Formal Syntax and Semantics of Case Stacking Languages

Christian Ebert
Lehrstuhl für Computerlinguistik
Universität Heidelberg
Karlstr. 2
D-69117 Heidelberg

Marcus Kracht
II. Mathematisches Institut
Freie Universität Berlin
Arnimallee 3
D-14195 Berlin

Abstract

In this paper the phenomenom of case stacking is investigated from a formal point of view. We will define a formal language with idealized case marking behaviour and prove that stacked cases have the ability to encode structural information on the word thereby allowing for unrestricted word order. Furthermore, the case stacks help to compute this structure with a low complexity bound. As a second part we propose a compositional semantics for languages with stacked cases and show how this proposal may work for our formal language as well as for an example from Warlpiri.

1 Introduction

Case stacking is a phenomenom that occurs in many Australian languages (such as Warlpiri and Kayardild) and e.g. Old Georgian. Case stacking is known to pose problems for the treatment of case in many formal frameworks today[1]. In (Nordlinger, 1997) the problem was attacked by extending the framework of LFG. Nordlinger claims that case morphology can construct grammatical relations and larger syntactic contexts.

In Section 2 we will introduce an ideal languge, which exhibits perfect marking. This language captures Nordlinger's idea of case as constructors of grammatical relations, but is independent of any syntactic framework. We will prove of this language that the case stacks provide all the information needed for reconstructing the functor-argument relations and the syntactic context a word appears in. Additionally, since structure lies encoded in these case stacks there is no need to assume any phrase structure or restriction on word order. At the end of this section we consider the computational complexity of our language and draw some conclusions about grammar formalisms that are able to generate it.

In Section 3 we propose a compositional semantics for our case stacking languages. Unlike Montague semantics, where the management of variables is not made explicit, we will use referent systems to keep track of variables and to make semantic composition efficently computable[2]. The proposal will be applied to our formal language and to an example from Warlpiri.

2 Syntax

In this section a perfectly case marked formal language will be defined and investigated. The definition of this language is based on *terms* consisting of functors and arguments and thus cases will be taken to mark arguments.

In the following we let $\mathbb{N}$ denote the set of non-negative integers and $^\frown$ the concatenation of strings, which is often omitted. We shall use `typewriter font` to denote true characters in print for a formal language.

2.1 Basic Definitions

An abstract definition of terms runs as follows. Let F be a set of symbols and $\Omega\colon F \to \mathbb{N}$ a function. The pair $\langle F, \Omega\rangle$ is called a *signature*. We shall often write Ω in place of $\langle F, \Omega\rangle$. An element $f \in F$ is called a *functor* and $\Omega(f)$ the *arity* of f. We let $\omega := \max\{\Omega(f) \mid f \in F\}$ denote the maximal arity. Terms are denoted here by strings in Polish Notation, for simplicity.

Definition 1. *Let Ω be a signature. A* ***term*** *over Ω is inductively defined as follows.*

[1] see (Nordlinger, 1997) and (Malouf, 1999) for a discussion on LFG and HPSG respectively.

[2] see (Vermeulen, 1995) and (Kracht, 1999) on these issues.

1. *If $\Omega(f) = 0$, then f is a term.*
2. *If $\Omega(f) > 0$ and t_i, $1 \leq i \leq \Omega(f)$, are terms, so is $ft_1 \dots t_{\Omega(f)}$.*

The set C of *case markers* will be the set $\{1, \dots, \omega\}$, which we assume to be disjoint from F. Given a term, each functor will be case marked according to the argument position it occupies. This is achieved through the notion of a *unit*, which consists of a functor and a sequence of case markers called *case stack*.

