A Peircean Framework of Syntactic Structure

József Farkas and Janos Sarbo

University of Nijmegen, The Netherlands
janos@cs.kun.nl

Abstract. A semiotic framework for the syntactic structure of language is introduced. From properties of syntactic signs a parsing algorithm is derived. Using English as an example it is shown that, by means of its syntactic structures, the English language implements signs, analogous to those of Peirce's semiotic triads.

1 Introduction

Traditional models of natural language take as their starting point that hierarchical structure is somehow given. However, this assumption is sometimes too rigid, and this may partly explain the limited success of such models in natural language processing. The approach proposed in this paper is based on Peirce's semiotic ([5]) which provides us with a deeper foundation of language. In this approach, which is monostratal, hierarchical structure arises, via the interaction of language symbols, as a result of linguistic semiosis. Because interactions are events, language may be considered a set of symbol-events, a process. We apply the process view of language to syntactic (and morphologic) structure, and by using English as an example we illustrate the potential of the Peircean framework in parsing ([6]). A comparison with the 'generative' approach, from a philosophical point of view, can be found in [3].

2 The Peircean view

Peirce's semiotic is strongly related to his categories. In his doctrine of categories Peirce states that all phenomena present three aspects which, though irreducible to one another, have a different degree of dependency. The aspect of firstness is the aspect by virtue of which each phenomenon has an absolutely novel *quality*, unrelated to anything whatever. The aspect of secondness is the aspect by virtue of which each phenomenon involves an *interaction*. The aspect of thirdness is the aspect by virtue of which each phenomenon involves some *habit* (lawfulness, meaning etc.).

Though secondness cannot be reduced to firstness, it presupposes firstness, and, similarly, though thirdness cannot be reduced to either firstness or secondness, it presupposes both firstness (through secondness) and secondness. This dependency of the categories is formalised by the ordering firstness<secondness<thirdness, where "<" is a total order on categories.

2.1 Peirce's semiotic

Peirce's early papers suggested a convergence of his theory of the three categories and his presentation of the various semiotic triads. The most important of these triads is the triad of sign, object, and interpretant, which is a kind of *ontological* triad telling us what there is in the world. Based on the ontological triad, Peirce defined three triads of sign: the triad of icon, index, and symbol, dealing with how signs refer to their objects; the qualisign, sinsign and legisign triad, referring to the sign itself, prior to any relational possibilities and actualities; and the triad of rheme, dicent sign and argument, characterising the formal rules that associate signs and objects. These triads, which are called the *relational*, *material* and *formal* triad ([7]), are shown in fig. 1.

		Ontological type		
		1 Material	2 Relational	3 Formal
Phenomenological type	1 Quality	Qualisign	Icon	Rheme
	2 Indexicality	Sinsign	Index	Dicent sign
	3 Mediation	Legisign	Symbol	Argument

Fig. 1. *Peirce's classification of signs*

The above order relation on categories can be applied to Peirce's signs, according to their category exhibited. For example, Icon<Index<Symbol is an expression of degeneracy with respect to the realisation of the sign's object and interpretant; Qualisign<Icon<Rheme is an expression of degeneracy with respect to the sign's ontological type.

2.2 Language as a process

Language appears only as a form of interaction, whether we speak it, write it, or read it. In terms of Peirce's categories, interactions, or events, represent the category of secondness. An event involves the fact *that* something happens, but says nothing whatever about *what* happens. The latter aspect is the aspect of thirdness. *What* happens in an event requires that the event be embedded in a context of events which are related to each other. Such web of related events is what is called a *process* ([2], [3]).

Language consists of symbols which are signs. Because signs are generated from signs, and in turn generate other signs, every sign must be related to an event. From this it follows that language symbols are sign-events which, by virtue of their intepretants, are embedded within a process.

3 Language and ontological perspective

Language is a process involving symbol-events which are themselves generated according to rules which, in Peircean terms, are habits evolving from interaction with other symbol-events. Language processes involve both syntactic and semantic rules or habit.

Linguistic symbols may be considered as gesture-events within a process of interactive responses. Syntactic symbols may be considered from two angles: the messenger and the receiver. From the point of view of the messenger, a syntactic symbol is a gesture announcing other gestures to be generated in view of the interpretant of the entire unit of meaning (e.g. a sentence). From the point of view of the receiver, a syntactic symbol-event elicits an abduction regarding a range of possible subsequent symbol-events. Thus, in English, the symbol-event 'the' elicits an indefinitely large field of possible subsequent symbol-events, but excludes, for instance, the possibility of the next symbol-event being 'is'.

