Marcus Kracht [1]
II. Mathematisches Institut
Arnimallee 3
D – 14195 Berlin

The Combinatorics of Cases

October 25, 2000

ABSTRACT: In this paper we will introduce a semantics for languages with stacked cases. In these languages, the burden of building up the structure is carried entirely by the morphology, and syntactic structure is redundant. This is the exact opposite of standard Montague Semantics, which is based on λ–calculus, where morphology is redundant and the meaning is computed from the structure alone. Furthermore, we shall demonstrate that only case marking languages can afford free word order. Although the languages fall out of standard hierarchies of languages (they are not even multiply context–free since they are not semilinear) we will show that their complexity is very low, with an upper bound of $O(n^{3/2} \log n)$. The semantics is also of low complexity and far easier to implement than λ–calculus.

[*]This paper has been prepared in long discussions with Alan Dench and Jens Michaelis, both of whom have contributed significantly to the genesis of this paper. The stay of Alan Dench at Potsdam has been made possible through the Innovationskolleg INK 12. Furthermore, the paper contains ideas presented at a workshop on Morphology and Complexity of Language at the University of Potsdam, in December 1996, likewise financed through the Innovationskolleg INK 12. I wish to thank the audience there, especially Gisbert Fanslow, Ed Keenan and David Perlmutter for constructive criticism. Thanks also to Christian Ebert for discussions on the complexity of case marking. Two anonymous referees have helped me with their critical assessment. Above all, I am grateful to Dick Oehrle for his generous help and numerous comments.

1 Introduction

The dominant paradigm in formal semantics ever since its beginnings in the 70's has been Montague Semantics. In Montague Semantics, the meaning of a sentence is computed on the basis of its structural description. Although Montague Semantics is potentially very powerful, it cannot explain the role of agreement and case marking. From its point of view, both should be redundant, since we have structure to give us all the information we need. In fact, Oehrle [19] notes that

> [...] such grammars (GPSG and standard categorial grammars, MK) would be simplified if agreement relations [...] failed to exist. The same point holds for a wide variety of other syntactic theories: the 'government–and–binding' theory of Chomsky (1981), the 'relational grammar' of Perlmutter and Postal, the 'lexical–functional grammar' of Bresnan and Kaplan.

We have argued elsewhere (see [12]) that assuming highly articulated syntactic structures is the wrong route to take. If however we assume that there is not enough structure to begin with, then agreement or case marking is no longer redundant. In this paper, we shall explore the potential of case marking as a substitute for syntactic structure. We shall show that in the presence of case marking of a specific kind, no syntactic structure whatsoever is needed. [1] The basic mechanism needed to make this work is called *suffixaufnahme*. [2] We shall develop an idealized model for such languages, called *ideal case marking languages*. Although this model is mainly of theoretical significance, it may have far reaching consequences for the study of formal semantics, some of which we shall indicate at the end of this paper.

The idea that cases contribute information concerning the functor–argument relationship is not new. This view is implicit in the school

[1] This does not mean that no structure actually exists. However, its presence is not strictly required. More on this will be found in later discussions, cf. Section 8.

[2] Literally translated it means 'taking up of suffixes', see [21] for the history of this term.

grammars of classical languages. Moreover, Nordlinger [18] has already claimed that cases can be a substitute for syntactic structure and proposed an account within LFG. Some authors have also tried to analyse the various ways in which functor–argument relationships are encoded in natural languages, for example Oehrle [19] or Nichols [17]. There have also been discussions on the relationship between free word order and case–marking (see for example Steele [22] to name just one). However, these papers have remained sketchy on the theoretical level. The literature has laid almost exclusive emphasis on the variation that is actually observed in languages themselves. However, without knowing the theoretical limits of dependent marking there is no proper understanding of the nature of natural languages. For example, we shall show that head–marking languages are necessarily ambiguous, while case–marking languages need not be. We expect therefore that free word order should occur more often with case–marking languages than with head–marking languages. If this is not so — as has been claimed — then this is actually a remarkable property and deserves close investigation.

Cases are difficult for language theory because of their dual nature. On the one hand they do service in syntax by marking arguments for their syntactic status and on the other hand cases also have a meaning, although to a varying degree. Core cases tend to be more syntactic, while other cases (for example locative and instrumental cases) have a clearly definable semantic meaning, which makes them more semantic in nature. The degree to which a given case in a given language is semantic or syntactic differs from one construction to another, so it is by no means fixed. One and the same case (for example the allative in Finnish) can on one occasion be completely semantic (in which case it is typically an adverbial, indicating the movement of some object towards the allative marked NP) and the next time a purely syntactic one (being selected by some verb, for example).

While the semantic side of cases is surely interesting, we shall deal in this paper exclusively with the syntactical side of cases. We consider cases in their role as devices to keep track of the syntactic status of an argument. Our investigation will be held theory neutral. This is why we have chosen to speak of the combinatorics of cases. We shall investigate in what ways cases can help to keep track of arguments. As we shall show, the more elaborate the case marking system, the freer the word order is allowed to be. This approach has hardly been prominent in the last decades. All syntactic theories we know of study the mechanics of cases independently of any word order phenomenon (a notable exception is the dissertation by Nordlinger [18]). Moreover, we find in Baker [2] the remarkable statement that head–marking languages tend to have free word order, while case marking languages do not. (Steele [22] seems to imply the same.) As we shall show in the beginning of this paper, there are purely combinatorial arguments that this is unlikely. Several empirical arguments against this view will also be adduced.

The paper is organized as follows. We begin with an overview of language strategies to encode syntactic relations. It will emerge that there are three systems which yield unambiguous languages: Polish Notation, Reverse Polish Notation and case marking, provided that no functor selects the same overt case twice. We shall subsequently develop a semantics for case marking languages that is flexible enough for both types of case marking: group and word marking languages. We will then compare this semantics to Montague Semantics. Finally, we will pick up some remaining issues on the relationship between grammatical restrictions and semantical restrictions.

2 Three Regimes

In semantical as well as syntactical terms the main problem in understanding a sentence is to know which of the words is the head or functor and which of the others are which arguments of the functor. This problem can be looked at in two ways: as a *language choice* or as a *speaker choice*. If looked at as a speaker choice, we will talk of *strategies*. A *strategy* is a way to say clearly and unambiguously what one wants to say in a given language. If looked at as a language choice we will talk of *regimes*. A *regime* is a fixed way of encoding relations in a language. Strategies and regimes are of course closely connected. Strategies depend on the regimes of the language. To allow for a formal definition of regimes, we shall fix a language of semantical structures, which should be unambiguous. We may take, for example, semantic derivations in Montague Grammar. The regime is then a recipe of 'sugaring' (to use a term by Aarne Ranta) a derivation into a legitimate string of the language with the corresponding meaning. Sugaring is a process that eliminates parts of the structure and adds some functional elements in its place.

Let us illustrate the problem with a thought experiment. Imagine a language, *Ektian*, with only three words, eg, which means 'cat', ag, which means 'dog', and the word um, which means 'to chase'. [3] There are no determiners, so eg may mean, depending on circumstances, either 'a cat' or 'the cat'. But this will hardly matter. Now consider the English sentence (2.1) and its semantic translation. [4]

(2.1) The cat chases the dog.
$\mathsf{chase}'(x, y) \wedge \mathsf{cat}'(x) \wedge \mathsf{dog}'(y)$

Suppose we want to translate this sentence into Ektian. We certainly expect the words eg, ag and um to occur exactly once. [5] But, in what order will they appear? What if anything is added to these words? In particular, which additional information (order, markers,

---

[3] This is an adaptation of a language described by Franz Hohler in his *Wegwerfgeschichten*. It might be worth explaining why we engage in a thought experiment rather than taking real language examples. The reason is simply that a fictitious language can be made to function any way we please. So we can make it consistently SOV, for example. If we take real language example we always run the risk that one objects that such and such construction is actually not SOV, and that one finds marginal constructions of type VSO, and so on. Such objections will however never matter for what we really want to say and so we have opted for the thought experiment.

[4] In what is to follow we shall pay no attention to the distinction between definite and indefinite NPs. Quantifiers are suppressed when not needed. So, in the present example we should have used the definite description operator ι. Likewise, everything is extensional. These extra complications, though necessary in dealing with language in general, would simply obscure the issue at hand if added.

[5] Dick Oehrle has rightly pointed out to me that we may code for example accusative by means of reduplication. However, reduplication seems to be a rather iconic device in language, restricted to formation of plural, intensives and so on — however also in changing category (eg Indonesian baling *to rotate* and baling baling *propeller*). Therefore, reduplication will be excluded from consideration hereafter.

etc) does Ektian use to identify the subject and the object? There are three basic choices, [6] all exemplified in natural languages.

1. **Inferential** or **Null Regime.** Subject and Object must be inferred from their meaning and the context. *Example.* Lusi. [7]
2. **Positional Regime.** It is agreed that the elements follow in a certain order, for example subject first. *Example.* English, Chinese.
3. **Marking Regime.** The words get markers that allow one to disambiguate the sentence.
   (a) **Head Marking Regime.** The head is marked for some quality of the subject and the object. We write as follows

$$N_1 \quad N_2 \quad n_1\text{–}n_2\text{–}V$$

   Here n_1 and n_2 are markers (which we write as prefixes, even though they may also be suffixes or clitics etc) that reflect a property of N_1 and N_2 (gender, class etc). We call these markers also *agreement markers*. Also, n_1 is the *reflex* of N_1 in V. Some agreement markers may be absent. *Examples.* Mohawk, Bantu languages.

   (b) **Dependent** or **Case Marking Regime.** The head identifies the subject by a subject marker and the object by an object marker. These markers are called *cases*. We write as follows.

$$N_1\text{–}\gamma_1 \quad N_2\text{–}\gamma_2 \quad V$$

   Note that γ_1 and γ_2 must be distinct in order for this to disambiguate the sentence. *Example.* Tagalog.

Before we continue, some remarks are in order. We assume by default that case marking languages do not have functors selecting the same (overt!) case twice. If for example, some verb in German

---

[6] There are of course infinitely many regimes. So, this classification owes much to the existing regimes in natural languages and is by no means complete. Below we will exemplify some more occurring regimes which do not fit this scheme.

[7] We take this example from [20], who quotes Li and Thompson [15] for the claim that Lusi does not distinguish subject and object. Also, Dick Oehrle (p. c.) has pointed out to me that colloquial Japanese allows to drop case markers when they can be recovered in the context.

assigns accusative twice, the encoding is actually imperfect. Further, with respect to head marking languages, two cases arise. (a) The set of subject reflexes and the set of object reflexes are disjoint. Then it is not necessary to order them in a word, and omission of, say, the object reflex marker does not create ambiguities. (b) The two sets are not disjoint. Again, this can give rise to ambiguities. These deficiencies will be discussed to some extent later.

Let us illustrate the regimes with Ektian. If it uses the inferential regime, then any permutation of the sequence eg ag um means the same, all circumstances being equal. Furthermore, eg ag um means both 'the cat chases the dog' as well as 'the dog chases the cat'. In the positional regime it is declared, for example, that the subject precedes the object (this is in fact all that is needed in the present case). Then the sentences are unique in meaning. Thus, eg ag um only means 'the cat chases the dog', while ag eg um only means 'the dog chases the cat'. When the marking regime is employed, we may either have cases, say -su for subject and -ob for object. Then any permutation of the sequence eg-su ag-ob um will mean 'the cat chases the dog', and any permutation of the sequence eg-ob ag-su um will mean 'the dog chases the cat'. If we finally choose the head marking regime, we shall have certain prefixes, say e- for cats and a- for dogs, and then any permutation of eg ag e-a-um means 'the cat chases the dog', while any permutation of eg am a-e-um means 'the dog chases the cat'. So much for an illustration of the basic regimes.

The case marking regime actually splits into two further subregimes. The first is called *group marking* and the second the *word marking regime*. Group marking languages use only one item of a case marker, which appears either peripherally or on the head. (For example, assume that Ektian has a word bul, which means 'blue'. Then 'blue cat' in the nominative is rendered as bul-su eg-su if Ektian is word marking, and as bul eg-su if it is group marking. Alternatively, we might have bul-su eg or even eg bul-su, depending in the way group marking works in particular.) Examples of group marking languages are Japanese, Turkish, and Hungarian. In word marking languages, the case marker is iterated on every single word of the constituent. This definition is actually somewhat inexact, and we shall return to this question below. Examples are Finnish, German, Latin and Russian.

Some notes are in order. First, languages usually use a mixture of these regimes. Indo–European languages usually have subject verb agreement in addition to case marking (if present). The interested reader is referred to Plank [21] for an exhaustive list of the simple and combined marking regimes existing in natural languages. Furthermore, as we shall see, in languages the systems are never pure enough to guarantee full disambiguation (for example, in German there is no distinction between nominative and accusative in feminine and neuter nouns as well as all plural nouns). In that case some other regime jumps in. The interplay between different regimes caused by syncretism, for example, has to our knowledge not been subject to typological investigation and can lead to misunderstandings in applying the labels free/non–free word order. (See Section 7 for a discussion.) Second, a marker is to be understood as an abstract entity. For example, intonation is also added as a cue for disambiguation. In the German case, the subject–first preference jumps in as soon as the cases fail to distinguish nominative from accusative. This can be overruled by adding intonation. Also, the context may help (then we are in Regime 1). In what is to follow, we will exclude both possibilities. Third, the markers can be suffixes or words or clitics. To keep matters simple, we often act as if they are part of their host word, neither clitics nor separate words. However, this will hardly make a difference. [8]

Before we go on, we note some direct consequences. The head marking regime and the dependent marking regime are very distinct in the following sense. If the head marking regime is chosen, the reflex n_1 of N_1 depends on the actual choice of N_1. This can be

[8] Terminologically, it is simpler if we think of the markers as affixes. For example, we will show that pure word marking languages are uniquely readable even if they have completely free word order. However, this can only be true if the markers are not separate words. Otherwise, the association between a word and its marker is simply lost. In case the markers are appositions, they must be exempt from permutation. This fact explains why appositions tend to be very strict in the adjacency requirement with respect to their complements. Too much depends on finding out the argument status of constituents.

a semantic property or what we call a *ritualized* property. Here by *ritualized* we mean roughly *must be learnt by heart.* These two, semantic and ritualized head marking, are not sharply distinguished. Gender systems in Indo–European languages are semantically motivated but to a large extent ritualized. If the reflex is semantic it can be inferred from the meaning of N_1. It may well be that the reflex depends on the phonetic form of N_1, but we ignore that choice as it makes no difference in principle. A neutral term in place of *semantic* is *inferential.* In contrast to head marking languages, dependent marking languages need to record in the lexicon what the cases of the actants must be. Again, this can be inferential (i. e. mainly semantic) or ritualized. The first option is realized in Manipuri (a Tibeto–Burman language, see [5]) and the second in Indo–European languages. Mixtures do occur. In Finnish, the case of the direct object varies between accusative and partitive depending on the meaning of the sentence and whether the object is entirely affected by the action etc.