Definition 2. *Let t be a term over a signature Ω. The **corresponding bag,** $\Delta(t)$, is inductively defined as follows.*

1. *If $t = f$, then $\Delta(t) := \{t\}$.*
2. *If $t = ft_1 \dots t_n$, then $\Delta(t) := \{f\} \cup \bigcup_{i=1}^{n} \{z^\frown i \mid z \in \Delta(t_i)\}$.*

*An element $f\gamma \in \Delta(t)$ is called a **unit** and $\gamma \in C^*$ its **case stack**.*

For example, if f, g, and x are functors of arity 2, 1 and 0, respectively, the bag $\Delta(fxgx)$ is $\{\mathtt{f}, \mathtt{x1}, \mathtt{g2}, \mathtt{x12}\}$. The meaning of a unit `x12` could be described by 'x is the functor of the first argument of the second argument of the term'.

Definition 3. *Let t be a term over a signature Ω, $\Delta(t)$ the corresponding bag and $\Delta(t) = \{\delta_i \mid i \leq n\}$ an arbitrary enumeration of its units. Then the string $\delta_1{}^\frown\delta_2{}^\frown \dots {}^\frown\delta_{n-1}{}^\frown\delta_n$ is said to be a $\Delta(t)$**-string**.*

Some of the $\Delta(fxgx)$-strings are e.g. `fx1g2x12` and `g2x1fx12`. We are now prepared to define a formal language over the alphabet $F \cup C$ by collecting all $\Delta(t)$-strings for a given signature:

Definition 4. *Let Ω be a signature. The **ideal case marking language** $\mathcal{ICML}^\Omega$ **over this signature** consists of all $\Delta(t)$-strings such that t is a term over Ω.*

2.2 Trees and Unique Readability

There is a strong correspondence between bags and labelled trees since case stacks can be identified with *tree addresses*:

Definition 5. *A nonempty finite set $D \subseteq \mathbb{N}_+^*$ is a **tree domain** if the following hold:*

1. *$\epsilon \in D$.*
2. *If $d_1 d_2 \in D$ then $d_1 \in D$.*
3. *If $di \in D$, $i \in \mathbb{N}$ then $dj \in D$ for all $j < i$.*

*The elements of a tree domain are called **tree addresses**. A **Ω-labelled tree** is a pair $\langle D, \tau \rangle$ such that D is a tree domain and $\tau \colon D \to F$ a labelling function such that the number of daughters of $d \in D$ is exactly $\Omega(\tau(d))$.*

To formalize the correspondence we define a function T that assigns every bag $\Delta(t)$ a Ω-labelled tree:

$$T(\{\mathtt{f}_1\gamma_1, \dots, \mathtt{f}_n\gamma_n\}) := \langle \{\gamma_i^R \mid 1 \leq i \leq n\}, \\ \tau : \gamma_i^R \mapsto f_i \ (1 \leq i \leq n) \rangle$$

The function T reverses the case stacks of all units to get a set of tree addresses. Then the functor of the unit is assigned to the tree address. E.g. if the bag contains a unit `g321` the resulting tree domain will contain a tree address 123 and the labelling function will assign g to it.

Similarly one can define an inverse function assigning a bag to each Ω-labelled tree. Thus there is a bijection between Ω-labelled trees and bags. Therefore different bags correspond to different ordered labelled trees. This shows that we have unique readability for bags and since every $\mathcal{ICML}^\Omega$ string can be uniquely decomposed into its units we may state the following proposition.

Proposition 6. *Let Ω be a signature. Then every $\mathcal{ICML}^\Omega$ string is uniquely readable.*

2.3 Pumpability and Semilinearity

We will first consider the property of being *finitely pumpable* as defined in (Groenink, 1997).

Definition 7. *A language L is **finitely pumpable** if there is a constant c such that for any $w \in L$ with $|w| > c$, there are a finite number k and strings $u_0, \dots, u_k$ and $v_1, \dots, v_k$ such that $w = u_0 v_1 u_1 v_2 u_2 \cdots v_k u_k$ and for each $i, 1 \leq |v_i| < c$ and for any $p \geq 0$ the string $u_0 v_1^p u_1 v_2^p u_2 \cdots v_k^p u_k$ belongs to L.*

Proposition 8. *Let Ω be a signature. Then $\mathcal{ICML}^\Omega$ is not finitely pumpable.*