From a syntactic point of view, symbol-events have a specific function, regardless of their semantic function. The syntactic value of the language symbols making up the unit of meaning may be seen in the function of the value which they have in forming such a unit.

By virtue of their secondness, events are marked by a binary relation. Therefore, linguistic symbol-events must be also binary. This is why, strictly speaking, one lexical item by itself has no meaning. The syntactic value of the symbol-events will therefore depend upon the *sort* of relation that obtains between two language symbols. If one of the symbols has by itself no information content and therefore is a mere quality (a phoneme or a visible character), it will need another symbol to actualise its 'potential' content. Such nexus of two symbols, one of which is self-sufficient, but the other has mere potential content, may be called a *proto-symbol* (P) which corresponds to the category of firstness. An example of this is the symbol-nexus of free morpheme and affix.

Similarly, when the nexus is constituted by an asymmetrical relation between one language symbol which derives its full content from its association with another language symbol which is in principle self-sufficient, it may be called a *deutero-symbol* (D) which corresponds to the category of secondness. An example of this is the symbol-nexus of adjective and noun, or the one of determiner and noun.

Finally, when the nexus consists of two language symbols which are self-sufficient but together generate the interpretant of the unit formed by the string, e.g. a sentence, it will be called a *trito-symbol* (T) which, by its aspect of thirdness, mediates between the language symbols constituting a unit of meaning, or a thought, in the Fregean tradition. An example of this is the symbol-nexus between verb and subject.

To complete the picture, it is necessary to say a word about the *triadic relation* characterising each of these signs, because without such relation, they would not be signs, let alone syntactic signs. But precisely what makes them *syntactic* signs is the very fact that they stand for specific *rules* or habits. Thus, the object of syntactic signs is the rule for which they stand. Their interpretant on the other hand is the generation of the selection of the next symbol-event. The interpretant of the entire string of language symbols is, from a syntactic point of view, the establishment of the correctness of the string, regardless of its semantic content.

Inasmuch as linguistic symbols are also syntactic symbols, proto-, deutero-, and trito-symbols constitute a Peircean triad of linguistic symbols. By virtue of their category exhibited, these signs define the ordering P<D<T which in turn defines the *levels* of syntactic signs.

In the remaining, we will denote by X a level of syntactic signs, and by X' the level subsequent to X. A sign (or symbol-event) of some level X will be called an X-level sign (or symbol-event).

3.1 Classes of syntactic sign levels

Language implements syntactic signs basically by lexical items and their relations. These are called *syntactic structures* or, equivalently, *language units*, depending on whether we want to emphasise their structural or linguistic properties. In the mapping of syntactic signs to syntactic structures (*syntactic mapping*), the notion of argument and functor, an abstraction from the combinatorial properties of lexical items, plays a crucial role. This combinatorial property can be characterised as relational or argumental need. A lexical item has *relational need* if it can be a functor, and *argumental need* if it can be an argument in some relation.

By analysing the structure of the three types of syntactic sign, we can recognise an argument and a functor symbol in each of them. In the case of trito symbols, the functor is that symbol which has the most relational need in the determination of the interpretant. We tacitly assume that such a distinction can always be made.

We denote the constituents of an X-level symbol-event, the argument and the functor symbol, and the syntactic symbol itself as X_1, X_2 and X_3. By virtue of the category and dependency which different signs respectively exhibit, syntactic signs may be said to define the ordering $X_1 < X_2 < X_3$ which in turn defines the *classes* of level X. The total order on levels and classes can be extended, by flattening, to a total order on syntactic signs.

The syntactic sign emerging from a symbol interaction is called its *descendant*. A syntactic sign that has no combinatorial need is a *completed* or well-formed sign. Two symbols are said *incompatible* if they cannot establish a relation syntactically, and *compatible*, otherwise. A completed sign is incompatible with any symbol. A sign of class X_i (i=1,2,3) of some level X is denoted an X_i sign.

3.2 The emerging syntactic sign

From a receiver's point of view, input symbols have merely potential content according to the receiver's (parser's) hypothesis. The set of such hypotheses is called the parser's dictionary. Input symbols appear one after the other, interact, and syntactic signs emerge by *symbol relation*. This might be called the 'automatic' type of sign generation. Because the descendant sign contains, besides the meaning of its constituents, the additional meaning of the relation itself, the signs generated by symbol relation are monotonously increasing.