Let us close with a remark on the null regime. In a language using only this regime, (2.2) (and any permutation thereof) is ambiguous between (2.2a) and (2.2b).

(2.2) eg ag um.
(2.2a) *The dog chases the cat.*
(2.2b) *The cat chases the dog.*

This can lead to the following regime. It is fixed that the sentence means only one of the above. Typically, arguments are ranked (for agentivity) and the higher ranked argument takes the role of subject. [9] To generate the other meaning some other sentence must be used. An obvious possibility is to have a marker on the verb (let it be inv) that exchanges the role of subject and object. So, if (2.2) expresses (2.2a), (2.3) will express (2.2b), and if (2.2) expresses (2.2b) then (2.3) will express (2.2a).

[9]The diligent reader may notice that nothing is said in the case when they have equal rank. Languages have developed means to avoids this. In Plains Cree, for example, third person arguments are assigned two possible discourse functions: proximate or remote. The proximate argument is ranked higher than the remote argument. It is not allowed to have more than one proximate argument.

(2.3) eg ag um-inv.

This leads to a regime known as *inverse marking*. An example is Plains Cree (see [23]). In principle the problem could be solved also by marking an argument as being 'demoted'. So, assume that the suffix dem means 'is demoted' (which is attached to the subject and says that it becomes the new object). Then if (2.2) expresses (2.2a), (2.4) now expresses (2.2b), and if (2.2) expresses (2.2a) then (2.5) expresses (2.2b).

(2.4) ag-dem eg um.
(2.5) ag eg-dem um.

(A note of clarity. If regimes other than the Positional Regime are employed, the sentence above may in principle be replaced by any permutation of its words with identical meaning.)

The list above is surely not complete. We could in principle invent a new word per (for permuted) which functions just like inv, which however can appear anywhere in the sentence. (See also the discussion surrounding the reflexes, whether they are separate words.) We know of no language where this option is chosen. There are obvious shortcomings of this regime, though it is of course not prima facie excluded.

Some of the regimes discussed earlier have some gaps in them. Here to contain a *gap* means that there exist sentences which are functionally ambiguous if that regime is used alone. Here, a language is *functionally ambiguous* if there is a string which has two meanings with different functor–argument structure. A sentence is *ambiguous* if it can have two meanings, judging from the mere string alone. An unambiguous language is always functionally unambiguous but the converse need not hold. This is the case when the language has scope ambiguities, for example, since scope ambiguities are independent of the functor–argument relations. (This was brought to my attention by Dick Oehrle.) Since we do not have much to say about scope ambiguities, we shall simply not deal with them for rest of the paper.

Though a sentence may mean one thing in one context and another in a different context, it has two meanings, if the context is not specified. This allows us to measure the disambiguating potential of the regime alone. The higher the disambiguating potential of the regime, the less context dependent the language is. We begin with the inferential regime. Here matters are clear: any sentence where a verb has two arguments is ambiguous.

Now consider head–marking. Suppose that there are two NPs with identical reflex. Then it is not possible to decide which of the two is subject and which of the two is the object. We summarize this in the following observation.

> *Observation.* The Inferential Regime has gaps if there exist heads selecting two different arguments. The Head Marking Regime has gaps if there exist two arguments of the same head with identical reflex.

The conditions mentioned in the observation are always satisfied in natural languages. There always exist transitive verbs in any language we know of. Furthermore, head marking languages typically use some classification system that distiguishes a few types of things. These are gender or class systems. A more elaborate case is noun incorporation languages. In such languages, object agreement is realized by a noun of the language rather than an agreement marker of the usual sort. If we made our language into a noun incorporation language, we get the following sentences:

(2.6) eg ag-um
the cat dog-chase
the cat chases the dog

(2.7) ag eg-um
the dog cat-chase
the dog chase the cat

However, even this is not enough even though it avoids many of the abovementioned problems. It so happens in these languages that there is no incorporation when there is talk of people with names. So, we do not say in such languages John Paul-hit, rather we say John Paul hit (or John hit Paul, depending on the word order type).

So, in these examples there is no incorporation. [10] Typically, this does not mean that there is no marking at all. Mohawk, for example, also has a gender system. But, as we have seen, gender systems have gaps. This concludes the discussion of incorporation.

What about the other two regimes? We will see shortly that in both cases there exist idealized languages which have no gaps, that is, in which every sentence is (at least functionally) unambiguous! For languages with word order regimes this was shown already in the 20's. This is what we shall now turn to.

3 The Positional Regime: Polish Notation

It was observed by Łukasiewicz in the 20's that it is possible to write down unambiguously any formula without the use of brackets. All that needs to be done is to put the function symbol always first (alternatively: always last). So, rather than writing `(x + y) + (z - 3)` one must write `++xy-z3`. This way of writing down formulae is known as Polish Notation. It has hardly gained much popularity except in some logical literature and in some pocket calculators. For reasons we shall not go into these calculators use the so–called Reverse Polish Notation, which differs from the Polish Notation in that the function symbol is put last rather than first. You first enter the arguments (say, first 7 and then 14, separated by carriage return) and then you press the function key (say, $\times$), rather than pressing first 7 then $\times$ and finally 14.

A note on our notation. We shall use typewriter font to denote true characters in print for a formal language. This helps to distinguish between a metavariable and a variable. For example, x is a metavariable, but `x` is a variable, used by the language itself. Moreover, concatenation is sometimes denoted by using the symbol $^\frown$, especially when dealing with symbols that merely denote strings. Between true characters in print, however, $^\frown$ is superfluous

[10] One wonders if there can be a formal proof that incorporation is insufficient. We are quite confident that this is so, but it needs a precise definition of incorporation to begin with. If the verb can simply form an entire constituent with its object, as seems to be possible in Mohawk, then of course no ambiguities arise from this regime.

and omitted.

An abstract definition of terms runs as follows. Let F be a set of symbols, and $\Omega : F \to \mathbb{N}$ a function, where $\mathbb{N}$ denotes the set of natural numbers including zero. F is assumed to be finite throughout this paper. The pair $\langle F, \Omega \rangle$ is called a *signature*. We shall often write Ω in place of $\langle F, \Omega \rangle$. $\Omega(f)$ is called the *arity* of f.

Definition 3.1
Let Ω be a signature. The set of **terms** over Ω (or Ω–**terms**) is the least satisfying the following conditions.

1. If $\Omega(f) = 0$, then f is an Ω–term.
2. If $\Omega(f) > 0$ and t_i, $i < \Omega(f)$, are Ω–terms, so is $f(t_0, \ldots, t_{\Omega(f)-1})$.

Terms are represented in Polish Notation by strings over F. The set of terms is denoted by Tm_Ω. Define a mapping ^{p} from Tm_Ω to strings over F in the following way. (Notice that throughout this paper we start counting from 0, not 1.)

1. $t^p := t$, if $t \in F$ and $\Omega(t) = 0$.
2. $(f(t_0, \ldots, t_{n-1}))^p := f^\frown t_0^{p\frown} \ldots {}^\frown t_{n-1}^p$, if $n = \Omega(f) > 0$.

We say that $\vec{x}$ *represents* t if $t^p = \vec{x}$. Furthermore, we shall write PN_Ω for the set of strings representing some $\langle F, \Omega \rangle$–term. PN_Ω is context–free. Here is a grammar that generates it:

$$\begin{array}{lcl}
\mathtt{S} & \to & \mathtt{F}_0 \\
\mathtt{S} & \to & \mathtt{F}_1\mathtt{S} \\
\mathtt{S} & \to & \mathtt{F}_2\mathtt{SS} \\
 & & \ldots \\
\mathtt{F}_0 & \to & \mathtt{f}_0^1 \mid \mathtt{f}_0^2 \mid \ldots \\
\mathtt{F}_1 & \to & \mathtt{f}_1^1 \mid \mathtt{f}_1^2 \mid \ldots \\
\mathtt{F}_2 & \to & \mathtt{f}_2^1 \mid \mathtt{f}_2^2 \mid \ldots \\
 & & \ldots
\end{array}$$

(Here, $\mathtt{f}_i^j$ is a symbol for the jth function of arity i.) Call this grammar G_Ω. We shall note some properties of this grammar. Before we can do so, we have to define a few notions from the theory of context–free languages. A grammar G assigns to a string $\vec{x}$ a set of derivations. In turn, each derivation defines a parse tree in the usual way (see [9]). G is called *ambiguous* if there is a string that is assigned more than one parse tree. A language L is context–free iff it is recognized by some nondeterministic pushdown automaton. L is called *deterministic* iff it is recognized by some deterministic pushdown automaton.

Let $\vec{x}$ be a string. A *substring occurrence* of $\vec{y}$ in $\vec{x}$ is a pair $\langle \vec{u}, \vec{v} \rangle$ such that $\vec{x} = \vec{u}\vec{y}\vec{v}$. Given a parse tree (or analysis) $\mathcal{P}$ of $\vec{x}$, we can associate with each node of the tree a substring occurrence of some substring of $\vec{x}$. Given $\mathcal{P}$, such a substring occurrence is called a *constituent* of $\vec{x}$ under the analysis $\mathcal{P}$. A constituent of $\vec{x}$ under some analysis $\mathcal{P}'$ which is not a constituent of $\vec{x}$ under $\mathcal{P}$ is called an *accidental constituent occurrence* of $\vec{x}$ in $\mathcal{P}$. Now, a context–free language is called *transparent* if no constituent occurrence in a given string and analysis is accidental. [11] From this a deterministic parser is easily constructed, for example a shift–reduce parser. Just note that whenever it finds the right hand side of a rule on its stack, it can assume that this rule has been applied and it can reduce the stack using that rule. This shows that a transparent language is also deterministic.

THEOREM 3.2
Let Ω be a signature. Then G_Ω is a context–free transparent grammar generating PN_Ω. Consequently, PN_Ω is deterministic.

One can show that each string represents at most one term.

PROPOSITION 3.3
Suppose that $\vec{x}$ represents t and s. Then $t = s$.

This follows from the transparency of the grammar and the fact that the parse trees are in one–to–one correspondence with the Ω–terms. Now we shall establish the following terminology. [12]

[11] This terminology as well as the results are due to Kit Fine ([8]).

[12] This definition is somewhat roundabout in order to take care of the term representations that will follow. In actual fact, this definition must remain somewhat obscure for the reason that term representations need not be straightforwardly homomorphic.

Definition 3.4
Let $\vec{x}$ be a string. $\vec{x}$ is called a **constituent** if it represents a term. Suppose that $\vec{x}$ represents $t = f(s_0, \ldots, s_{n-1})$ and suppose that the $\vec{y}_i$ are disjoint occurrences of substrings, where $\vec{y}_i$ represents s_i, $i < n$. Finally, suppose that $\vec{y}_i$ occurs to the left of $\vec{y}_j$ whenever $i < j$. Then we shall call f the **head** of $\vec{x}$, and $\vec{y}_i$ the ith **argument** of $\vec{x}$.

The main advantage of Polish Notation is that it does not use any syncategorematic symbols, ie brackets. Its disadvantage is among others the complete regimentation of word order, and the fact that all arguments must be present. If we allow certain arguments to be dropped, ambiguity arises. Furthermore, a symbol has exactly one arity. If that is not the case, again ambiguities may arise. For example, let $\mathtt{f}$ have arity 0 or 1 and $\mathtt{g}$ arity 2. Then the following string is ambiguous: $\mathtt{gfff}$. It may either represent $\mathtt{g}((\mathtt{f}(\mathtt{f}), \mathtt{f})$ or $\mathtt{g}(\mathtt{f}, \mathtt{f}(\mathtt{f}))$. [13]

The semantics of such a language is straightforward to define. [14] An Ω–algebra is a pair $\langle A, I\rangle$, where A is a non–empty set and I a function sending each $f \in F$ to an $\Omega(f)$–ary function on A. So, the symbol $\mathtt{+}$ will denote a binary function on the set of numbers, while $\mathtt{0}$ denotes a 0–ary function on A (ie a constant). A complete expression is a term without variables. The value of a complete expression is then simply an element of A. For example, the term $\times(\mathtt{+}(\mathtt{3}, \mathtt{5}), \mathtt{7})$ is evaluated in the natural numbers (with the usual interpretation of $\mathtt{+}$ and $\times$) to the number 56. Notice that terms such as $\times(\mathtt{+}(\mathtt{x}, \mathtt{5}), \mathtt{7})$ have no meaning since $\mathtt{x}$ is a variable.