Proof. It is easy to observe that the pumpable parts cannot contain a functor since that would lead to pumped strings containing the same units more than once. Hence the number of units cannot be increased by pumping and all pumpable parts must consist of case markers solely. But since the length of an $\mathcal{ICML}^{\Omega}$ string consisting of a fixed number of units is bounded each pumpable string could be pumped up such that it exceeds this bound. Thus $\mathcal{ICML}^{\Omega}$ is not finitely pumpable at all. □

Now we are concerned with *semilinearity.*

Definition 9. *Let $M \subseteq \mathbb{N}^n$. Then M is a*

1. ***linear set**, if for some $k \in \mathbb{N}$ there are $u_0, \ldots, u_k \in \mathbb{N}^n$, such that $M = \{u_0 + \sum_{i=1}^{k} n_i u_i \mid n_i \in \mathbb{N}\}$,*
2. ***semilinear set**, if for some $k \in \mathbb{N}$ there are linear sets $M_1, \ldots, M_k \subseteq \mathbb{N}^n$, such that $M = \bigcup_{i=1}^{k} M_i$.*

*A language L over an alphabet $\Sigma = \{w_i \mid 0 \leq i < n\}$ is called a **semilinear language** if its image under the **Parikh mapping** is a semilinear set, where the Parikh mapping $\Psi : \Sigma^* \to \mathbb{N}^n$ is defined as follows:*

$$\begin{array}{lcll} \epsilon & \mapsto & \langle 0, \ldots, 0\rangle & \\ w_i & \mapsto & e^{(i)} & \text{for } 0 \leq i < n \\ \alpha\beta & \mapsto & \Psi(\alpha) + \Psi(\beta) & \text{for all } \alpha, \beta \in \Sigma^* \end{array}$$

where $e^{(i)}$ is the $i+1$-th unit vector, which consists of zeros except for the i-th component, which is 1.

Note that – given a term t – the Parikh image of all $\Delta(t)$-strings is the same since these are just concatenations of different permutations of the units in $\Delta(t)$.

In the following we make use of a proof technique used in (Michaelis and Kracht, 1997) to show that Old Georgian is not a semilinear language. We cite a special instance of a proposition given therein:

Proposition 10. *Let $P(k) = \frac{\omega}{2}k^2 + \frac{2-\omega}{2}k$ and M be a subset of $\mathbb{N}^n$, where $n \geq 2$, which has the properties*

1. *For any $k \in \mathbb{N}_+$ there are some numbers $l_2^{(k)}, \ldots, l_{n-1}^{(k)} \in \mathbb{N}$ for which the n-tuple $\langle k, P(k), l_2^{(k)}, \ldots, l_{n-1}^{(k)}\rangle$ belongs to M.*
2. *For any $k \in \mathbb{N}_+$ the value $P(k)$ provides an upper bound for the second component l_1 of any n-tuple $\langle k, l_1, \ldots, l_{n-1}\rangle \in M$ (that means $l_1 \leq P(k)$ for any such n-tuple).*

Then M is not semilinear.

In order to investigate the semilinearity of $\mathcal{ICML}^{\Omega}$ we choose distinct symbols $f, x \in F$, such that $\Omega(f) = \omega$ and $\Omega(x) = 0$. We shall construct terms s_i by the following inductive definition:

1. $s_0 := x$
2. $s_n := f(s_{n-1}, x, \ldots, x) \qquad \text{for } n > 0$

It is easy to observe that by virtue of construction s_n consists of n leading functors f and that in each iteration the number of x increases by $(\omega - 1)$.

Lemma 11. *Let $F \cup C = \{f, 1, x, 2, \ldots, \omega, f_1, \ldots, f_{|F|-2}\}$ be an enumeration of the alphabet underlying $\mathcal{ICML}^{\Omega}$, where $f_1, \ldots, f_{|F|-2}$ are the remaining functors in $F - \{f, x\}$ Then the Parikh image of some $\Delta(s_n)$-string δ_n is*

$$\Psi(\delta_n) = \langle n, \frac{\omega}{2}n^2 + \frac{2-\omega}{2}n, (\omega-1)n+1, \underbrace{n, \ldots, n}_{\omega-1}, \underbrace{0, \ldots, 0}_{|F|-2}\rangle$$

Furthermore, $\frac{\omega}{2}n^2 + \frac{2-\omega}{2}n$ imposes an upper bound on the second component of $\Psi(\delta_n)$.