But there are also cases of a degenerate symbol interaction. One of them is the interaction between incompatible symbols. In such a case, one of the interacting symbols, which is a sign generated in one symbol-event, is coerced to an argument or a functor, but *not* both, in another symbol-event. This type of sign generation is called *symbol coercion*.

Symbol coercion affects the interpretant of one of the symbols, but leaves the other symbol unchanged. Accordingly, we will say that the symbol coerced increases its meaning, and enters a higher class and/or level of syntactic signs. By virtue of their more developed interpretants, the signs generated by symbol coercion are monotonously increasing.

From the monotonicity property of syntactic signs, it follows that a lower level combinatorial need must have priority over a higher level one. The handling of one combinatorial need may require the elaboration of another, recursively.

Syntactic signs are composite signs which meet certain criteria. Thus, if a syntactic sign consists of related signs, the signs involved must in principle be *contiguous* to one another. This requirement is based upon the triadic structure of a syntactic sign the object of which is always a rule expressive of the expectation that a certain type of language unit must be followed by another type of language unit. The contiguity property can be defined as a covering relation on syntactic signs (classes which are empty are transitively closed).

The contiguity property is the driving force behind symbol coercion. Let us assume that two symbols S_1 and, subsequently, S_2 are recognised, which are incompatible. Assume, furthermore, that S_2 has combinatorial need on some higher level. Then S_1 is forced to enter a higher class without symbol relation. This follows from the assumption that the entire input, e.g. a sentence, is a well-formed syntactic sign. Due to the monotonicity property, if S_2 (or its descendant) enters a higher level, then the contiguous signs S_1 and S_2 (or its descendant) will only be able to establish a symbol relation on a higher level. Therefore S_1 must enter a higher class, as well, conform to its combinatorial need.

Syntactic sign generation respects, besides the combinatorial need of the symbols, also their syntactic properties. For example, an interaction between an X-level sign, and a sign entering the X-level subsequently, is syntactically different from the one in which the two signs are interchanged.

3.3 Towards an algorithm for syntactic signs

The properties of symbol coercion are formalised as follows ($X_i \to Y_j$ denotes that a sign of class i of level X may enter class j of level Y, and $X_i \to Y_j \vee Y_k$ is a shorthand for $X_i \to Y_j \vee X_i \to Y_k$):

(α_1) $X_1 \to X_3$; (α_2) $X_1 \to X_1' \vee X_2'$; (α_3) $X_3 \to X_1'$.

In sum, X_1 and X_3 signs can increase their meaning without symbol relation, but an X_2 sign, due to its indexicality (aspect of secondness), presupposes an X_1 sign, and must relate with it. This meets our expectation that a relational need must be fulfilled always, though an argumental need can be optional.

Because language possesses a finite number of lexical items only, some syntactic signs must be generated *incrementally*, via degenerate symbol relations. In such a relation, the mediation aspect is incomplete, and the descendant of the X-level symbol interaction will become an X_1 or X_2 sign on the same level. Accordingly, the rules of symbol relation are formalised as follows (the symbol relation of X_1 and X_2 is denoted as X_1-X_2):

(β_1) $X_1\text{-}X_2 \rightarrow X_1 \vee X_2$; (β_2) $X_1\text{-}X_2 \rightarrow X_3$; (β_3) $X_1\text{-}X_2 \rightarrow X_1' \vee X_2'$.

3.4 Cumulative signs

The third type of sign generation is related to the incremental nature of syntactic signs, according to which, there may be encountered simultaneously more than one sign of the same class. By virtue of its aspect of firstness, an X_1 class may contain a number of signs which are *unrelated*, but the collection of which is a sign. By virtue of its aspect of secondness, an X_2 class may contain a number of signs which are *unrelated*, but which share a *common referent*. By virtue of its aspect of thirdness, an X_3 class may contain a completed sign which must be a single sign. In each case, the signs belonging to a class must have the same combinatorial need, and must be consistent with each other, with respect to their meaning (i.e. with the interactions involved in the signs).