Montague deviates from this picture in the following way. He starts with typed first–order logic and defines λ–terms on top of it. He assumes that there are only unary functions, whose values can also be functions rather than objects (Currying). In this way he gets full binary branching, but uses typing in an essential way. So, if f is

[13] In what is to follow we shall exclude that a symbol has more than one translation, in particular that it has more than one arity. Of course, there are examples in natural languages where this is not so, but if we disregard argument omission, this is actually a negligeable phenomenon for the purpose at hand.

[14] Throughout this paper we shall ignore the question of typing. This will simplify the discussion considerably.

an n–ary symbol, the term $f(t_0, \ldots, t_{n-1}$ is computed by applying (the Curried equivalent) of f to t_0, the result to t_1 etc. So we are effectively computing the value of $(\ldots((f t_0)t_1)\ldots)t_{n-1}$. In order to make this work for natural language, Montague deviates from linguistical practice and makes the object the functor in the VP and the subject the functor of the entire sentence. Moreover, he treats quantified phrases as sentential adjuncts, in order to get quantifiers scope ambiguities for free. His solution however overgenerates, and so it is probably better to obtain scope inversion by adding explicit rules. If we therefore step back from this, we more or less end up with the following basic system. Even though the verb is the head (and therefore intuitively plays the role of f here), it is an argument of the associated function of the object. This is necessitated by the need to give a compositional analysis of quantifiers. The Ektian DP eg in its meaning 'a cat' is translated into two distinct formulae, depending on whether it constitutes an object NP or a subject NP:

$$\lambda\mathcal{P}.\exists x.\mathsf{cat}'(x) \wedge \mathcal{P}(x), \qquad \lambda\mathcal{Q}.\lambda y.\exists x.\mathsf{cat}'(x) \wedge \mathcal{Q}(y)(x)$$

The verb um is translated by

$$\lambda x.\lambda y.\mathsf{chase}'(y, x)$$

where — finally — $\mathsf{chase}'(y, x)$ means that y is chasing x.

If we assume the canonical regime with the functor last, only SOV languages are generated. So, only the sentences (3.1) and (3.2) are grammatical and they mean (3.1a) and (3.2a), respectively.

(3.1) eg ag um
(3.1a) $\exists y.\exists x.\mathsf{cat}'(y) \wedge \mathsf{dog}'(x) \wedge \mathsf{chase}'(y, x)$
(3.2) ag eg um
(3.2a) $\exists y.\exists x.\mathsf{dog}'(y) \wedge \mathsf{cat}'(x) \wedge \mathsf{chase}'(y, x)$

In fact, Montague Semantics allows for more choices in the positional regime, but we shall be concerned here only with this case.

If one wants to insist — as we shall need to do — that the verb is the head of the construction, there is an easy fix. [15] We raise

[15] In fact, taking the verb a the functor rather than the argument appears already in Keenan and Faltz [11]. There are differences in technical execuction, but we shall not be concerned with the details here anyway.

the verb over its object and its subject and assume that it has the following meaning:

$$\lambda\mathcal{O}.\lambda\mathcal{S}.\mathcal{S}(\mathcal{O}(\lambda x.\lambda y.\mathsf{chase}'(y,x)))$$

Here, $\mathcal{S}$ is a variable over subject NPs and $\mathcal{O}$ a variable ranging over object NPs. It is still the case that only SOV and VOS languages are generated, but now the semantics is in line with the syntactic analysis.

We shall derive the translation of the sentence

(3.4) A woman sees every man.

We start with sees every man.

$$\begin{array}{ll} \multicolumn{2}{l}{(\lambda\mathcal{O}.\lambda\mathcal{S}.\mathcal{S}(\mathcal{O}(\lambda x.\lambda y.\mathsf{see}'(y,x))))(\lambda\mathcal{Q}.\lambda y.\forall x.\mathsf{man}'(x) \to \mathcal{Q}(x)(y))} \\ = & \lambda\mathcal{S}.\mathcal{S}((\lambda\mathcal{Q}.\lambda y.\forall x.\mathsf{man}'(x) \to \mathcal{Q}(x)(y))(\lambda x.\lambda y.\mathsf{see}'(y,x))) \\ = & \lambda\mathcal{S}.\mathcal{S}(\lambda y.\forall x.\mathsf{man}'(x) \to \mathsf{see}'(y,x)) \end{array}$$

Next we apply this to some woman.

$$\begin{array}{ll} \multicolumn{2}{l}{\lambda\mathcal{S}.\mathcal{S}(\lambda y.\forall x.\mathsf{man}'(x) \to \mathsf{see}'(y,x))(\lambda\mathcal{P}.\exists x.\mathsf{woman}'(x) \wedge \mathcal{P}(x))} \\ = & (\lambda\mathcal{P}.\exists x.\mathsf{woman}'(x) \wedge \mathcal{P}(x))(\lambda y.\forall x.\mathsf{man}'(x) \to \mathsf{see}'(y,x)) \\ = & \exists x.\mathsf{woman}'(x) \wedge (\forall x'.\mathsf{man}'(x') \to \mathsf{see}'(x,x')) \end{array}$$

So, this generates the desired result.

4 The Case Marking Regime

4.1 Group Marking

Now we shall move to case marking regimes. Again, we assume that $\langle F, \Omega\rangle$ is a finite signature. Now, let $\mu := max\{\Omega(f) : f \in F\}$. For each $i < \mu$ we take a distinct symbol c_i (which plays the role of the case marker). (Often, we shall also write $\mathsf{0}$ in place of c_0, $\mathsf{1}$ in place of c_1, and so on.) All case markers are distinct and $\mathsf{c}_i \notin F$ for all $i < \mu$. The case markers can be thought of as unary function symbols. Therefore, we may think of the language as a language over an enriched signature, which we denote by Ω^γ.

Define the *right peripheral group marking language* as follows. The map ^{r} is defined by

1. $t^r := t$, if $t \in F$ and $\Omega(t) = 0$.
2. $(f(t_0, \ldots, t_{\Omega(f)-1}))^r := t_0^r \mathtt{c}_0 t_1^r \mathtt{c}_1 \ldots t_{\Omega(f)-1}^r \mathtt{c}_{\Omega(f)-1} f$, $\Omega(f) > 0$.

Suppose that $\vec{x} = t^r$ for some t. We understand the notions of constituent, head and argument as defined in the previous section. Then let $\vec{y}$ result from $\vec{x}$ by permuting within a constituent the arguments among each other. (This induces a permutation of the occurrences of the $\vec{y}_i$ representing the immediate subterms s_i. After permutation, the occurrence of $\vec{y}_i$ continues to be the ith argument of t. Cf. Definition 3.4 for the notation.) Then $\vec{y}$ is said to *rpg–represent* t. In particular, $\vec{x}$ rpg–represents t. For example, the arithmetical term $t := \mathtt{-}(\mathtt{+}(\mathtt{a}, \mathtt{b}), \mathtt{+}(\mathtt{x}, \mathtt{y}))$ is translated by ^{r} into

$$\mathtt{ac_0bc_1{+}c_0xc_0yc_1{+}c_1{-}}$$

(Notice that the first argument is $\mathtt{ac_0bc_1{+}}$ and the second one $\mathtt{xc_0yc_1{+}}$. Hence, the occurrences of the arguments and the functor do not add up to the entire word! For the case markers of the arguments belong to neither of them. In particular, it does not permute like the members of F (see also Footnote 8 on this point).) But also the following string rpg–represents t:

$$\mathtt{xc_0yc_1{+}c_1bc_1ac_0{+}c_0{-}}$$

DEFINITION 4.1
Let Ω be a signature. Then RPG_Ω denotes the set of all strings rpg–representing some Ω-term.

THEOREM 4.2
RPG_Ω is context–free. It is transparent and therefore deterministic.

The following is a grammar for this language:

$$\begin{array}{lll@{\qquad}lll}
\mathtt{S} & \to & \mathtt{F_0} & \mathtt{S} & \to & \mathtt{Sc_1Sc_0Sc_2F_3} \\
\mathtt{S} & \to & \mathtt{Sc_0F_1} & \mathtt{S} & \to & \mathtt{Sc_1Sc_2Sc_0F_3} \\
\mathtt{S} & \to & \mathtt{Sc_0Sc_1F_2} & \mathtt{S} & \to & \mathtt{Sc_2Sc_0Sc_1F_3} \\
\mathtt{S} & \to & \mathtt{Sc_1Sc_0F_2} & \mathtt{S} & \to & \mathtt{Sc_2Sc_1Sc_0F_3} \\
\mathtt{S} & \to & \mathtt{Sc_0Sc_1Sc_2F_3} & \mathtt{S} & \to & \mathtt{Sc_0Sc_2Sc_1F_3} \\
 & & & & & \ldots
\end{array}$$

(The rules for expanding $\mathtt{F}_n$ are as before in G_Ω.) Call this grammar H_Ω. For a linguist, this grammar seems to miss important generalizations. However, this is just a matter of notation not of substance, which can easily be resolved.

THEOREM 4.3
Let Ω be a signature. Then H_Ω is a transparent context–free grammar generating RPG_Ω. Consequently, RPG_Ω is deterministic.

Proof. Let h be the following string–homomorphism. $h(\mathtt{c}_i) := \varepsilon$, $h(f) = f$ for all $f \in F$. It is easy to see that if $\vec{x}$ rpg–represents some term, then $h(\vec{x})$ also represents some term (though not necessarily the same one). Furthermore, given a parse tree of $\vec{x}$, every constituent occurrence of $\vec{y}$ is mapped onto a constituent occurrence of $h(\vec{y})$. This is seen by showing that if h is applied to a derivation in H_Ω, it yields a G_Ω derivation. This shows the claim on the basis of Theorem 3.2. Q. E. D.

We note the following corollary.

COROLLARY 4.4
Let $\vec{x}$ rpg–represent s and t. Then $s = t$.

One can play different variations on that theme, allowing some more freedom or placing some restrictions. However, the following variant yields an ambiguous language. Suppose that we allow the head to be placed anywhere between its arguments. Suppose that in this case it takes its case marker along. This type of regime is a group marking regime, where only the head is marked, but is free in its relative position. Then unary function symbols may be either before their arguments or following them. The following string is then ambiguous: `f0a0g0h`. (Recall that we write `0` in place of $\mathtt{c}_0$.) It may represent either `h(f(g(a)))` or `h(g(f(a)))`. On the other hand, if we assume that the head is free in its position but the case marker remains right peripheral, then we once again get a transparent language. (This applies in particular to complementizers, which are typically peripheral. The analogy between complementizers and case markers is not perfect, but actually quite close.) In this language, the case markers mark the right edge of the constituent. One can show, namely, that there are no accidental minimal constituents, and the same arguments go through once again. Let us call the language RC_Ω. It too is context–free. Here is a grammar generating it:

$$\begin{array}{lcl}
\mathtt{S} & \to & \mathtt{F}_0 \\
\mathtt{S} & \to & \mathtt{Sc}_0\mathtt{F}_1 \mid \mathtt{F}_1\mathtt{Sc}_0 \\
\mathtt{S} & \to & \mathtt{Sc}_0\mathtt{Sc}_1\mathtt{F}_2 \mid \mathtt{Sc}_0\mathtt{F}_2\mathtt{Sc}_1 \mid \mathtt{F}_2\mathtt{Sc}_0\mathtt{Sc}_1 \\
\mathtt{S} & \to & \mathtt{Sc}_1\mathtt{Sc}_0\mathtt{F}_2 \mid \mathtt{Sc}_1\mathtt{F}_2\mathtt{Sc}_0 \mid \mathtt{F}_2\mathtt{Sc}_1\mathtt{Sc}_0 \\
 & & \dots
\end{array}$$

Theorem 4.5
RC_Ω is a transparent, deterministic context–free language.

The same applies to the mirror image of this language, where case markers are consistently left peripheral. All these case marking languages (with the exception of the language where the head is case marked) have the following properties.

1. The case marker is consistently right (left) peripheral.
2. All semantic constituents are continuous, that is, subwords of the string.

To define the notion of a semantic constituent, we shall go back to the definition of a string representing some term. If $\vec{x}$ represents t, we find a correspondence between subterm occurrences in t and subparts of $\vec{x}$. We shall not spell out the details here. Suffice it to say that in this way each subterm occurrence corresponds to a unique substring of $\vec{x}$. The definition can be made independent of the actual grammar generating the language. This is why we call this a semantic constituent. The idea is to replace the subterm by a fixed symbol, say Z, and to observe in what ways we must change $\vec{x}$ for it to represent the new term.

The two abovementioned properties constitute a positional regime. The second condition restricts the word order in such a way that whatever is a constituent must be a substring. The first condition comes down to a positional regime as soon as the case markers are words rather than morphemes. [16] Continuity of constituents is the rule rather than the exception. NPs tends to be continuous in virtually all languages we know of, and deviations from this do occur but are severely restricted. For example, Kayardild has a case marking regime that allows for discontinuous constituents without creating (functional) ambiguities. Yet, [7], page 249 states that 'NP-splitting obeys precise rules and has a clear semantic rationale. It is always a single modifier being split off; split NPs always straddle a verb.'

4.2 Word Marking

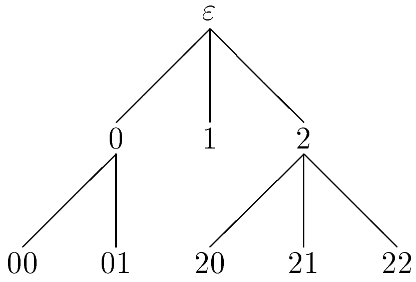
Word marking brings us to the last type of string representations, which will define a language that needs no positional regime. Let ^{w} differ from ^{r} in that the case markers are not distributed at the end of each phrase, but at each individual word. Then what we get is the ideal model for a word marking language. Unfortunately, this language is not definable by means of a string homomorphism (this can be shown: otherwise the language would be semilinear which it isn't). So, we must choose a different approach. We shall first define a *set* representing a term and on the basis of that set we define the strings. The definitions are based on the notion of a *tree domain*. A *tree domain* is a subset T of $\mathbb{N}^*$ such that the following holds.