Proof. The first part of the lemma can be proved in a straightforward way by induction on n. The claim on the upper bound follows from the observation that the number of occurrences of case marker 1 can be maximized by repeated embedding of terms in the first argument position. □

Proposition 12. *Let Ω be a signature. Then $\mathcal{ICML}^{\Omega}$ is not semilinear.*

Proof. Let $n = \omega + |F|$ and consider the linear, and hence semilinear, set $R :=$

$$\{(n_2(\omega-1)+1)e^{(2)} + \sum_{i=0, i\neq 2}^{\omega+2} n_i e^{(i)} \mid n_i \in \mathbb{N}\}$$

Then the full preimage L_R of R under the Parikh map consists of all strings which contain $n_2((\omega - 1) + 1)$ occurrences of the symbol `x` (where n_2 is any number) and any number of occurrences of the symbols $\mathtt{f}, \mathtt{1}, \ldots, \omega$, and no other symbols. We define the language L_M as the set of all strings belonging to L_R and the ideal case marking languages. Then L_M contains all $\Delta(s_n)$-strings.

Considering the Parikh image M of L_M we get

$$\begin{aligned} M = \Psi[L_M] &= \Psi[L_R] \cap \Psi[\mathcal{ICML}^\Omega] \\ &= R \cap \Psi[\mathcal{ICML}^\Omega] \end{aligned}$$

because of the definition of L_R as the full preimage of R. But then the set M fulfills the conditions of Proposition 10 due to Lemma 11. Hence M is not semilinear. Since R is semilinear by definition and semilinearity is closed under intersection $\mathcal{ICML}^\Omega$ is not semilinear. □

2.4 Computational Complexity

In this subsection the computational complexity of $\mathcal{ICML}^\Omega$ is considered. The results are achieved by defining a 3-tape-Turing machine acceptor (depending on a given signature) that recognizes $\mathcal{ICML}^\Omega$.

Proposition 13. *Let Ω be a signature. Then*

$$\mathcal{ICML}^\Omega \in DTIME(n\sqrt{n}\log n)$$
$$\mathcal{ICML}^\Omega \in DSPACE(n).$$

Proof. In the following we let n denote the length of the input string. The Turing machine algorithm can be subdivided into three main parts:

1. The input string is segmented into its units: The algorithm steps through the input and adds separation markers in between two units. This can be done in $O(n)$ time.
2. The units are sorted according to their case stacks: More formally a *2-way straight merge sort* is performed. This sorting algorithm is known for its worst case optimal complexity: it performs the sort of k keys in $O(k \log k)$ steps. In our case the keys are units and thus their number is clearly bounded by n. The additional square root factor comes from the comparison step. One can show that the maximal length of a case stack occuring in an $\mathcal{ICML}^\Omega$ string of length n is bounded above by $O(\sqrt{n})$. Hence a comparison of two units takes at most $O(\sqrt{n})$ steps. Thus the overall complexity of the sorting part is $O(n\sqrt{n}\log n)$.
3. The sorted sequence of units is checked: The algorithm successively generates case stacks according to the functors it has read. Each case stack is compared to the unit of the input. If they coincide the algorithm advances to the next unit on the input and generates the next case stack. After all case stacks have been generated the whole input string must have been worked through. In this case the algorithm accepts. This can be done in $O(n)$ time.

Summing up the complexities of these three parts shows that the time complexity is as claimed in the proposition. Furthermore, the algorithm uses only the cells needed by the input plus at most $k - 1$ cells for additional separation markers (due to the first part), where k is the number of units the input string consists of. This shows that the space complexity is linear. □

2.5 Discussion

A first conclusion we may draw is that cases have the ability to construct the context they appear in. $\mathcal{ICML}^\Omega$ strings encode the same structural information as ordered labelled trees do thereby allowing unconstrained order of units. Additionally each such string can be read unambigously. This was shown by means of a bijection between bags and ordered labelled trees.