Based upon the above properties of sign classes we refine our definition of incompatibility as follows. Symbols will be said *incompatible* if (i) they are of different classes of a level and cannot establish a relation syntactically, or (ii) they share the same class, but have different combinatorial need or meaning, or (iii) one of the symbols is a completed sign.

Besides symbol coercion, another case of a degenerate symbol interaction is the one between compatible symbols of the same class. In such a case, the symbols involved are accumulated on a stack. This type of sign generation is called *stacking*. The need for a stack is explained as follows. Assume that the subsequent symbols, S_a and S_b, enter the same class of some level X. Let S_a and S_b be such that they cannot enter a higher level, because their combinatorial need on level X is not yet fulfilled. As a consequence, both symbols have to be stored, temporarily. Let another symbol, S_c, enter the other class of level X. S_c and S_a are *not* contiguous symbols and, therefore, should not interact, however, S_c and S_b must interact. These requirements can be satisfied by storing S_a and S_b on a stack. A stack of signs is considered a single sign, represented via the topmost item of the stack. Symbols generated by stacking are, trivially, monotonously increasing.

The stacking of symbols is subject to restrictions implied by the property of contiguity, the binary nature of interactions, and the properties of symbol classes. Let us denote the state of some level X by the pair of stacks of X_1 and X_2 (the X_3 stack has no effect on the stacking of symbols and therefore omitted). Let s_i (i=1,2) denote a stack, $\perp$ the empty stack, s the next symbol entering level X, and '/' a left-associative stack constructor. Then, symbol stacking must respect the following rules, specified as transitions on the state of level X:

$(s_1, \perp) \rightarrow (s_1/s, \perp)$; $\quad (\perp, s_2) \rightarrow (s, s_2)$; $\quad (s_1, s_2) \rightarrow (s_1, s_2/s)$.

In sum, an X_1 stack can grow if the X_2 stack is empty, or otherwise, by a single item only (the common referent); and an X_2 stack can grow if the X_1 stack is constant.

3.5 Primary signs

We assume that the input symbols enter a lowest class (prm) as a sequence of primary signs, e.g. phonemes or characters. Prm, which has the aspect of firstness, is by definition a class of syntactic signs. The input primary signs, which have no combinatorial need, are collected in prm, as long as their sequence forms a morphological symbol which is a dictionary entry. When this happens, the symbol receives its combinatorial need from the dictionary, and enters the lowest level, in particular, P_1 if it has no P-level relational need; and P_2, otherwise:

(γ) prm$\rightarrow P_1 \vee P_2$.

3.6 Mediating evaluation

Syntactic signs are yielded by symbol interaction. But the decision as to *when* the mediation takes place depends upon the type of evaluation, which can be lazy or greedy. In general, we will assume lazy evaluation of relations, because it can be more economic in some cases.

The lazy evaluation of syntactic sign-events affects the modelling of the terminator symbol (e.g. the point symbol) which, therefore, will be treated as an incompatible argument and a nullary functor on each level, thereby forcing the realisation of pending relational needs.

In sum, syntactic signs arise in language due to (1) the quality of contiguity, (2) symbol interaction, and (3) mediating evaluation which, respectively, have the aspect of the categories, firstness, secondness and thirdness.

The emerging syntactic sign may become part of a cumulative sign, or change its aspect of correspondence with its object, or establish a relation with another sign. In sum, symbol interactions do emerge by (1) symbol stacking, (2) symbol coercion, and (3) symbol relation, which, again, exhibit the aspects of Peirce's categories.

4 Syntactic mapping

We illustrate the syntactic mapping of language by using English as an example. In this mapping we capitalise on the semiotic properties of syntactic signs and on the conceptual distinctions that may be expressed in English ([1],[4]).

Trito-symbols correspond to the symmetric relation between two constituents which are both self-sufficient and require the presence of the other, e.g. the relation between noun and verb (subject and predicate). Such a relation is called *predication*(p).

Deutero-symbols correspond to the asymmetric relation between an action/state or participant, and its properties: both are self-sufficient, and the latter requires the presence of the former, but the reverse does not hold. In English, two instantiations of this type of relation can be identified: *modification*(m), e.g. the relation between adjective and noun; and *qualification*(q), e.g. the relation between determiner and noun.

The third type of symbols, proto-symbols, correspond to the morphological process of *affixation*(a). An affixation relation distinguishes between a root (or base), e.g. a free morpheme, and an affix: the root is self-sufficient; the affix has only potential meaning actualised by the root.