1. $\varepsilon \in T$.
2. If $\sigma\sigma' \in T$ then $\sigma \in T$.
3. If $\sigma i \in T$ and $j < i$ then also $\sigma j \in T$.

Tree domains are useful because they allow us to write down a tree using a set of sequences of natural numbers. We can define the

[16] In formal language theory this distinction is hardly made. However, in ordinary usage as well as in programming languages, the following terminology is used. There is a blank symbol $\Box$, and given a string $\vec{x}$, each maximal substring not containing $\Box$ is called a *word* of $\vec{x}$. Words may be composed from smaller units by simple concatenation, while a word is a string with a right peripheral boundary marker (see Section 8). Regimes in our sense are defined over words. We may therefore call them *syntactic regimes* to distinguish them from *morphological regimes*. In virtually all languages we know of, there is a morphological regime. (We have been told though by Farrell Ackerman that there exist languages in Siberia where the order of verbal affixes is free.) We will turn to this issue in Section 8.

FIG. 1. A Tree Domain



relations $<$ (less than) and $\sqsubset$ (to the left of) as follows: $\tau < \tau'$ iff τ' is a proper prefix of τ, and $\tau \sqsubset \tau'$ iff there exist σ, i, j such that $\tau < \sigma^\frown i$ and $\tau' < \sigma^\frown j$ and $i < j$. (Figure 1 shows the tree domain $\{\varepsilon, 0, 00, 01, 1, 2, 20, 21, 22\}$.) Notice that for every subset of $\mathbb{N}^*$ it is decidable whether or not it is a tree domain, and that two tree domains are identical iff they define isomorphic trees. Now, the problem in parsing a sentence is that we are given a string of symbols but we have to guess the syntactic structure. As we have just seen, the syntactic structure can also be given by means of a tree domain.

The tree domain forms the tree on which we hang the term symbols. An Ω–*labelled tree domain* is a pair $\langle T, \ell\rangle$ such that T is a tree domain and $\ell : T \to F$ a function such that the number of daughters of $\sigma \in T$ is exactly $\Omega(\ell(\sigma))$. With a term we can associate a canonical labelled tree domain t^δ as follows. If $t = g$, $\Omega(g) = 0$, we associate with t the tree domain $\{\varepsilon\}$, where ε gets the label g. If $t_i^\delta = \langle T_i, \ell_i\rangle$, are given for each $i < \Omega(f)$, we form a new tree domain $\langle S, \ell\rangle$ as follows:

$$\begin{array}{lcl} S & := & \{\varepsilon\} \cup \bigcup_{i<\Omega(f)} \{i^\frown\sigma : \sigma \in T_i\} \\ & := & \\ \ell(\sigma) & := & \begin{cases} f & \text{if } \sigma = \varepsilon \\ \ell_j(\tau) & \text{if } \sigma = j^\frown\tau \end{cases} \end{array}$$

This is the tree domain associated with $f(t_0, \ldots, t_{\Omega(f)-1})$. It is easy to see that this representation is unique.

Define the transpose σ^T to be the transpose of σ (ie the string written in reverse). For example, $\mathtt{213}^T = \mathtt{312}$. [17]

DEFINITION 4.6
A **bag over** Ω is a set of the form $\Delta(\mathfrak{T}) := \{f^\frown\sigma : \ell(\sigma^T) = f\}$, where $\mathfrak{T}$ is an Ω–labelled tree domain. A **partial bag** (over Ω) is a subset of a bag.

Let us give an example. The terms $\mathtt{+}(\mathtt{-}(\mathtt{x}, \mathtt{y}), \mathtt{z})$ and $\mathtt{-}(\mathtt{x}, \mathtt{+}(\mathtt{y}, \mathtt{z}))$ correspond to the sets

$$\{\mathtt{+}, \mathtt{z1}, \mathtt{-0}, \mathtt{x00}, \mathtt{y10}\}, \qquad \{\mathtt{-}, \mathtt{x0}, \mathtt{+1}, \mathtt{y01}, \mathtt{z11}\}$$

We have remarked above that a tree domain uniquely encodes an ordered tree. This means that a bag uniquely encodes an ordered labelled tree. It follows different bags correspond to different ordered trees. This shows that we have unique readability for bags, despite the fact that the bag is a set and not a sequence. Finally, let Δ be a bag, and let $\Delta = \{\delta_i : i \leq n\}$ be an enumeration of its members. Then the string

$$\delta_1{}^\frown\delta_2{}^\frown \ldots {}^\frown\delta_{n-1}{}^\frown\delta_n$$

is said to be a Δ–**string**.

DEFINITION 4.7
Let Ω be a signature. The **ideal case marking language** over this signature is the set of all Δ–strings where Δ is a bag over Ω. It is denoted by ICM_Ω. The **weak ideal case marking language** over Ω, $WICM_\Omega$ is the set of strings associated with subsets of bags over Ω.

We can also define the bags by a generating system over sets of sequences. A *unit* is a sequence $f^\frown\sigma$, where $f \in F$ and σ is a

[17] The reader will wonder why we need the transpose here. The reason is that the notation for tree domains clashes with natural languages precisely at this point. We write the sequence of a node in a tree domain starting with the top node; however, case sequences are typically suffixes, and so they start with the lowest node.

finite sequence of natural numbers. (We may actually assume that no number of σ is larger than the maximum of the $\Omega(f)$, $f \in F$.) The set of units is denoted by U. Now we shall define sets of units, which correspond to terms. To do that, we shall use an auxiliary symbol $\mathtt{X}$. This symbol will allow us to define incomplete derivations. We shall define S, the set of terms, as follows. For a set M let M^+ be the result of replacing an occurrence of $\mathtt{X}^\frown\sigma$ by $\{f^\frown\sigma\} \cup \{\mathtt{X}^\frown i^\frown\sigma : i < \Omega(f)\}$. Let S be the smallest set containing $\{\mathtt{X}\}$ and which is closed under the transition from M to M^+. We call S the set of *partial terms*. A *bag* is a set in S which does not contain any occurrence of $\mathtt{X}$.

The first bag is generated as follows.

$$\begin{array}{lll} \{\mathtt{X}\}, & \{\mathtt{+}, \mathtt{X0}, \mathtt{X1}\}, & \{\mathtt{+}, \mathtt{-0}, \mathtt{X00}, \mathtt{X01}, \mathtt{X1}\}, \\ \{\mathtt{+}, \mathtt{-0}, \mathtt{x00}, \mathtt{X10}, \mathtt{X1}\}, & \{\mathtt{+}, \mathtt{-0}, \mathtt{x00}, \mathtt{y10}, \mathtt{X1}\}, & \{\mathtt{+}, \mathtt{-0}, \mathtt{x00}, \mathtt{y10}, \mathtt{z1}\} \end{array}$$

The following can be shown.

THEOREM 4.8 (Ebert)
Let Ω be a signature. The languages ICM_Ω and $WICM_\Omega$ are uniquely readable.

Further results will be established below. They will show that these two languages are indeed very natural and can be recognized faster than context–free languages.

5 A Compositional Semantics: A Worked Example

We are now going to outline a semantics for word marking languages. The basic principle is rather simple: variables will be identified with sequences of case–functions. This is just a naming convention. [18] The semantics uses two levels: a DRS–level,

[18]This approach is actually not far fetched. In a run–of–the–mill logical language one usually assumes that variables have the form x_i, where i is a natural number. But this defines an infinitary language to start with, since each variable is a distinct symbol. In fact, if one manipulates these variables in a computer, one will end up coding variables as sequences of the form $\mathtt{x}\alpha$, where α is a sequence of zeros and ones. This allows the computer's inbuilt arithmetical functions to perform substitutions, which must

which contains DRSs, and a referent level, which talks about the names of the referents used by the DRS. Recall that a DRS has the form $[V : \Delta]$, where V is a set of variables and Δ a set of formulae or DRSs. The meaning of V will not be of importance throughout this paper. When constituents are put together to form a new constituent, the corresponding semantics is computed by means of a (binary) operation, called *merge*. [19] This is the terminology of Vermeulen [25], see also Vermeulen and Visser [26]. Merge is here a semantic operation in contrast to the merge operation defined in generative grammar.

There is one additional symbol: $\circ$. It is a variable over names of referents. A simple lexical entry, for example for the verb to teach looks — depending on semantics analysis — like this:

$$
\begin{array}{c}
/\mathsf{teach}/ \\
\begin{array}{|c|}\hline \circ : \circ \\ \hline \end{array} \\
\begin{array}{|c|}\hline \varnothing \\ \hline \mathsf{teach}'(\circ); \\ \mathsf{act}'(\circ) \doteq \textsc{nom}^\frown\circ; \\ \mathsf{thm}'(\circ) \doteq \textsc{acc}^\frown\circ. \\ \hline \end{array}
\end{array}
\qquad
\begin{array}{c}
/\mathsf{teach}/ \\
\begin{array}{|c|}\hline \circ : \circ \\ \hline \end{array} \\
\begin{array}{|c|}\hline \varnothing \\ \hline \mathsf{teach}'(\circ), \textsc{nom}^\frown\circ). \\ \hline \end{array}
\end{array}
$$

Here, the upper part is the referent system, and the lower part an ordinary DRS, with a head section containing a set of variables and a body section containing a set of clauses. This means that there is an event of teaching whose actor is some x and whose theme or goal is y. However, it is not x and y that appear here, as is usual. Rather, instead of x we find the variable $\textsc{nom}^\frown\circ$, and instead of y we find $\textsc{acc}^\frown\circ$. So, x and y are metavariables, and the actual variables are sequences of case markers followed by $\circ$. $\circ$ can be instantiated to any sequence of case functions. Now the meaning

be explicitly defined. So, if one wants to implement Montague Semantics, one will end up programming the substitutions, for which λ–calculus was originally responsible. However, in the semantics we are going to propose here, this is already done. The reader is made aware of the fact that this semantics needs no explication in terms of λ–calculus, since it is already at a lower level. The substitutions, being defined as string substitutions (which in turn are easily implemented), are computable in linear time using a regular transducer.

[19] Syntactic constituents need not be binary. This creates a certain tension between syntax and semantics, about which we will talk later.

of $\circ : \circ$ is as follows: it says that nothing is added to the sequence under merge, $\circ$ is simply replaced by $\circ$. This will become clear in a minute.

We shall assume that the phrase a doctor in the nominative has the following semantics:

$$\begin{array}{c} /\mathsf{a\ doctor}/ \\ \boxed{\circ : \circ} \\ \begin{array}{|c|} \hline \{x\} \\ \hline \mathsf{doctor}'(\textsc{nom}^\frown\circ) \\ \hline \end{array} \end{array}$$

The two structures for /teach/ and /a doctor/ can merge by identifying $\circ$. This means that they now unify $\circ$ to the same sequence. We will then get

$$\begin{array}{c} /\mathsf{a\ doctor\ teach}/ \\ \boxed{\circ : \circ} \\ \begin{array}{|c|} \hline \varnothing \\ \hline \mathsf{teach}'(\circ); \\ \mathsf{act}'(\circ) \doteq \textsc{nom}^\frown\circ; \\ \mathsf{thm}'(\circ) \doteq \textsc{acc}^\frown\circ; \\ \mathsf{doctor}'(\textsc{nom}^\frown\circ). \\ \hline \end{array} \end{array}$$

If done in this way, we can merge any two constituents and get any kind of semantics. However, the outcome is constrained in just the right way as we shall see.

Now, a case suffix has the habit of adding something to the stack of cases. We shall assume that in addition to any semantic effect that it might have it contributes to the name of the variables in the following way. Their names are prolonged by one function. To see how this works, let us write down the semantics of, say, nominative:

$$\begin{array}{c} /\textsc{nom}/ \\ \boxed{\circ : \textsc{nom}^\frown\circ} \\ \begin{array}{|c|} \hline \varnothing \\ \hline \varnothing \\ \hline \end{array} \end{array}$$

Here we find the statement $\circ : \textsc{nom}^\frown\circ$. This says that when the above structure is merged with another one, say $\Delta(\circ)$, then the variable $\circ$ of Δ is instantiated to (or replaced by) $\textsc{nom}^\frown\circ$. This means that the entire sequence of case suffixes is increased by one element. As a result of the merge we shall get however $\circ : \circ$. In order that this is well defined, Δ need not contain exactly $\circ : \circ$. If it contains, say, $\circ : \textsc{acc}^\frown\circ$, then we only have to stipulate how the resulting sequence will be, ie whether we get $\circ : \textsc{nom}^\frown\textsc{acc}^\frown\circ$ or whether we get $\circ : \textsc{acc}^\frown\textsc{nom}^\frown\circ$. The choice is made so that merge is associative. However, throughout this paper we shall make the assumption that merge is successful only if one of the semantic structures has an upper line of the form $\circ : \circ$. Such a structure we shall also call *plain*. Here is now an official definition.

Definition 5.1
A **variable** over F is a sequence $x = \sigma^\frown\circ$, where σ is a sequence of case markers. If y is a variable, then y^x denotes the result of replacing in y the symbol $\circ$ by x. For a set Δ of formulae, $\Delta^x := \{\delta^x : \delta \in \Delta\}$.

Definition 5.2
A **semantic structure** is a pair $A = [\circ : x, [V : \Delta]]$, where x is a variable and $[V : \Delta]$ a DRS, that is, V is a set of variables and Δ a set of formulae. A is **plain** if $x = \circ$. Let $A = [\circ : x, [V : \Delta]]$ and $B = [\circ : y, [V' : \Delta']]$. The **merge** of A and B, denoted by $A \oplus B$ is defined if and only if either A or B is plain.

- If A is plain, $A \oplus B := [\circ : \circ, [V^y \cup V' : \Delta^y \cup \Delta']]$.
- If B is plain, $A \oplus B := [\circ : \circ, [V \cup V'^x : \Delta \cup \Delta'^y]]$.