The fact that ideal case marking languages are neither finitely pumpable nor semilinear means that they fall out of a lot of hierarchies of formal languages. As (Weir, 1988) shows, *multi-component tree adjoining grammars*[3] generate only semilinear languages. Consequently, ideal case marking languages are not MCTALs. However, (Groenink, 1997) defines a class of grammars, called *simple literal movement grammars*,

[3] and hence *linear context-free rewrite systems*, which are shown to be weakly equivalent to MCTAGs in (Weir, 1988)

which generate all and only the PTIME recognizable languages. Ideal case marking languages should therefore be generated by some simple literal movement grammar.

We note furthermore that the (theoretical) time complexity is significantly better than the best known for recognizing context-free grammars. In fact, we implemented a practically applicable algorithm which constructs the corresponding tree out of a given $\mathcal{ICML}^{\Omega}$ string in linear time (in average).

3 Semantics

We are now going to propose a semantics for languages with stacked cases. The basic principle is rather easy: we are going to identify variables by case stacks thereby making use of *referent systems*.

3.1 Referent Systems

The semantics uses two levels: a DRS-level, which contains DRSs, and a referent level, which talks about the names of the referents used by the DRS. Referent systems were introduced in (Vermeulen, 1995). We keep the idea of a referent system as a device which administrates the variables (or referents) under merge. The technical apparatus is however quite different. In particular, the referent systems we use define explicit global string substitutions over the referent names.

There is one additional symbol $\circ$. It is a variable over names of referents. If we assume that a functor g has meaning $\mathtt{g}$ a simple lexical entry for g looks like this:

/g/
$\circ : \circ$
$\varnothing$
$\circ \doteq \mathtt{g}(1^\frown\circ, 2^\frown\circ, \ldots, \Omega(g)^\frown\circ)$

Here, the upper part is the referent system, and the lower part an ordinary DRS, with a head section, containing a set of referents, and a body section, containing a set of clauses. This means that the semantics of a functor g is given by the application of $\mathtt{g}$ to its arguments. However, instead of variables x, y, etc. we find $1^\frown\circ, 2^\frown\circ$, etc. The semantics of a 0-ary functor x and a case marker, say $\mathtt{2}$, are:

/x/
$\circ : \circ$
$\varnothing$
$\circ \doteq \mathtt{x}$

/2/
$\circ : 2^\frown\circ$
$\varnothing$
$\varnothing$

When two such structures come together they will be *merged*. The merge operation $\oplus$ takes two structures and results in a new one thereby using the referent systems to substitute the names of referents if necessary and then taking the union of the sets of clauses. E.g. the result of the merge $\mathtt{/g/} \oplus \mathtt{/2/}$ is

/g2/
$\circ : \circ$
$\varnothing$
$2^\frown\circ \doteq \mathtt{g}(12^\frown\circ, 22^\frown\circ, \ldots, \Omega(g)2^\frown\circ)$

The meaning of $\circ : 2^\frown\circ$ is as follows. If some structure A is merged with one bearing that referent system, then all occurrences of the variable $\circ$ in A are replaced by $2^\frown\circ$. As the resulting referent system we get $\circ : \circ$. This is exactly what is done in the merge shown above. We shall call a structure with referent system $\circ : \circ$ *plain*. Merge is only defined if at least on structure is plain.

3.2 Semantics for $\mathcal{ICML}$

To see how the semantics works we shall reproduce an earlier example and take the $\mathcal{ICML}^{\Omega}$ string `g2x1fx12`. Motivated by the definition of the ideal case marking language we shall agree to the conventions that

1. Case markers may only be suffixes
2. Case markers may only be attached to functors or case marked functors

By these conventions the string under consideration must be parsed as `(g2)(x1)(f)((x1)2)`. They force us to combine the functors with their case stacks first and afterwards combine the units. We shall understand that this is a syntactic restriction and not due to any semantics.