From the semiotic point of view, q- and m-signs form subsets of deutero-symbols. Using the analogy of the relational triad, q- and m-signs represent iconic and indexical meaning, respectively, and, therefore, these signs may be said to define the ordering $q<m$. Eventually, this yields the ordering $a<q<m<p$ which in turn defines the *levels* of the English syntactic signs.

Though the above characterisation of the English syntactic mapping is based upon simple considerations, it may not be language model independent. In section 6 we will show, however, that this mapping is certainly not accidental. It will be argued that the English language implements signs by means of syntactic structures analogous to those of Peirce's semiotic triads.

4.1 Syntactic relations

Syntactic relations emerge due to the combinatorial need of syntactic symbols. In general, a syntactic symbol can have argumental need, optionally, but its relational need is a function of that of its constituents, or, in the case of a lexical item, it is some constant value. Lexical items can contribute to the relational need of syntactic signs, on each level.

A lexical item has a potential combinatorial need, which is a finite set. The combinatorial need of a syntactic symbol generated by a symbol relation is the disjoint *union* of the combinatorial need of its constituents, possibly modified (i.e. restricted) by the interaction itself. The potential relational need of the types of lexical items is exemplified in fig. 2 (respectively, a '+' or '-' represents the presence or absence of a relational need on the level indicated by the column). The relational need of a particular lexical item is the subset of that of its type.

For example, the q-level relational need of adjectives and adverbs allows symbol-events like *keep awake*, or *walk by*; and their m-level relational need the modification relation like *happy girl*, or *walk quickly*. In the case of a preposition, the q-level relational need contributes to the relation with the obligatory argument (qualification); and the m-level one to the modification of the optional argument by the qualification yielded.

Verb–complement relation is classified as modification, typically. Indeed, such a symbol-nexus fits the definition of a deutero sign: verb and complement are both self-sufficient, but the verb derives its full content from the complement. Because the descendant symbol of a verb–complement relation has an indexical character (the verb points in the direction of its complement it is acting on), this type of relation must be identical with modification (we admit that the terminology might be confusing for the linguist).

In sum, a verb relates with its complement(s) due to its m-level relational need (which is fulfilled when all necessary complements are found), and with the subject, due to its p-level one. A copula or an auxiliary relates with its complement due to its q-level relational need, but the copula relates with the subject due to its p-level relational need.

The development of the relational need of syntactic signs can be illustrated as follows. The potential m-level relational need of a preposition will be actual if the q-level relation it is involved in does not disallow that. For example, there will be such need in the case of *in London*, and there won't be, in the case of *drive in*.

	a	q	m	p
primary	-	-	-	-
affix	+	-	-	-
noun	-	-	-	-
determiner	-	+	-	-

	a	q	m	p
preposition	-	+	+	-
adjective	-	+	+	-
adverb	-	+	+	-
verb	-	+	+	+

Fig. 2. *Potential relational need*

5 Example

Below we give the analysis of the sample example: *Mary drank some wine yesterday*. We leave out the morphological analysis, and assume that the input signs leave the a-level and enter q_1 or q_2 conform to their combinatorial need.

The potential relational need of the lexical items is as follows: *Mary*=$\{\}$, *drank*=$\{$m,p$\}$, *some*=$\{$q$\}$, *wine*=$\{\}$, *yesterday*=$\{$m$\}$. Symbols having a non-empty relational need are written in capitals. In the table below, an item represents the content of the storage of a class (column) prior to the evaluation of the next input symbol (row).

step	next input	q_1	q_2	q_3	m_1	m_2	m_3	p_1	p_2	p_3
0	*Mary*(m)									
1	*drank*(D)	m								
2	*some*(S)	D		m						
3	*wine*(w)		S			D	m			
4	*yest*(Y)	w	S			D	m			
5	.	Y		s-w		D	m			
6	.	.		.	D-s-w	Y	m			
7	.	.		.	D-s-w-y		m			
8	.	.		.	.		.	m	D-s-w-y	
9	.	.		.	.		.	.		m-d-s-w-y

In step$_1$, the symbol *Mary* enters q_1. Next, the symbol *drank* enters q_1 (it has no q-level relational need, but it has argumental need, optionally). Because *Mary* and *drank* are incompatible on this level, therefore, q_1 (*Mary*) must increase its class by α_1. The next input, *some*, forces q_1 (*drank*) to increase its level by α_2 which, in turn, forces q_3 (*Mary*) to do the same by α_3 and α_1.