To see the mechanism work, let us start with the following lexical entry for a simple noun:

/doctor/
$\circ : \circ$
$\varnothing$
$\mathsf{doctor}'(\circ)$

Now we shall compute the merge with the semantic structure for nominative:

$$\begin{array}{c}/\mathsf{doctor}/\\ \begin{array}{|c|}\hline \circ : \circ\\ \hline \varnothing\\ \hline \mathsf{doctor}'(\circ)\\ \hline\end{array}\end{array} \oplus \begin{array}{c}/\textsc{nom}/\\ \begin{array}{|c|}\hline \circ : \textsc{nom}^\frown\circ\\ \hline \varnothing\\ \hline \varnothing\\ \hline\end{array}\end{array} = \begin{array}{c}/\mathsf{doctor} + \textsc{nom}/\\ \begin{array}{|c|}\hline \circ : \circ\\ \hline \varnothing\\ \hline \mathsf{doctor}'(\textsc{nom}^\frown\circ)\\ \hline\end{array}\end{array}$$

To see why this is so, notice that the symbol $\circ$ in the first structure is replaced by $\textsc{nom}^\frown\circ$ according to the laws of merge. Now, in order to understand the potential of this proposal let us repeat this example with a relational noun, teacher:

$$\begin{array}{c}/\mathsf{teacher}/\\ \begin{array}{|c|}\hline \circ : \circ\\ \hline \varnothing\\ \hline \mathsf{teach}'(\circ, \textsc{gen}^\frown\circ)\\ \hline\end{array}\end{array}$$

If we add the suffix NOM, we get

$$\begin{array}{c}/\mathsf{teacher} + \textsc{nom}/\\ \begin{array}{|c|}\hline \circ : \circ\\ \hline \varnothing\\ \hline \mathsf{teach}'(\textsc{nom}^\frown\circ, \textsc{gen}^\frown\textsc{nom}^\frown\circ)\\ \hline\end{array}\end{array}$$

Notice that by the mechanics of replacement, it is not only the main variable that changes its name but also the variable of the complement. This is what we will make use of.

Cases may or may not have a semantics. This actually does not make much of a difference for this calculus. Take the genitive, which in many languages is used for marking possession:

$$\begin{array}{c}/\textsc{gen}/\\ \begin{array}{|c|}\hline \circ : \textsc{gen}^\frown\circ\\ \hline \varnothing\\ \hline \mathsf{belong\text{-}to}'(\circ, \textsc{gen}^\frown\circ)\\ \hline\end{array}\end{array}$$

So, when a genitive is attached, it says that the thing to which it attaches owns something. Here, $\circ$ represents the thing that is possessed, while $\mathrm{GEN}^\frown\circ$ is the thing that owns it. To see how this works, we shall turn to a real example. The following is a construction of Old–Georgian, taken from [3], Page 103.

(5.1) sarel-ita   man-isa-jta
name-INST   father-GEN-INST
with father's name

The first part is clear: sarel-ita is

$$(A) \qquad \begin{array}{c} /\mathsf{sarel\text{-}ita}/ \\ \hline \multicolumn{1}{|c|}{\circ : \circ} \\ \hline \multicolumn{1}{|c|}{\varnothing} \\ \hline \multicolumn{1}{|c|}{\mathsf{name}'(\mathrm{INST}^\frown\circ);} \\ \multicolumn{1}{|c|}{\mathsf{instr}'(\mathrm{INST}^\frown\circ).} \\ \hline \end{array}$$

Here, $\mathsf{inst}'(x)$ means that x is an instrument. Actually, it would be more accurate to use a binary predicate $\mathsf{inst}'(x, e)$, with the intended interpretation that x is used as an instrument in the event x. We shall generously ignore this detail, however. Now let us turn to man-isa-jta. First we attach the genitive to man:

$$\begin{array}{c} /\mathsf{man}/ \\ \hline \multicolumn{1}{|c|}{\circ : \circ} \\ \hline \multicolumn{1}{|c|}{\varnothing} \\ \hline \multicolumn{1}{|c|}{\mathsf{father}'(\circ, \mathrm{GEN}^\frown\circ)} \\ \hline \end{array} \oplus \begin{array}{c} /\mathsf{isa}/ \\ \hline \multicolumn{1}{|c|}{\circ : \mathrm{GEN}^\frown\circ} \\ \hline \multicolumn{1}{|c|}{\varnothing} \\ \hline \multicolumn{1}{|c|}{\mathsf{belong\text{-}to}'(\circ, \mathrm{GEN}^\frown\circ)} \\ \hline \end{array}$$

$$= \begin{array}{c} /\mathsf{man\text{-}isa}/ \\ \hline \multicolumn{1}{|c|}{\circ : \circ} \\ \hline \multicolumn{1}{|c|}{\varnothing} \\ \hline \multicolumn{1}{|c|}{\mathsf{father}'(\mathrm{GEN}^\frown\circ, \mathrm{GEN}^\frown\mathrm{GEN}^\frown\circ);} \\ \multicolumn{1}{|c|}{\mathsf{belong\text{-}to}'(\circ, \mathrm{GEN}^\frown\circ).} \\ \hline \end{array}$$

Next we attach the instrumental suffix:

(B)

/man-isa-jta/

$\circ : \circ$
$\varnothing$
$\mathsf{father}'(\textsc{gen}^\frown\textsc{inst}^\frown\circ, \textsc{gen}^\frown\textsc{gen}^\frown\textsc{inst}^\frown\circ)$; $\mathsf{belong\text{-}to}'(\textsc{inst}^\frown\circ, \textsc{gen}^\frown\textsc{inst}^\frown\circ)$.

Finally, the two structures (A) and (B) are merged to derive the final representation (C).

(C)

/sarel-ita man-isa-jta/

$\circ : \circ$
$\varnothing$
$\mathsf{name}'(\textsc{inst}^\frown\circ)$; $\mathsf{father}'(\textsc{gen}^\frown\textsc{inst}^\frown\circ, \textsc{gen}^\frown\textsc{gen}^\frown\textsc{inst}^\frown\circ)$; $\mathsf{instr}'(\textsc{inst}^\frown\circ)$; $\mathsf{belong\text{-}to}'(\textsc{inst}^\frown\circ, \textsc{gen}^\frown\textsc{inst}^\frown\circ)$.

(C) is true in a model under an assignment for the variables if $\textsc{inst}^\frown\circ$ is instantiated to a thing x that is a name, and $\textsc{gen}^\frown\textsc{inst}^\frown\circ$ is instantiated to a thing y that is a father, and x belongs to y. And that y is a father of the value of $\textsc{gen}^\frown\textsc{gen}^\frown\textsc{inst}^\frown\circ$. This is exactly as it should be.

6 Semantics for the Ideal Case Marking Language

The semantics outlined in the previous section can be used to give a compositional account of the semantics of the ideal case marking language. We shall assume that the function symbol `f` is of arity $\Omega(\mathtt{f})$ and has meaning f. We shall also assume that we have symbols `0`, `1` etc. The lexical entry for `f` is therefore the following:

`f`

$\circ : \circ$
$\varnothing$
$\circ \doteq \mathsf{f}(\mathsf{0}^\frown\circ, \mathsf{1}^\frown\circ, \ldots, \Omega(f) - 1^\frown\circ)$

System variables like x and y are treated just like 0–ary function symbols:

$$\begin{array}{c} \texttt{x} \\ \begin{array}{|c|}\hline \circ : \circ \\ \hline \end{array} \\ \begin{array}{|c|}\hline \{\circ\} \\ \hline \circ \doteq \texttt{x} \\ \hline \end{array} \end{array}$$

(The reader may wonder why the variables come with a nonempty set V while all other functions have an empty set in their DRS. This is however just for purpose of illustration and of no further significance.) The elements 0, 1 have the following semantics:

$$\begin{array}{c} \texttt{0} \\ \begin{array}{|c|}\hline \circ : \texttt{0}^\frown\circ \\ \hline \end{array} \\ \begin{array}{|c|}\hline \varnothing \\ \hline \varnothing \\ \hline \end{array} \end{array}$$

There are only two conventions:

1. 0, 1 etc may only be suffixes.
2. 0, 1 etc may only be attached to simple expressions.

Here a simple expression is one that contains only one function or variable symbol. A simple expression is one that corresponds to a single branch of the tree domain. We shall understand that this is a syntactic restriction that is not due to any semantics. We shall return to the implications of these conventions below.

To see that this works as intended we shall reproduce an earlier example. Take the arithmetical term +(-(x, y), z). It is represented by the following string:

$$\texttt{y10+-0x00z1}$$

By the conventions, this must be parsed in the following way:

$$\texttt{((y1)0)(+)(-0)((x0)0)(z1)}$$

Among the terms enclosed in brackets, one must go from left to right otherwise the merge is undefined. For all other terms, the direction is completely irrelevant.

We start from the left end. We compose `y` and `1`.

$$\begin{array}{c}
\mathtt{y1} \\
\begin{array}{|c|} \hline \circ : \circ \\ \hline \end{array} \\
\begin{array}{|c|} \hline \{\mathtt{1}^\frown\circ\} \\ \hline \mathtt{1}^\frown\circ \doteq \mathtt{y} \\ \hline \end{array}
\end{array}$$

Next we compose with `0`:

$$\begin{array}{c}
\mathtt{y10} \\
\begin{array}{|c|} \hline \circ : \circ \\ \hline \end{array} \\
\begin{array}{|c|} \hline \{\mathtt{10}^\frown\circ\} \\ \hline \mathtt{10}^\frown\circ \doteq \mathtt{y} \\ \hline \end{array}
\end{array}$$

We compose with `+`:

$$\begin{array}{c}
\mathtt{y10+} \\
\begin{array}{|c|} \hline \circ : \varepsilon \\ \hline \end{array} \\
\begin{array}{|c|} \hline \{\mathtt{10}^\frown\circ\} \\ \hline \mathtt{10}^\frown\circ \doteq \mathtt{y}; \\ \circ \doteq \mathtt{+}(\mathtt{0}^\frown\circ, \mathtt{1}^\frown\circ). \\ \hline \end{array}
\end{array}$$

Next we compose `-` and `0` and get

$$\begin{array}{c}
\mathtt{-0} \\
\begin{array}{|c|} \hline \circ : \circ \\ \hline \end{array} \\
\begin{array}{|c|} \hline \varnothing \\ \hline \mathtt{0}^\frown\circ \doteq \mathtt{-}(\mathtt{00}^\frown\circ, \mathtt{10}^\frown\circ) \\ \hline \end{array}
\end{array}$$

Together with the last this gives

$$\begin{array}{c}
\mathtt{y10+-0} \\
\begin{array}{|c|} \hline \circ : \circ \\ \hline \end{array} \\
\begin{array}{|c|} \hline \{\mathtt{10}^\frown\circ\} \\ \hline \mathtt{10}^\frown\circ \doteq \mathtt{y}; \\ \circ \doteq \mathtt{+}(\mathtt{0}^\frown\circ, \mathtt{1}^\frown\circ); \\ \mathtt{0}^\frown\circ \doteq \mathtt{-}(\mathtt{00}^\frown\circ, \mathtt{10}^\frown\circ). \\ \hline \end{array}
\end{array}$$

The term $\texttt{x00}$ is analogous to $\texttt{y10}$ and $\texttt{z1}$ to $\texttt{y0}$:

$$\begin{array}{cc}
\texttt{x00} & \texttt{z1} \\
\begin{array}{|c|}\hline \circ : \circ \\ \hline \{\texttt{00}^\frown\circ\} \\ \hline \texttt{00}^\frown\circ \doteq \texttt{x} \\ \hline \end{array}
&
\begin{array}{|c|}\hline \circ : \circ \\ \hline \{\texttt{1}^\frown\circ\} \\ \hline \texttt{1}^\frown\circ \doteq \texttt{z} \\ \hline \end{array}
\end{array}$$

Composing them with the above result we get:

$$\begin{array}{c}
\texttt{y10+-0x00z1} \\
\begin{array}{|c|}\hline \circ : \circ \\ \hline \{\texttt{10}^\frown\circ, \texttt{00}^\frown\circ, \texttt{1}^\frown\circ\} \\ \hline
\texttt{10}^\frown\circ \doteq \texttt{y}; \\
\circ \doteq \texttt{+}(\texttt{0}^\frown\circ, \texttt{1}^\frown\circ); \\
\texttt{0}^\frown\circ \doteq \texttt{-}(\texttt{00}^\frown\circ, \texttt{10}^\frown\circ); \\
\texttt{00}^\frown\circ \doteq \texttt{x}; \\
\texttt{1}^\frown\circ \doteq \texttt{z}. \\ \hline
\end{array}
\end{array}$$

We shall verify that the value of $\circ$ is actually the same as the value of $\texttt{+}(\texttt{-}(\texttt{x}, \texttt{y}), \texttt{z})$. This can be shown by reducing the system of equations. Notice first of all that in the body of the DRS there is an equation saying that $\texttt{00}^\frown\circ$ has the same value as $\texttt{x}$, $\texttt{10}^\frown\circ$ the same value as $\texttt{y}$ and $\texttt{1}^\frown\circ$ the same value as $\texttt{z}$. We may therefore reduce the body of the last structure to

$$\begin{array}{c}
\circ \doteq +(\texttt{0}^\frown\circ, \texttt{z}) \\
\texttt{0}^\frown\circ \doteq \texttt{-}(\texttt{x}, \texttt{y})
\end{array}$$

We may finally replace $\texttt{0}^\frown\circ$ by $\texttt{-}(\texttt{x}, \texttt{y})$ in the first line. We then get

$$\circ \doteq \texttt{+}(\texttt{-}(\texttt{x}, \texttt{y}), \texttt{z})$$

Notice that the merge is always defined, so that any binary parse of the string yields a string–to–meaning translation. However, not all of them can be correct. A specific example and a thorough discussion of this will follow in Section 8. So, we need a characterization of those strings that yield a proper translation. Call a sequence of case markers a *register*. Call a string of the form $f^\frown\sigma$ a *block*, if $f \in F$ and σ a register. Any string over Ω^γ can be naturally decomposed into a sequence of blocks. The grammar that generates a parse is the following. (Here, $\mathtt{S}$ stands for *string*, $\mathtt{B}$ for *block*, $\mathtt{C}$ for *case*, $\mathtt{F}$ for *function*, and $\mathtt{R}$ for *register*. $|$ is the usual disjunction sign for rules.)

$$\begin{array}{lcll}
\mathtt{S} & \to & \mathtt{B} \mid \mathtt{SB} & \\
\mathtt{B} & \to & \mathtt{FR} & \\
\mathtt{R} & \to & \varepsilon \mid \mathtt{CR} & \\
\mathtt{F} & \to & f & (f \in F) \\
\mathtt{C} & \to & \mathtt{c}_i & (i < \mu)
\end{array}$$

THEOREM 6.1
A string $\vec{x}$ over Ω^γ is in ICM_Ω iff

1. No two blocks of $\vec{x}$ have the same register.
2. If $\sigma^\frown\mathtt{c}_k$ is a register of $\vec{x}$, then so is σ.
3. If $f^\frown\sigma$ is a block of $\vec{x}$, then
   (a) there is no register of the form $\sigma^\frown\mathtt{c}_k$ where $k \geq \Omega(f)$,
   (b) for all $k < \Omega(f)$ there is a register of the form $\sigma^\frown\mathtt{c}_k$.

