptive" parent, looking for a place to put a "proform", or "movement trace".
- A non-tree link is created between the displaced element and its proform.

Walking the syntactic tree from the displaced element to its proform, we go one step up (to the "adoptive parent") and several steps down (to its "true parent", under which the proform is attached). The movement rule specifies the descending part of this path in a **path template**. As in the previous section, we will not discuss the exact syntax of path templates. In a way, they resemble regular expressions.

At one phase of the translation process, a structure is transformed so that all movements are discarded, and all displaced elements return to their "true" parents.

2.4. Syntactic control

A similar mechanism to movement is used to capture control constructions, discussed extensively by the generative grammarians. Following the generative tradition, we create null subjects for non-finite verb forms. As these proforms have no children of their own, they are not important at the early stages of analysis, but are created when the syntactic tree has been built. In some contexts they have a generic meaning; while in others they are controlled.

(11) *Alice* ***promised*** *Bob to come.*

(12) *Alice* ***persuaded*** *Bob to come.*

(13) *Bob was* ***persuaded*** *to come.*

In this construction, the null subjects of infinitives are controlled either by subject, as in (11) and (13), or by direct object, as in (12). Technically, the infinitive is attached in a slot that triggers the rule of control. This rule has several variants, corresponding to the configurations in (11)–(13). Note that the variants have to check both syntactic grammemes (active vs. passive voice), and classifying grammemes (what type of control is associated with specific verbs).

Contrary to movement, which is discarded during translation, the non-tree links of control are preserved in the structure when the target language allows.

3. Semantics

3.1. The Semantic Hierarchy

Compreno captures the distinction between words and their meanings in the following way. On one hand, for every language in the system there is a collection of “words”, or lexemes, based on a standard morphological dictionary. On the other hand, there is a collection of language-independent “meanings”, or semantic classes, in the form of a thesaurus hierarchical tree. Finally, there are language-specific lexical classes, which serve as points of contact between the two collections. They fit into the universal semantic hierarchy (as children of semantic classes), but they also have a language-specific lexeme, or several morphologically related lexemes (e.g. face and facial).

Phraseological and terminological collocations can serve as children of semantic classes as well; but the mechanism is beyond the scope of this paper.

With the semantic hierarchy and a morphological dictionary in place, lexicographic description consists mostly in creating lexical classes, so that every word is connected to at least one class (≈ has at least one meaning). It is also desirable, though not always possible, for every semantic class to have at least one language-specific incarnation. The lexicographic description of Russian and English is reasonably complete; the work on German, French and Chinese is in progress.

Classes of the upper levels are classes denoting general notions — such as entities or actions. Classes of the lower levels represent more particular notions. The resulting taxonomy mostly varies from 3 to 10 levels (unlike “natural” taxonomies that are limited to five levels, according to [1]). The deeper the terminal branch of the taxonomy is the more specific notion it can contain, e. g. *’freely convertable currency’* is located on the 6th level of the hierarchy while ‘money’ is on the 4th:

FREELY CONVERTABLE CURRENCY < CURRENCY < MONEY < INFORMATION AND SOCIAL OBJECTS < ENTITY < ENTITY-LIKE CLASSES.

The upper-level semantic classes (in this case, INFORMATION AND SOCIAL OBJECTS, ENTITY, ENTITY-LIKE CLASSES) usually contain other semantic classes. The terminal branches (in this case, FREELY CONVERTABLE CURRENCY), by definition, contain no sub-branches, but only “leaves”, language-specific **lexical classes** (in this case, English term *‘freely convertible currency’*, Russian term *‘свободно конвертируемая валюта’* and Russian abbreviation for the term — *‘СКВ’*). The intermediate classes (CURRENCY and MONEY) can include both lexical classes and semantic classes: i. e., MONEY includes lexical classes *‘деньги — money’* as well as semantic classes CURRENCY, DEPOSIT, FUND, INTEREST, SAVINGS and so on. Lexical classes adjoining semantic classes (like *‘money’* here) are hyperonyms for the classes located below.

Lexical class is often a set of several words with the same root: the lexical class *‘деньги’* (‘money’), for instance, includes lexemes *‘деньги’*, *‘денежный’* (‘monetary’) and *‘безденежный’* (‘moneyless’). The lexemes *‘деньги’* and *‘денежный’* differ in syntactic category only, while *‘безденежный’* also has evident semantic distinctions. So lexical classes (as well as semantic classes) usually represent not a single meaning, but rather a set of closely related meanings.

All words the hierarchy contains are provided with grammatical and semantical information. We refer to the semantic information units as semantemes by analogy with grammemes (more often the term seme is used for similar purposes).