The composition of `g` and `2` was already shown above and is repeated on the left hand side, using that $\Omega(g) = 1$. The result of composing `x` and `1` is shown to the right.

/g2/

$\circ : \circ$
$\varnothing$
$2^\frown\circ \doteq g(12^\frown\circ)$

/x1/

$\circ : \circ$
$\varnothing$
$1^\frown\circ \doteq x$

Merging these two structures we get

/g2x1/

$\circ : \circ$
$\varnothing$
$2^\frown\circ \doteq g(12^\frown\circ)$
$1^\frown\circ \doteq x$

Together with f this gives

/g2x1f/

$\circ : \circ$
$\varnothing$
$2^\frown\circ \doteq g(12^\frown\circ)$
$1^\frown\circ \doteq x$
$\circ \doteq f(1^\frown\circ, 2^\frown\circ)$

By composing the structures for x and 1 we get the structure /x1/ shown above. We merge this one with that for 2 and get

/x12/

$\circ : \circ$
$\varnothing$
$12^\frown\circ \doteq x$

The merge of the two structures above finally gives

/g2x1fx12/

$\circ : \circ$
$\varnothing$
$12^\frown\circ \doteq x$
$2^\frown\circ \doteq g(12^\frown\circ)$
$1^\frown\circ \doteq x$
$\circ \doteq f(1^\frown\circ, 2^\frown\circ)$

We shall verify that the value of $\circ$ is actually the same as the value of $f(x, g(x))$. Notice first that in the body of the DRS we find that $12^\frown\circ$ and $1^\frown\circ$ have the same value as x. We may therefore reduce the body of this structure to

$$2^\frown\circ \doteq g(x)$$
$$\circ \doteq f(x, 2^\frown\circ)$$

Finally we may replace $2^\frown\circ$ by $g(x)$ in the second line. We get then

$$\circ \doteq f(x, g(x))$$

which is the intended result.

After the semantics of the units has been computed the order of merge is unimportant. If we choose to merge these semantics in an order different from the one above, we get the same result.

3.3 An example from Warlpiri

To show how this proposal may work for natural languages we give an example from Warlpiri[4] in which case stacking occurs. We have to deal with the four case markers ergative (ERG), past tense (PST), absolutive (ABS), and locative (LOC).

Japanangka-rlu *luwa-rnu* *marlu*
Japanangka-ERG shoot-PST kangaroo-ABS
pirli-ngka-rlu
rock-LOC-ERG

Japanangka shot the kangaroo (while) on the rock

We extend the proposal by taking into account that cases may not only function as argument markers but have a semantics, too. This actually does not make much of a difference for this calculus. We propose the following semantics for the locative and the past tense case marker

/LOC/

$\circ : \text{LOC}^\frown\circ$
$\varnothing$
$\text{located}'(\circ, \text{LOC}^\frown\circ)$

/PST/

$\circ : \circ$
$\varnothing$
$\text{past}'(\circ)$

So, when the locative is attached, it says that the thing to which it attaches is located somewhere. Here, $\circ$ represents the thing that is located, while $\text{LOC}^\frown\circ$ is the location. The past tense semantics simply says that the thing which it attaches to happened in the past.

We construe the meaning of the ergative as being the actor and the meaning of the absolutive as being the theme[5].

[4] This example is taken from (Nordlinger, 1997), p.171

[5] In fact, ergative and absolutive should mark for grammatical functions, but since linking of grammatical functions and actants is quite a complicated matter (see (Kracht, 1999)) we make this simplification.