The proof is rather straightforward and omitted. Likewise, the weak ideal case marking language is characterized by omitting the existential clauses of the previous definition:

THEOREM 6.2
A string $\vec{x}$ over Ω^γ is in $WICM_\Omega$ iff

1. No two blocks of $\vec{x}$ have the same register.
2. If $f^\frown\sigma$ is a block of $\vec{x}$, then there is no register of the form $\sigma^\frown\mathtt{c}_k$ where $k \geq \Omega(f)$.

We shall use these results to establish some complexity results about these languages.

DEFINITION 6.3
Let $A = \{a_i : i < n\}$ be a finite set and $\mathbb{N}^n$ the monoid of n–tuples of natural numbers consisting of the vector $\vec{0}$ (consisting of n consecutive zeros) and addition as its operations. Let $\vec{e}_i := \langle \delta^i_j :$

$j < n\rangle$ where $\delta^i_j := 1$ iff $i = j$ and $\delta^i_j := 0$ otherwise. Define $\psi : A^* \to \mathbb{N}^n$ inductively as follows.

$$\begin{array}{lcl} \psi(\varepsilon) & := & \vec{0} \\ \psi(\vec{x} \cdot a_i) & := & \psi(\vec{x}) + \vec{e}_i \end{array}$$

Call $S \subseteq A^*$ **linear** if there is a number λ and $\vec{u}_i \in \mathbb{N}^n$, $1 \leq i \leq \lambda$, such that

$$\psi[S] = \{\vec{u}_0 + \sum_{i=1}^{\lambda} \mu_i \vec{u}_i : \mu_i \in \mathbb{N}, 1 \leq i \leq \lambda\}$$

Finally, call S **semilinear** if it is a finite union of linear sets.

By a theorem of Parikh, context–free languages are semilinear (see [9] for a proof). It can be shown that also linear indexed languages and MCTALs (= Multiple Component Tree Adjoining Grammars) as well as multiply context–free languages are semilinear. However, the following is shown in [6], building on results of [16].

THEOREM 6.4 (Ebert)
ICM$_\Omega$ and *WICM*$_\Omega$ are not semilinear, unless they are finite.

This does not mean, however, that these languages are complex. First, they can be parsed using a Turing machine with linearly bounded space. Hence, they are context–sensitive. Moreover, the parsing complexity is even polynomial.

THEOREM 6.5 (Ebert)
Membership in *ICM*$_\Omega$ and *WICM*$_\Omega$ can be decided in $O(n^{3/2} \log_2 n)$ time on a deterministic multihead Turing machine with linearly bounded space, or in $O(n^{5/2} \log_2 n)$ time on a single head Turing machine with linearly bounded space.

Context–free languages can be parsed in $O(n^{\log_2 7})$ steps on a multi–head Turing machine. The exponent is larger than 2.7, so the word marking languages are faster to recognize than many context–free languages. Polish Notation is however still faster. It is recognizable in linear time on a deterministic, linearly space bounded two–head Turing machine.

The strategy of proof is to first order the blocks lexicographically (this takes $O(n^{3/2} \cdot \log_2 n)$ time, using efficient sorting techniques, for example Merge Sort). Compliance with the conditions is then verified in linear time. Another strategy is as follows. It takes time linear in the input to construct a semantic representation on the basis of a parse using the above grammar. Since this grammar is deterministic, the translation is done in overall linear time. The conditions can be checked on the semantics as well. For ICM_Ω there is even a completely semantic procedure. Namely, a string is in ICM_Ω iff its semantics can be reduced to a single term using a step by step elimination of equations. This gives another way of checking membership in ICM_Ω. The two methods are of the same complexity. Notice also that the merge operation in this semantics is of very low complexity: it is linear in the length of the input structures, if the case stacks are implemented as pointer structures. The set of all case stacks forms a tree rooted at $\circ$. Substitution is nothing but putting one more cell at the root of the tree.

7 A Comparison

We have proposed in the previous sections two alternative ideal languages: (Reverse) Polish Notation and the Ideal Case Marking Language. We have presented them both in their pure abstract instantiation. Both enjoy unique readability and with each we can associate a formal semantics. While Polish Notation lends itself easily to a functional interpretation, the case marking languages are better treated using a relational interpretation. [20] In this section we will discuss in some depth the similarities and differences between the notations.

[20] Without going into much detail, we shall explain the basis of this remark. It is known that in predicate logic, a theory formulated in an arbitrary signature can be replaced by a theory that uses only relations. Namely, each n–ary function symbol f is replaced by an $n+1$–ary relation symbol R. An equation $f(x_0, \ldots, x_{n-1}) \doteq y$ is replaced by the formula $R(x_0, \ldots, x_{n-1}, y)$. Terms are eliminated in a stepwise construction. As it turns out, the semantics we are proposing here for word marking languages resembles exactly the result of turning a functional signature (that of Montague Grammar) into a relational one.

A first difference that is worth noting is that Polish Notation neither allows for word order variation nor for argument elision. While this does not seem to be problematic for computer languages, natural languages are different in this respect. Speakers like to drop all material in a sentence that is given by the context. Although languages resist this (you cannot simply omit what you want) there is generally some freedom. Specifically, those languages with rich case marking do allow for argument elision. Australian languages are generally very liberal in this respect. In Martuthunira one is allowed to drop any constituent or word whatsoever (see Section 9, and [4]).

Now we are in a certain dilemma: we have shown that case marking languages allow for free word order and free omission, but why is it that rich case marking does not lead to random word order? There are two kinds of answers. The first answer is historical: if a language starts with poor morphology, it has strict word order. If it acquires richer morphology, there is no need to free the word order in tandem. Languages do not generally strive for word order freedom (see Lehmann [13] on this point). But here again there is the question why this should be so. This brings us to the second answer: the order of thought. If a language is free in its word order this practically means that speakers can use a word when they need or want to. That in turn means that they can map the order of thought almost directly onto the linear order. If that is so, it is unlikely, for example, that NPs or subordinate sentences become discontinuous just because there is the possibility for that. However, omission of argument is much different. We just choose not to say something, that's all. There is no need to adapt the mapping from thought to speech in any complicated way.

It has been suggested to me by some referee that lack of free word order is usually due to topic focus articulation or other (mainly called pragmatic) factors. This is a statement that can be found in the description of languages with free word order. A language with free word order is claimed to have restricted word order since word order is constrained by pragmatic principles. However, this simply rests on a terminological mismatch in what is to count as free word order. First of all, languages vary on the degree of pragmatic fixation of word order. While Hungarian and Finnish clearly show pragmatically constrained word order this does not seem to be the case at all with many Australian languages (Alan Dench, p. c.). Now, we shall have to dstinguish clearly two types of languages.

1. Languages where you simply cannot have different word orders at all, holding constant the functor–argument relations. (This is effectively what we have called the positional regime.)
2. Languages where you can have different word orders, holding constant the functor–argument relations.

Now, if we have a language of the second kind, it is another matter if the different word orders are made to signal additional meanings. So there are the following subtypes.

2.a. Languages where we do not have different word orders, holding constant the functor–argument relations as well as the (semantic/pragmatic) meaning.

2.b. Languages where we do have different word orders, holding constant the functor–argument relations as well as the (semantic/pragmatic) meaning.

Thus, while we speak of languages of Type 2 as free word order languages, others prefer to reserve this for Type 2.b languages. We stick with our terminology here. Let us call therefore Type 2.a *semantically (pragmatically) constrained*. Throughout this paper we have been solely concerned with the free/non–free word order dichotomy, and not with the constrained/unconstrained dichotomy, and it is certainly wise to keep these two apart so as to be able to control for the kinds of information that word order can give. It can be syntactic (showing us the argument status of words) or it can indeed be semantic or pragmatic. If it is syntactic, we are in Type 1, otherwise in Type 2. If it is semantic/pragmatic, we are in Type 2.a, otherwise in Type 2.b. (Again, mixtures occur more often than not.)

Both of these models have wider application. In fact, languages do admit ambiguity and the most favoured language in absence of morphology is SVO (English). The ambiguities that formally arise are actually now welcome because they reflect a basic trait of the language under analysis. In this vein, the analysis can be extended to languages of type SVO, OVS, SOV and VOS, and if one allows the verb to form a constituent with the subject, also to the remaining types OSV and VSO. (See [10] for arguments that in such languages the verb forms a constituent with its subject.)

The semantics for case marking languages works also for other types of languages. In particular, it can be used for languages that are consistently peripherally group marking. In these languages, the case marker is put either in front or at the end of the entire phrase. If we want to do this, nothing needs to be changed, except that the case marker will now be a phrasal affix rather than a word affix. To illustrate this, we shall compute the semantics of the phrase 'good doctor'.

/NOM/	/good/	/doctor/
$\circ : \text{NOM}^\frown\circ$	$\circ : \circ$	$\circ : \circ$
$\varnothing$	$\varnothing$	$\varnothing$
$\varnothing$	$\mathsf{good}'(\circ)$	$\mathsf{doctor}'(\circ)$

We may either compose both good and doctor with the nominative and then compose the result, or we may first compose good and doctor and then compose with the nominative. In both cases the result is the same:

/good doctor/
$\circ : \circ$
$\varnothing$
$\mathsf{good}'(\text{NOM}^\frown\circ)$; $\mathsf{doctor}'(\text{NOM}^\frown\circ)$.

We shall also prove this in a rather formal manner:

Proposition 7.1

Let A and B be plain, and let C be a case marker. Then

$$(A \oplus B) \oplus C = (A \oplus C) \oplus (B \oplus C)\ .$$

Proof. Let $A = [\circ : \circ, [V : \Delta]]$, $B = [\circ : \circ, [V' : \Delta']]$ and $C = [\circ : x, [V'' : \Delta'']]$. Denote as before by ^{x} the substitution $\circ \mapsto x$. Then $A \oplus B = [\circ : \circ, [V \cup V' : \Delta \cup \Delta']]$ and so $(A \oplus B) \oplus C = [\circ : \circ, [V^x \cup V'^x \cup V'' : \Delta^x \cup \Delta'^x \cup \Delta'']]$. On the other hand, $A \oplus C = [\circ : \circ, [V^x \cup V'' : \Delta^x \cup \Delta'']]$ and $B \oplus C = [\circ : \circ, [V'^x \cup V'' : \Delta'^x \cup \Delta'']]$ and so $(A \oplus C) \oplus (B \oplus C) = [\circ : \circ, [V^x \cup V'^x \cup V'' : \Delta^x \cup \Delta'^x \cup \Delta'']]$. This shows the claim. Q. E. D.

In languages which have phrasal case affixes, these affixes may actually end up in a sequence, giving the appearance of suffixaufnahme (which is the multiple case marking produced by the word marking regime). The following is found in Sumerian (see [21]).

(7.1) é lugal-ak
house king-GEN
house of the king

(7.2) é lugal-ak-a
house king-GEN-LOC
in the house of the king

(7.3) é šeš lugal-ak-ak-a
house brother king-GEN-GEN-LOC
in the house of the brother of the king

Here, the case markers are phrasal suffixes, and the genitive complement also follows the head noun. Hence the last example is to be bracketed as follows.

(é (šeš (lugal)-ak)-ak)-a

This is admittedly a rare example, though we find in the abovementioned source also examples from Late Elamite and Kanuri (a Nilo–Saharan language).

Most languages are in between these types. Latin has case agreement, but no stacking of cases. These languages are intermediate cases, since they are neither entirely word marking (because then they would look like Kayardild or Martuthunira) nor are they group marking. Nevertheless, if the morphological and syntactic restrictions are properly implemented, the semantics works for these examples, too. There is, however, a condition. If for example, we wish to give a semantics for the genitive that parallels that of Old Georgian, we must assume that a genitive specifier agrees in case with the head noun. We must therefore posit an invisible case marker for the specifier. Likewise, we must assume that relative clauses, in fact all clauses, are case marked.

8 Semantic versus Syntactic Restrictions

The model which we have developed is ideal for so–called *flat* languages. It assumes no phrase structure whatsoever and nevertheless gets the semantics right. However, two issues must still be addressed: (1) what does it mean that the syntax has flat structure and (2) are the restrictions on phrase structure semantically or syntactically motivated?