Semantemes are language-independent meaning elements that perform several functions. **Distributional semantemes**, for instance, are used for grouping classes with similar properties from different branches of the semantic tree which facilitates to describe the semantical compatibility of such classes (i. e., ‘plants’ and ‘metals’ both have a <<Substance>> semanteme, ‘soup’ and ‘tears’ — semanteme <<Liquid>>). **Differential semantemes** help to differentiate lexical items within one semantic class (*‘fat’* has a <<PolarityPlus>> semanteme vs *‘thin’* — <<PolarityMinus>>; *‘бабки’* (slang word for ‘money’) differs from neutral *‘деньги’* with a <<SocialStatus-Low>> semanteme).

The descendants of one lexical class that differ in semantemes are called **semantic derivates**. The above-mentioned *‘безденежный’*, for example, is a semantic derivate of the lexical class *‘деньги’* marked with a semanteme <<NotToHave>> and thus is translated as *‘moneyless’* with the same semanteme.

There are as well derivates, especially verbal, that are formed by regular morphological models, express the same semantical relations and differ from the ‘neutral’ derivate in the semantic valencies they can have. For instance, verbs like *‘вшить — sew in, вклеить — glue in, ввязать — knit in’* express the semantic relations of contact with another object and localization inside the other object, and are all formed with the same means — ‘в-’prefix in Russian and ‘in’-particle in English, which influences on the semantic valencies they can have as well.

Such semantic derivates are marked with **derivatemes** — set combinations of corresponding grammemes and semantemes, which helps to describe both the syntactic and the semantic features of these derivates (for more detailed analyses see [9]). Now we are in a position to define several lexicographical terms in our system. Antonymy is a relation between same-language lexical classes that are children of the same semantic class and have opposing semantemes (e.g. "warm" with <<PolarityPlus>> and "cold" with <<PolarityMinus>> in the class "TEMPERATURE"). Synonymy is a relation between same-language lexical classes that are children of the same semantic class and have identical semantemes. Translation equivalence is a relation between different-language lexical classes that are children of the same semantic class and have identical semantemes; that is to say, translation equivalence is cross-language synonymy. Translation of words (in the trivial case) is a transition: source_lexeme → source_lexical_class → semantic_class → target_lexical_class → target_lexeme. Hyperonymy/hyponymy is basically a relation between two same-language lexical classes, one of which is a child of the other's distant ancestor (e.g. "animal" is a child of "ANIMAL", which is a distant ancestor of "cat"). And so on.

3.2. Semantic slots

The semantic links and relations between words are expressed with the help of **semantic slots**, which, to some extent, correlate with semantic valencies in L. Tesnière's dependency grammar theory [7], deep cases in Ch. Fillmore's case grammar theory [3] or semantic and thematic roles in later linguistic models and conceptions.

An important difference is that the earlier theories as well as later studies generally focus on verbal arguments, underlining the difference between complements and modifiers, while in Compreno project all possible semantic dependencies are taken into account. I.e., there are not only semantic slots corresponding to widely used semantic roles such as [Agent] in *'[the boy] works'* or [Instrument] in *'the letter is written [with a pen]'* but also characteristical slots like [Ch_Evaluation] in *'[beautiful] dress'* or [Ch_Emotion] in *'He looked [surprisingly] exhausted'*, parenthetical slots like [ParentheticalSpecification] in *'you, [for example]'* and plenty of others: [RepresentedFormOfObjectOrCharacteristic]: *'rain [in large drops]'*, [Function]: *'work [as a teacher]'*, [Specifier_Number]: *'gate [1]'* and so on, more than 300 slots in total.

Semantic slots are language-independent objects, like semantic classes, and acquire their surface syntactic realizations in every language. I.e., semantic slots correspond to surface, or syntactic, slots like $Subject, $Object_Direct, $Modifier_Attributive and so on (surface slots are marked with the '$' sign).

The semantic hierarchy is organized according to the **inheritance principle**: many slots are introduced on the upper levels and the child semantic classes inherit them. For instance, locative and temporal adjuncts as well as some characteristical slots like the above-mentioned [Ch_Evaluation] are introduced on the very top of the hierarchy as such constituents can be governed by almost any cores: *'a book [on the table], most important [in the world], working [at school]'*.

Conditional and concessive clauses, in turn, are usually governed by verbal classes only, so the [Condition] and [Concession] slots are introduced on a lower level and cores with entity-like semantics don't inherit them. Constituents with the semantics of motive (such as *'to love smb. [for his talent], to criticize smb. [for wanting* *to break the rules]'*) or attributes like *'powerful'*, *'twenty-watt'* are attached just to the classes with rather particular semantics, so [Motive] and [Ch_Parameter_Power] slots are introduced even lower.

Another important restriction on the semantical compatibility is a **filling** of the semantic slots: each semantic slot can be filled with a strict set of the semantic classes. I.e., [Agent] is mainly filled with beings, organizations and some territorial units: *'[we/our school/Russia] agrees, that…'*, while [Condition] slot can be filled with any verbal classes.