The rest of the evaluation is as follows (we specify for each step which rule is applied to which sign(s), but we omit the treatment of the point symbol):

step$_4$ $\beta_2(q_1,q_2)$; step$_5$ $\alpha_2(q_1)$, $\alpha_3(q_3)$, $\beta_1(m_1,m_2)$; step$_6$ $\beta_1(m_1,m_2)$;
step$_7$ $\alpha_2(m_1)$, $\alpha_3(m_3)$; step$_8$ $\beta_2(p_1,p_2)$.

6 English syntactic symbols and Peirce's signs

The ordering of the classes of a level is depicted in fig. 3a (edges represent the "$<$" relation). In the case that degenerate signs are allowed, this graph can be paraphrased as a two-level scheme consisting of a finite automaton (FA) and a number of stacks. A state of the FA corresponds to a sign class, and a transition to an application of an α or β rule, represented by solid and dashed lines, respectively (but in later graphs we will use solid lines for both types of transition). The resulting graph is depicted in fig. 3b (an edge which is a cycle is omitted).

In virtue of the syntactic mapping, the classes of English syntactic signs define a total ordering as shown in fig. 3c. By using the interpretation of fig. 3b to fig. 3c, we get a two-level system (this is not illustrated). We map the X_3 and X_1' classes, e.g. a_3 and q_1, to same states (same signs, different interpretants). The initial state is prm, all others are final states; each state has a stack. The output language of the system is the set of signs in the different stacks, upon termination.

We argue that the above system is terminating. This follows from the properties of the α and β rules. Indeed, an α rule increases the class of the sign it is applied to, but there are finitely many classes; and a β rule fulfils a combinatorial need, but the combinatorial need of syntactic signs is finite.

The priority of a lower level combinatorial need over a higher level one implies a stack-like behaviour of the levels of syntactic signs; the properties of symbol stacking imply a stack-like behaviour within a level of syntactic signs. It follows from this that the individual stacks of the classes can be simulated by a single stack. Eventually we get that our two-level system is equivalent to a context free (CF) grammar, and therefore, it has the same complexity as CF parsing.

Below we describe an equivalency transformation of our two-level system with respect to its input and output languages. In step1 (see fig. 4), we remove $\mathrm{m}_2 \rightarrow \mathrm{p}_1$. If the descendant sign m_1-m_2 has no m-level combinatorial need and enters the p-level, then, equivalently, we can push it to the m_1-stack, and let the m-level sign generated subsequently force it to enter the p-level by α_2.

In step2 (cf. fig. 5), we close the $\mathrm{q}_1 \rightarrow \mathrm{m}_1 \rightarrow \mathrm{p}_1$ transitions by $\mathrm{q}_1 \rightarrow \mathrm{p}_1$. Because these transitions only apply when the m-level stacks are empty, the storages are not affected by this transformation. Furthermore, we merge q_2 and m_1, because of the orthogonality of their signs (respectively, functor and argument signs), and the incompatibility of the q_1 and m_1 signs (both are argument signs). The q_2 stack is 'pushed' upon the m_1 stack, thereby providing that the property, that q_2 signs represent lower level combinatorial need, and therefore, have priority over m_1 signs, holds invariantly after the transformation.

Last, in step3 (see fig. 6a), we remove $a_2 \to q_1$ (cf. step1 above). Furthermore, we close $a_2 \to q_2 m_1 \to m_2$ by $a_2 \to m_2$ (the stacks are not affected by this transformation), and remove the redundant transition $q_1 \to m_2$. The resulting system is depicted in fig. 6b (edge directions are omitted).

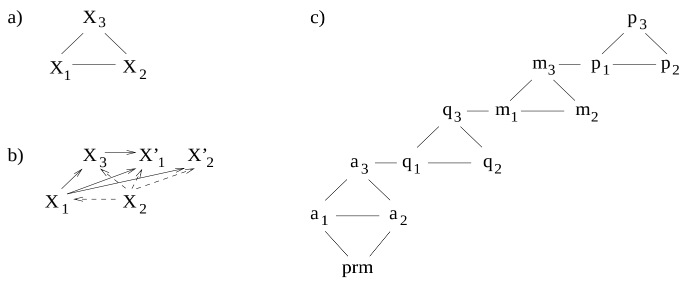

Fig. 3. *English syntactic signs*

Fig. 4. *Transformation step1*

The system of fig. 6b can be interpreted, the other way round, as a specification of a parsing algorithm, and also as a classification of syntactic signs (except for degenerate ones): states can be mapped to sign classes, the ordering of which is derived from the transitions, as shown in fig. 6b (cf. fig. 3a). A comparison of this classification with Peirce's triads shows their isomorphism (the reader might turn fig. 1 in 135° to help noticing this). The analogy between the corresponding signs is justified as follows:

Fig. 5. *Transformation step2*

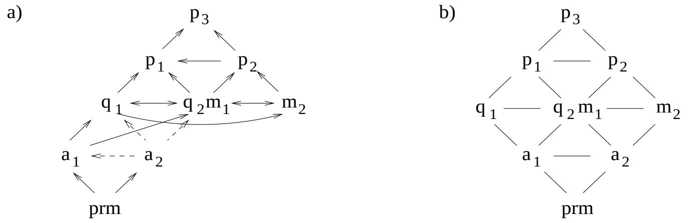

Fig. 6. *Transformation step3*

prm a pure quality, unrelated to anything else; a *primary.*

a_2 a particular quality, referring to an actually existing argument; an *affix.*

m_2 a sign involving the convention that arguments have certain properties which are general types; a *modifier.*

a_1 an image, a name of some 'thing', e.g. a free morpheme; a *root.*

q_2m_1 a morphological sign (a_3), or a qualifier (q_2), or a developing m-level sign (cf. section 3.3), each involving a reference to the argument; we call these signs collectively, a *relative.*

p_2 a sign involving the convention that arguments have some more basic properties; a *predicate.*

q_1 a sign representing the possible existence of some 'thing'; a *noun.*

p_1 a sign used to assert the actual existence of something, for example, a clause; a *nominal.*

p_3 a sign expressing a lawlike relation between subject (p_1) and predicate (p_2), a 'thought'; a *sentence.*

This completes our English syntactic mapping. In fig. 7, it is illustrated how English implements the signs of the 'real' world, syntactically. The upward-right diagonals represent the material, relational and formal ontological types; and modifier-, predicate- and nominal-formation, on the left- and righthand side of fig. 7, respectively; and, similarly, the upward-left diagonals represent the quality, indexicality and mediation phenomenological types; and, respectively, word-, expression- and sentence-formation.

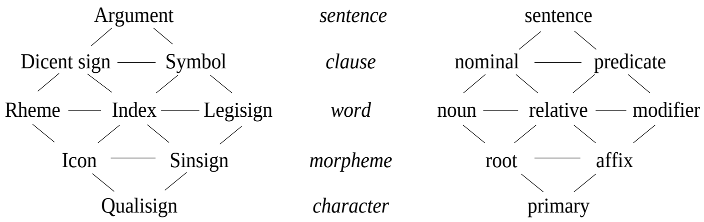

Fig. 7. *Peirce's signs and English syntactic signs*

Acknowledgements

We wish to thank Guy Debrock and Vera Kamphuis for the invaluable discussions during the course of the above work, and also the anonymous referee for the helpful remarks.

References

1. Aarts, F., Aarts, J.: English syntatic structures. Pergamon Press, Oxford (1982)
2. Debrock, G.J.Y., Sarbo, J.J.: Towards a Peircean model of language. Technical Report CSI-R9802, University of Nijmegen (1998)
3. Debrock, G.J.Y., Farkas, J., Sarbo, J.J.: Syntax from a Peircean perspective. In: Sandrini, P. (ed.): Proceedings of 5th International Congress on Terminology and Knowledge Engineering (in press). Innsbruck, Austria (1999)
4. Farkas, J., Kamphuis, V., Sarbo, J.J.: Natural Language Concept Analysis. Technical Report CSI-R9717, University of Nijmegen (1997)
5. Peirce, C.S.: Collected Papers of Charles Sanders Peirce. Harvard University Press, Cambridge (1931)
6. Sarbo,J.J.: Formal conceptual structure in language. In: D. M. Dubois (ed.): AIP Conference Proceedings of the American Institute of Physics (in press). Springer-Verlag (1999)
7. Sowa, J.F: Knowledge Representation: Logical, Philosophical, and Computational Foundations. PWS Publishing Company, Boston (1998)