To see that there is a problem, let us start with the words. The words are composed from roots by adding the case suffixes. By the definition of merge we can only attach a suffix to something that is not a suffix. But this restriction is not enough to guarantee Conventions 1 and 2 to hold. For there is nothing that says that cases are suffixes rather than prefixes and nothing tells us that they can only be attached to a word rather than a complex constituent. So, the right result is only obtained if we assume in addition a grammar that allows only the right kind of parses. To see this, let us note that if the cases can alternatively be prefixes as well as suffixes, an expression of the form `x1y` is ambiguous, because it can be parsed alternatively as (`x1`)`y` or as `x`(`1y`). In natural languages this problem does not arise with cases (since they are not separated by a word boundary) but if the language has pre– as well as postpositions a certain amount of ambiguity may accur. Secondly, if cases can be also phrasal affixes we will not generate the right analyses either. For example, notice that the following would be parsed as correct in Old Georgian:

(8.1) sarel manisajta

Moreover, by Proposition 7.1 it gets the same translation as (5.1).

Notice namely that the instrumental can also be added after the constituent sarel manisa (= 'name of the father') has been formed.

This demonstrates the need for regimentation. A context–free grammar defining the correct parses can easily be given. Let `R` denote the type of root, `F` the type of a suffix, `W` the type of a word, and `C` the type of a complex unit. Then we assume the following rules:

$$\begin{array}{lcl} \mathtt{C} & \to & \mathtt{W} \mid \mathtt{C}\ \mathtt{C} \\ \mathtt{W} & \to & \mathtt{R} \mid \mathtt{W}\ \mathtt{F} \end{array}$$

Thus, a complex constituent can be a word, or it can be made from two complex constituents. These rules allow to parse a string of words in any way one likes, as long as the structure is binary branching. A word is either a stem or a word followed by a suffix. That this much regimentation is enough is a consequence of the following

PROPOSITION 8.1

Let A, B and C be plain semantic structures, that is, semantic structures whose referent system consists of $\circ : \circ$ only. Then we have $A \oplus B = B \oplus A$ and $A \oplus (B \oplus C) = (A \oplus B) \oplus C$.

The proof is easy and omitted. (Notice that we get identity, not just semantic equivalence here.) Suffice it to say that if A and B are plain then the merge is the Zeevat–merge:

$$\begin{array}{c} /\mathsf{X}/ \\ \boxed{\circ : \circ} \\ \boxed{V_1} \\ \boxed{\Delta_1} \end{array} \oplus \begin{array}{c} /\mathsf{Y}/ \\ \boxed{\circ : \circ} \\ \boxed{V_2} \\ \boxed{\Delta_2} \end{array} = \begin{array}{c} /\mathsf{X\ Y}/ \\ \boxed{\circ : \circ} \\ \boxed{V_1 \cup V_2} \\ \boxed{\Delta_1 \cup \Delta_2} \end{array}$$

So, given any string of symbols, we must first parse it according to the grammar above and then assign the meanings to the parts and compose them according to the obtained constituent analysis. Now, it is certainly possible to constrain the syntax to be, for example, right branching on the words. So, rather than the above grammar we might propose the following one:

$$\begin{array}{lcl} \mathtt{C} & \to & \mathtt{W} \mid \mathtt{W}\ \mathtt{C} \\ \mathtt{W} & \to & \mathtt{R} \mid \mathtt{W}\ \mathtt{F} \end{array}$$

However, no one would propose such a structure for a language in absence of any evidence even if that would be consistent with the data (and the semantics just given). But why not? Apparently, in absence of anything that would tell us about the constituent structure we assume that it is free. This means however that the right kind of grammar would be as follows:

$$\begin{array}{lcl} \mathtt{C} & \to & \mathtt{W}^+ \\ \mathtt{W} & \to & \mathtt{R} \mid \mathtt{W}\ \mathtt{F} \end{array}$$

The first line is an abbreviation for a series of infinitely many rules. It means that `C` may be replaced by any nonzero number of `W`s. This is actually what has been proposed for some Australian languages (eg Warlpiri [21]). Here, a constituent can be analysed as a string of words of any length. However, the semantics outlined so far does not allow for such an analysis on the semantic level. Rather, it asks for a binary branching structure. To reconcile the semantical analysis with the constituent structure we propose to call a substring a *syntactic constituent* if it forms a unit in any semantical parse that derives the desired meaning. This means that no sequence of words of length > 1 which is not the entire sentence is a constituent since we could always avoid forming that constituent in a semantic parse and nevertheless get the same result. In this way, the syntactic analysis follows from the semantical analysis in virtue of Proposition 8.1.

However, still we must account for the fact that the case suffixes cannot be added to complex constituents, only to words. Up to this point this was treated as a syntactic restriction. (In fact, it is most likely a morphological restriction, but for the present argumentation this difference does not matter.) There is however a way to make that follow from the 'semantics'. First, we shall assume that a lexical item can state explicitly that it wants a complement to one side rather than another. So, rather than writing $\circ : \text{GEN}^\frown\circ$ into the header of the genitive case, we write $\langle \circ : \text{GEN}^\frown\circ : \triangleright\rangle$. (The brackets

[21] The word order for Warlpiri is not completely free, so the proposed grammars are actually slightly different.

are inserted for readability only.) This means that the item in question seeks a complement to its left and will substitute $\circ$ for $\textsc{gen}^\frown\circ$. This will ensure that the cases are always suffixes. Finally, we introduce another symbol, $*$, in place of $\circ$. Its combinatorics is the same as that of $\circ$. It serves mainly as a flag. Now lexical entries come in the form shown to the left rather than that shown to the right.

$$\begin{array}{|c|}\hline * : * \\ \hline V \\ \hline \Delta \\ \hline \end{array} \qquad\qquad \begin{array}{|c|}\hline \circ : \circ \\ \hline V \\ \hline \Delta \\ \hline \end{array}$$

We shall agree that composition is only defined if the two referent systems have the form $* : *$, or both $\circ : \circ$, or if one has a statement of the form $* : \alpha^\frown\circ$ ($\circ : \alpha^\frown *$) and the other of the form $* : *$ ($\circ : \circ$). This serves to make the merge well–defined. Case suffixes have the form

$$\begin{array}{c} /\textsc{gen}/ \\ \begin{array}{|c|}\hline \langle * : \mathrm{GEN}^\frown * : \rhd\rangle \\ \hline \varnothing \\ \hline \varnothing \\ \hline \end{array}\end{array}$$

Finally, we shall assume that the word boundary marker (#) has the following semantics:

$$\begin{array}{c} /\#/ \\ \begin{array}{|c|}\hline \langle * : \circ : \rhd\rangle \\ \hline \varnothing \\ \hline \varnothing \\ \hline \end{array}\end{array}$$

The idea behind this proposal is quite simple. As long as we keep stacking case suffixes, we are still at the level of a word. This is expressed by the fact that the stack variable has the form $*$. The word is completed when the word boundary marker is added to the right. Then the stack variable changes from $*$ to $\circ$ and allows the word to compose with other full words, but not with uncompleted words.

9 The Sentence Juncture

A text is a sequence of sentences. Texts are coherent, which shows up in the possibility of referring back to previously mentioned objects, in ellipsis, to name a few. In Australian languages of the kind considered here any material that can be inferred from the context can be omitted. This leads to structures that can from a traditional perspective hardly be considered sentences. Here, however, any sequence of words is a semantic constituent (though not a syntactic one as defined above). Hence there is no difficulty in parsing such utterances, and hence no reason not to call them sentences (see [4] for a defense). Notice that there is no need here to posit empty categories. It might therefore be deemed possible to propose the same semantics for the sentential juncture as we did sentence internally. Given that, the semantics would be as follows:

$$\begin{array}{|c|} \multicolumn{1}{c}{/.\!/} \\ \hline \langle \circ : \circ : \triangleright \rangle \\ \hline \varnothing \\ \hline \varnothing \\ \hline \end{array}$$

Yet, such a proposal is easily seen to be insufficient. It would predict that any main level nominative denotes the same object, so whenever in a sentence a nominal appears bearing nominative case, it will be associated with just the same individual as before. To exclude that, the following alternative might be proposed:

$$\begin{array}{|c|} \multicolumn{1}{c}{/.\!/} \\ \hline \langle \circ : \mathtt{1}^\frown\circ : \triangleright \rangle \\ \hline \varnothing \\ \hline \varnothing \\ \hline \end{array}$$

Here, $\mathtt{1}$ is a formal element that is attached to the sequence so as to make the set of variables of the first sentence disjoint from the set of variables of the second sentence. This certainly works, but we again have to impose restrictions. It is obvious that the juncture may not be applied to an incomplete sentence. Otherwise the result will be incorrect.

Now, we can follow the line of Section 8 and postulate a distinct stack symbol • with which we introduce the word/sentence distinction into the semantics. The rest is then rather straightforward. There are two types of sentence junctures, one called ';', which joins the material into the previous sentence, and the other called '.':

$$\begin{array}{c} /;/ \\ \hline \multicolumn{1}{|c|}{\langle \circ : \bullet : \triangleright \rangle} \\ \hline \multicolumn{1}{|c|}{\varnothing} \\ \hline \multicolumn{1}{|c|}{\varnothing} \\ \hline \end{array} \qquad \begin{array}{c} /./ \\ \hline \multicolumn{1}{|c|}{\langle \circ : 1^\frown\bullet : \triangleright \rangle} \\ \hline \multicolumn{1}{|c|}{\varnothing} \\ \hline \multicolumn{1}{|c|}{\varnothing} \\ \hline \end{array}$$

We shall illustrate the potential of the proposal by giving an analysis of a dialogue in Martuthunira, taken from [4]:

(9.1) A: Ngayu kangku-lha mayiili-marnu-ngu kulhampa-arta.
A: I take-PAST SoSo+1Poss-GRP-ACC fish-ALL
I took a group of my grandchildren for fish.

(9.2) B: Nganangu-ngara pawulu-ngara?
B: whoGEN-PL child-PL?
Whose children are they?

(9.3) A: Ngurnu-ngara-a yaan-wirriwa-wura-a.
A: thatOBL-PL-ACC spouse-PRIV-BELONG-ACC
(I took) the ones who belong to the one who is without a spouse.

(9.4) B: Ngaa, purrkuru pala. Ngarraya-ngu-ngara-a.
B: Yes Okay IT niece-GEN-PL-ACC
Yes. Okay that's it (I understand). (You took) niece's ones.

The interesting thing about this dialogue is that the utterances that follow the first one are highly elliptical and can only be understood in their context. Yet, in the semantics we have presented there is actually no need to assume that material is missing in them, and they can be interpreted as such, without any context, although of course the meaning of the utterances appears a little less fragmented when the context is given. We shall go through the dialogue assuming that sentences are conjoined by ';', even across speakers. We shall not concern ourselves with indexicals or how exactly a dialogue or a question is analysed. We will just show that any of the sentences can be interpreted by itself and fitted into the context. Now, here are the relevant structures for the particles that appear in the first text.

$$
\begin{array}{c}
/\mathsf{marnu}/ \\
\hline
\multicolumn{1}{|c|}{\langle * : * : \rhd \rangle} \\
\hline
\multicolumn{1}{|c|}{\varnothing} \\
\hline
\multicolumn{1}{|c|}{\mathsf{group}'(*)} \\
\hline
\end{array}
\qquad
\begin{array}{c}
/\textsc{acc}/ \\
\hline
\multicolumn{1}{|c|}{\langle * : \textsc{acc}^\frown * : \rhd \rangle} \\
\hline
\multicolumn{1}{|c|}{\varnothing} \\
\hline
\multicolumn{1}{|c|}{\mathsf{theme}'(*) \doteq \textsc{acc}^\frown *} \\
\hline
\end{array}
\qquad
\begin{array}{c}
/\mathsf{lha}/ \\
\hline
\multicolumn{1}{|c|}{\langle * : * : \rhd \rangle} \\
\hline
\multicolumn{1}{|c|}{\varnothing} \\
\hline
\multicolumn{1}{|c|}{\mathsf{past}'(*)} \\
\hline
\end{array}
$$

$$
\begin{array}{c}
/\textsc{all}/ \\
\hline
\multicolumn{1}{|c|}{\langle * : \textsc{all}^\frown * : \rhd \rangle} \\
\hline
\multicolumn{1}{|c|}{\varnothing} \\
\hline
\multicolumn{1}{|c|}{\mathsf{moves\text{-}to}'(\textsc{all}^\frown *, \textsc{acc}^\frown *, *).} \\
\hline
\end{array}
$$

The analysis of the accusative and the allative need comment. We construe the meaning of the accusative as identifying the object as the theme. This is of course superficial but does the trick here. Notice that in order for this proposal to work properly, the word that is in the accusative needs to be construed as the accusative object of some verb. Otherwise it is not meaningful to say that it is a theme. The analysis of the allative is as follows. It says that the accusative object of the verb moves towards the thing in the allative during event time. (This is an exegesis of $\mathsf{moves\text{-}to}'$.) It would be possible to make the semantics such that it does not depend on the verb being transitive and the object moving, but that would needlessly complicate matters. Now, nominative is treated like accusative. We assume that the object comes together with an indefinite determiner. (Moreover, nouns can be either bare or appear with an indefinite determiner. Alternatively, one might posit an empty indefinite.) (9.1) is then translated as follows:

$$\begin{array}{c}
(9.1)\\
\begin{array}{|c|}\hline \langle \bullet : \bullet \rangle \\ \hline \end{array}\\
\begin{array}{|c|}\hline \{\textsc{acc}^\frown\bullet\} \\ \hline
\mathsf{sub}'(\bullet) \doteq \mathsf{me}'; \mathsf{take}'(\bullet); \mathsf{past}'(\bullet);\\
\mathsf{theme}'(\bullet) \doteq \textsc{acc}^\frown\bullet;\\
\mathsf{group}'(\textsc{acc}^\frown\bullet); \mathsf{grandson}'(\textsc{acc}^\frown\bullet);\\
\mathsf{belong}'(\textsc{acc}^\frown\bullet, \mathsf{me}'); \mathsf{fish}'(\textsc{all}^\frown\bullet);\\
\mathsf{move\text{-}to}'(\textsc{all}^\frown\bullet, \textsc{acc}^\frown\bullet, \bullet).\\ \hline
\end{array}
\end{array}$$