Slots with similar semantic roles but different fillings are grouped in classes. Thus, [Agent_Class] includes [Agent_Route], [Agent_Device] and [Agent_Metaphoric] besides [Agent] itself. Whereas [Agent] is a slot widely used with different cores, [Agent_Route], for instance, is a slot that mainly verbs of motion have. It is filled with classes like 'ROAD', 'STAIRS', 'RAILWAYS' and other possible 'routes'. Introducing such a slot helps to describe regular metaphors like *'[the stairs] went up, [The railway line] follows the coast'* and to avoid creating corresponding homonyms for the motion verbs in the hierarchy (as it is usually done in most dictionaries).

Another strategy to describe selectional restrictions is using a widely filled slot, which can be introduced on a relatively high level and narrowed lower on some particular classes: i. e., [Object] slot can normally be rather widely filled (*'to see [a boy/a house/somebody's beauty/uncertainty]'*), but some verbs — like *'eat'*, *'drink'* or *'smoke'* — demand the narrowing of its filling (here is when the above-mentioned distributional semantemes help).

To avoid the 'repairing contexts' problem (compatibility problem occurring in contexts allowing the violations of the selectional restrictions, such as *'I'll eat [my hat] if Kim ate [a motor-bike]'* [5]), we define two sets of fillers for each semantic slot: the allowed one and the preferred one. So when the narrowing is necessary, generally only the preferred fillers are reduced.

4. Disambiguation

In Compreno, we talk of **homonymy** (not distinguished from polysemy) in a situation, when one lexeme belongs to several lexical classes, and of **synonymy** — when one semantic class has several lexical classes with the equivalent set of semantemes.

At the early stages of parsing we build the syntactic tree from lexemes, leaving the semantic ambiguity unresolved as long as possible. By delaying the choice of a specific lexical class, we gain computational efficiency.

The syntactic relations between words can also be rather ambiguous. English nominal premodifiers are a good example: cf. <u>*street*</u> *fight* and <u>*sword*</u> *fight* (with "street" and "sword" denoting place and instrument respectively). I.e., **syntactic homonymy**, or polysemy, occurs in a situation when one syntactic relation has several semantic interpretations. **Syntactic synonymy**, in turn, occurs in a situation when one semantic relation has several syntactic realizations (e. g. *John's father = the father of John*).

To deal with this effectively, we distinguish syntactic and semantic relations (through introducing the above-mentioned semantic and syntactic slots), just as we distinguished lexemes and semantic classes. Syntactic relations are language-specific, while semantic relations are language-independent. Translation of constructions (in the trivial case) is a transition: source_syntactic_slot → semantic_slot → target_syntactic_slot.

At the early stages of parsing the edges of the syntactic graph are labeled with syntactic relations only. The semantic ambiguity is kept unresolved as long as possible. By delaying the choice of a specific semantic relation, we also gain computational efficiency.

References

1. *Cruse D. A.* (1986), Lexical semantics, Cambridge.
2. *Sant-Dizier P., Viegas E.* (1995), An introduction to lexical semantics from a linguistic and a psycholinguistic perspective, in P. Sant-Dizier, E. Viegas (eds.), Computational lexical semantics, Cambridge, pp. 1–29.
3. *Fillmore Ch.* (1968),The case for case, in E. Bach, R. Harms (eds.), Universals in linguistic theory, New York, Holt, Rinehart and Winstonm, pp. 1–90.
4. *Pike K. L.* (1963), A syntactic paradigm, Language, vol. 39, No.2, pp. 216–230.
5. *Soehn J.-Ph.* Selectional Restrictions in HPSG: I'll eat my hat! Proceedings of the HPSG-2005 Conference. University of Lisbon, Portugal. Stanford, CSLI Publications, 2005, pp. 343–353.
6. *Stump G. T.* (2002), Morphological and syntactic paradigms: a theory of paradigm linkage, in G. Booij, J. van Marle (eds.), Yearbook of Morphology. 2001.
7. *Tesnière L.* (1959), Éleménts de syntaxe structurale, Paris, Klincksieck.
8. *Gruntova E. S.* Reguljarnye modeli upravlenija russkih pristavochnyh derivatov [Regular subcategarization frames of Russian prefixal verbs]. Komp'juternaja lingvistika i intellektual'nye tehnologii: Trudy mezhdunarodnoj konferencii «Dialog'2006» [Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference “Dialog 2006”]. Moscow, 2006.
9. *Zaliznjak A. A.* (1967), Russkoe imennoe slovoizmenenie [Russian nominal inflexion], Moscow.
10. *Plungjan V. A.* (2011). Vvedenie v grammaticheskuju semantiku: grammaticheskie znachenija i grammaticheskie sistemy jazykov mira [Introduction in grammatical semantics: grammatical values and grammatical systems in languages of the world]. Moscow.