Now we turn to the second sentence.

$$\begin{array}{c}
/\mathsf{nganangu}/\\
\begin{array}{|c|}\hline \langle * : * \rangle \\ \hline \end{array}\\
\begin{array}{|c|}\hline \varnothing \\ \hline
\mathsf{who}'(\textsc{gen}^\frown *);\\
\mathsf{belong}'(\textsc{gen}^\frown *, *).\\ \hline
\end{array}
\end{array}
\qquad
\begin{array}{c}
/\mathsf{ngara}/\\
\begin{array}{|c|}\hline \langle * : * : \triangleright \rangle \\ \hline \end{array}\\
\begin{array}{|c|}\hline \varnothing \\ \hline
\mathsf{card}'(*) > 1.\\ \hline
\end{array}
\end{array}$$

The speaker asks about the identity of the variable $\bullet^\frown\textsc{acc}$ of the first utterance. This however is not restorable from the utterance. Rather, putting everything together we get

$$\begin{array}{c}
(9.2)\\
\begin{array}{|c|}\hline \langle \bullet : \bullet \rangle \\ \hline \end{array}\\
\begin{array}{|c|}\hline \varnothing \\ \hline
\mathsf{who}'(\textsc{gen}^\frown\bullet); \mathsf{belong}'(\bullet, \textsc{gen}^\frown\bullet);\\
\mathsf{card}'(\bullet) > 1; \mathsf{child}'(\bullet).\\ \hline
\end{array}
\end{array}$$

At this point, we must make an additional assumption. The two discourses can neither be linked directly (via ';') nor using the full stop '.'). What happens is that the object carrying $\bullet$ in the second discourse refers to the object carrying $\textsc{acc}^\frown\bullet$ in the first. Therefore, we must substitute $\textsc{acc}^\frown\bullet$ for $\bullet$ in the second discourse and then use ';'. However, this might not be what is going on really. For, alternatively, Speaker B might have suffixed all the constituents with the accusative as in the last sentence. So, Speaker B simply refers to some discourse object, and as soon as that reference is resolved, an appropriate substitution is chosen. We shall not attempt to scrutinize this further, simply noticing that the processes we have described so far do not work automatically at this point.

Now we turn to the third sentence. Here we encounter two more suffixes

$$\begin{array}{c} /\text{BELONG}/ \\ \begin{array}{|c|}\hline \langle * : \text{BELONG}^\frown * : \triangleright\rangle \\ \hline \varnothing \\ \hline \mathsf{belong}'(*, \text{BELONG}^\frown *) \\ \hline \end{array} \end{array} \qquad \begin{array}{c} /\text{PRIV}/ \\ \begin{array}{|c|}\hline \langle * : \text{PRIV}^\frown * : \triangleright\rangle \\ \hline \varnothing \\ \hline \\ \neg \begin{array}{|c|}\hline \{\text{PRIV}^\frown *\} \\ \hline \mathsf{have}'(*, \text{PRIV}^\frown *). \\ \hline \end{array} \\ \\ \hline \end{array} \end{array}$$

The full interpretation of (9.3) is as follows.

$$\begin{array}{c} (9.3) \\ \begin{array}{|c|}\hline \langle \bullet : \bullet \rangle \\ \hline \varnothing \\ \hline \mathsf{belong}'(\text{BELONG}^\frown \bullet, \text{BELONG}^\frown\text{ACC}^\frown \bullet); \\ \\ \neg \begin{array}{|c|}\hline \{\text{PRIV}^\frown\text{BELONG}^\frown\text{ACC}^\frown \bullet\} \\ \hline \mathsf{have}'(\text{BELONG}^\frown\text{ACC}^\frown \bullet, \text{PRIV}^\frown\text{BELONG}^\frown\text{ACC}^\frown \bullet). \\ \hline \end{array}; \\ \\ \mathsf{that}'(\text{ACC}^\frown \bullet); \mathsf{card}'(\text{ACC}^\frown \bullet) > 1. \\ \hline \end{array} \end{array}$$

Finally, we turn to (9.4). Omitting the first sentence we treat only the second one, which consists of one single word. Its interpretation is

$$\begin{array}{c} /\mathsf{ngarrayangungraraa.}/ \\ \begin{array}{|c|}\hline \langle \bullet : \bullet \rangle \\ \hline \end{array} \\ \begin{array}{|c|}\hline \varnothing \\ \hline \mathsf{theme}'(\bullet) \doteq \textsc{acc}^\frown\bullet; \\ \mathsf{card}'(\textsc{acc}^\frown\bullet) > 1; \\ \mathsf{belong}'(\textsc{acc}^\frown\bullet, \textsc{gen}^\frown\textsc{acc}^\frown\bullet); \\ \mathsf{niece}'(\textsc{gen}^\frown\textsc{acc}^\frown\bullet). \\ \hline \end{array} \end{array}$$

It is now clear that (modulo adaptation of the second sentence) each of the sentences can be merged into the discourse using the semicolon, ie without adjusting the variables. This is so because the sentences are quite coherent, just exchanging information about the very same situation or event at hand, namely the taking for fishing of some people by Speaker A.

10 Conclusion

We have shown in this paper that languages can and do employ various ways of encoding dependencies: either by imposing word order, or by introducing special markers, which are called agreement markers or case markers, depending on the way they function in the syntax. These encodings are called *regimes*. There are two regimes which can stand alone in the sense that each sentence is unambiguous. These are the so–called Polish Notation (and its reverse image) and the case marking regime which is consistently word marking. For the positional regimes, Montague has defined a semantics that allows a compositional syntax–to–meaning translation. We have provided a semantics for the case marking languages. This semantics gives correct results for peripheral group marking languages as well as word marking languages, and so is quite flexible. Notice that neither Montague Semantics nor the semantics we have proposed here rely on the unambiguity of the language. Rather, they will predict that sentences can and must be ambiguous if the semantic mapping yields several distinct meanings, each corresponding to a different parse of that sentence. However, we have excluded that question from our discussion. In this way it was easier to motivate the usefulness of the various regimes.

This motivation can of course work only if one accepts the thesis that languages strive to make their sentences unambiguous. [22] This therefore comes down to claiming that avoiding ambiguity is a motor of language change. However, this works canonically only in one direction. Namely, when the morphology becomes unable to make enough distinctions, the languages tend to become more rigorous with word order. When the language develops some kind of marking mechanism, although it might afford to loosen the positional regime, this need not happen. A particularly outspoken proponent of this idea is Vennemann (see, for example, [24]). His claim is that languages with basic SOV order become SVO when the morphological distinctions between subject and object disappear as a consequence of the gradual morphological attrition. This is attested for virtually all Indo–European languages, for example. (See for example [14] for the development of Proto–IE to Late IE.) Now, while Vennemann's thesis is rather specific in the conditions and directions of change — he does not consider free word languages and the result is a specific regime that appears to be less than optimal from our results — he nevertheless emphasizes that it is the resulting ambiguity of sentences that motivates the change. It is worth a separate study why SVO languages are actually better suited than SOV languages. We shall briefly discuss some of the reasons. An interesting one is that question words and relativizers are clause initial and so a transitive question or relative clause would necessarily be ambiguous. Of course, it is not logically nec-

[22] A note of clarification is perhaps in order. It might be undisputed that speakers strive to make their sentences unambiguous. (This is one of Grice's maxims!) Since speakers can normally take the context into account, they might not always produce fully unambiguous sentences, and — as one learns — many sentences are ambiguous at closer look (see Altman [1]). Yet, there are certain basic ambiguities that must in general be avoided, such as a potential confusion between subject and object (*John pestered Peter* versus *Peter pestered John*). If that is accepted, then it is clear that speakers will tend not to use those kinds of constructions that are ambiguous. Initially, these constructions might still be grammatical, but the less they are used the less grammatical they appear. In the end, they will be judged ungrammatical, and then we say that the language has changed. For example, scrambling in German can yield sentences that are disfavoured for precisely the reason that they appear to be SOV but are intended as OSV. We predict therefore that German moves in direction of a language in which scrambling of the direct object is prohibited. In this way, pragmatic principles can bear on the question of language, and we can attribute to language roughly the same tendencies as to individual speakers.

essary for the question word or the relativizer to be clause initial, and this in turn means that the direction of language change depends in part on other characteristics of language. These might be universal or accidental, which means that Vennemann's thesis may actually be false in general should it turn out that some of the characteristics are not universal. But this has not been the topic of the present paper. What we were interested in are the formal (and therefore inescapable) consequences of particular language characteristics, in order to provide the background for such theories as those of Vennemann's.

The model presented in this paper is however still too simplistic. Even though there are languages which are consistently word marking, they do not match the theoretical model in cases of modification and coordination. A modifier does not case mark its argument, neither does a coordinator. It follows therefore that even the consistently word marking languages must have certain positional regimes or else a different semantic mapping algorithm must be found. This question needs investigation.

References

[1] Gerry Altmann. *The Ascent of Babel.* MIT Press, Cambridge, Massachussetts, 1998.

[2] Mark C. Baker. *The Polysynthesis Parameter.* Oxford Studies in Comparative Syntax. Oxford University Press, 1996.

[3] Barry J. Blake. *Case.* Cambridge Textbooks in Linguistics. Cambridge University Press, 1994.

[4] Alan C. Dench. *Martuthunira. A Language of the Pilbara Region of Western Australia.* Number C – 125 in Pacific Linguistics. Australian National University, 1995.

[5] R. M. W. Dixon. *Ergativity.* Number 69 in Cambridge Studies in Linguistics. Cambridge University Press, Cambridge, 1994.

[6] Christian Ebert. Formal Analysis of Languages with Stacked Cases. Master's thesis, Mathematisches Institut, Universität Heidelberg, 1999.

[7] Nicholas D. Evans. *A Grammar of Kayardild. With Historical–Comparative Notes on Tangkic.* Mouton de Gruyter, Berlin, 1995.

[8] Kit Fine. Transparency, Part I: Reduction. Unpublished manuscript, UCLA, 1992.

[9] Michael A. Harrison. *Introduction to Formal Language Theory.* Addison Wesley, Reading (Mass.), 1978.

[10] Edward L. Keenan. On semantics and binding theory. In John A. Hawkins, editor, *Explaining Language Universals.* 1988.

[11] Edward L. Keenan and Leonard M. Faltz. *Boolean Semantics for Natural Language.* Number 23 in Synthese Language Library. Reidel, Dordrecht, 1985.

[12] Marcus Kracht. Agreement Morphology, Argument Structure and Syntax. Unpublished manuscript, available at `http://www.math.fu-berlin.de/ kracht`, 1999.

[13] Christian Lehmann. Word order change by grammaticalization. In Marinel Gerritsen and Dieter Stein, editors, *Internal and External Factors in Language Change*, number 61 in Trends in Linguistics, Studies and Monographs, pages 395 – 416. Mouton de Gruyter.

[14] Winfred Lehmann. A Disucssion of Compound and Word Order. In C. N. Li, editor, *Word Order and Word Order Change*, pages 149 – 161. University of Texas Press, Austing and London, 1975.

[15] C. N. Li and S. A. Thompson. Subject and topic: A new typology of language. In C. N. Li, editor, *Subject and Topic*, pages 458 – 489. Academic Press, 1976.

[16] Jens Michaelis and Marcus Kracht. Semilinearity as a syntactic invariant. In Christian Rétoré, editor, *Logical Aspects of Computational Linguistics (LACL '96)*, number 1328 in Lecture Notes in Computer Science, pages 329 – 345, Heidelberg, 1997. Springer.

[17] Johanna Nichols. Head–marking and dependent–marking grammar. *Language*, 62:56 – 119, 1986.

[18] Rachel Nordlinger. *Constructive Case. Evidence from Australian Languages.* Dissertations in Linguistics. CSLI, Stanford, 1998.

[19] Richard T. Oehrle. Multi–Dimensional Compositional Functions as a Basis for Grammatical Analysis. In Emmon Bach Richard T. Oehrle and Deirdre Wheeler, editors, *Categorial Grammars and Natural Language Structures*, pages 349 – 389. Reidel, Dordrecht, 1988.

[20] Frank R. Palmer. *Grammatical roles and relations.* Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge, 1994.

[21] Frans Plank. (Re-)Introducing Suffixaufnahme. In Frans Plank, editor, *Double Case. Agreement by Suffixaufnahme*, pages 3 – 110. Oxford University Press, 1995.

[22] Susan Stelle. Word Order Variation. A Typological Study. In Joseph H. Greenberg, editor, *Universals of Human Language. Vol 4: Syntax*, pages 578 – 623. Stanford University Press, Stanford, 1978.

[23] Robert D. Van Valin and Randy J. LaPolla. *Syntax. Structure, meaning and function.* Cambridge Textbooks in Linguistics. Cambridge University Press, 1997.

[24] Theo Vennemann. An explanation of drift. In C. N. Li, editor, *Word Order and Word Order Change*, pages 269 – 305. University of Texas Press, Austing and London, 1975.

[25] Kees F. M. Vermeulen. Merging without Mystery or: Variables in Dynamic Semantics. *Journal of Philosophical Logic*, 24:405 – 450, 1995.

[26] Kees F. M. Vermeulen and Albert Visser. Dynamic bracketing and discourse representation. *Notre Dame Journal of Formal Logic*, 37:321 – 365, 1996.