/ERG/

$\circ : \text{ERG}^\frown\circ$
$\varnothing$
$\mathsf{actor'}(\circ) \doteq \text{ERG}^\frown\circ$

/ABS/

$\circ : \text{ABS}^\frown\circ$
$\varnothing$
$\mathsf{theme'}(\circ) \doteq \text{ABS}^\frown\circ$

Again, we shall agree in using the conventions stated in subsection 3.2. First we have to attach the case markers to compose the resulting structures afterwards. The semantics of the proper noun *Japanangka* is taken to be a plain structure with body $\circ \doteq \mathsf{japanangka'}$. The composition of this structure and the ergative semantics yields

/*Japanangka*-ERG/

$\circ : \circ$
$\varnothing$
$\text{ERG}^\frown\circ \doteq \mathsf{japanangka'}$
$\mathsf{actor'}(\circ) \doteq \text{ERG}^\frown\circ$

We assume that the nominals */pirli/* and */marlu/* have the following lexical entries:

/*pirli*/

$\circ : \circ$
$\{\circ\}$
$\mathsf{rock'}(\circ)$

/*marlu*/

$\circ : \circ$
$\{\circ\}$
$\mathsf{kangaroo'}(\circ)$

Thus the semantic composition of *pirli*, LOC, and ERG gives:

/*pirli*-LOC-ERG/

$\circ : \circ$
$\{\text{LOC}^\frown\text{ERG}^\frown\circ\}$
$\mathsf{rock'}(\text{LOC}^\frown\text{ERG}^\frown\circ)$
$\mathsf{actor'}(\circ) \doteq \text{ERG}^\frown\circ$
$\mathsf{located'}(\text{ERG}^\frown\circ, \text{LOC}^\frown\text{ERG}^\frown\circ)$

Similarly we can compose *marlu* with the absolutive case:

/*marlu*-ABS/

$\circ : \circ$
$\{\text{ABS}^\frown\circ\}$
$\mathsf{kangaroo'}(\text{ABS}^\frown\circ)$
$\mathsf{theme'}(\circ) \doteq \text{ABS}^\frown\circ$

The semantics of the verb is shown on the left hand side and its composition with /PST/ on the right hand side.

/*luwa*/

$\circ : \circ$
$\varnothing$
$\mathsf{shoot'}(\circ)$

/*luwa*-PST/

$\circ : \circ$
$\varnothing$
$\mathsf{shoot'}(\circ)$
$\mathsf{past'}(\circ)$

Finally by merging the structures /*Japanangka*-ERG/, /*luwa*-PST/, /*marlu*-ABS/, and /*pirli*-LOC-ERG/ in any order, we get the following result.

$\circ : \circ$
$\{\text{ABS}^\frown\circ, \text{LOC}^\frown\text{ERG}^\frown\circ\}$
$\text{ERG}^\frown\circ \doteq \mathsf{japanangka'}$
$\mathsf{shoot'}(\circ)$
$\mathsf{past'}(\circ)$
$\mathsf{actor'}(\circ) \doteq \text{ERG}^\frown\circ$
$\mathsf{theme'}(\circ) \doteq \text{ABS}^\frown\circ$
$\mathsf{kangaroo'}(\text{ABS}^\frown\circ)$
$\mathsf{rock'}(\text{LOC}^\frown\text{ERG}^\frown\circ)$
$\mathsf{located'}(\text{ERG}^\frown\circ, \text{LOC}^\frown\text{ERG}^\frown\circ)$

It says that there was an event of shooting in the past, whose actor is Japanangka and whose theme is something that is a kangaroo, and that there is a rock, such that Japanangka is located on it. Note that the only syntactic restriction were the conventions stated in subsection 3.2 and that we did not make any further assumptions on syntactic structure or word order.

References

Annius Groenink. 1997. *Surface without structure*. Ph.D. thesis, Centrum voor Wiskunde en Informatica, Amsterdam.

Marcus Kracht. 1999. Agreement morphology, argument structure and syntax. Course material ESSLLI '99.

Robert Malouf. 1999. A head-driven account of long-distance case assignment. Paper presented on HPSG '99.

Jens Michaelis and Marcus Kracht. 1997. Semilinearity as a syntactic invariant. In Christian Retoré, editor, *Proceedings of the 1st International Conference on Logical Aspects of Computational Linguistics (LACL-96)*, volume 1328 of *LNCS*, pages 329–345, Berlin, September 23–25. Springer.

Rachel Nordlinger. 1997. *Constructive Case. Dependent-Marking Nonconfigurality in Australia*. Ph.D. thesis, Department of Linguistics, Stanford University, Stanford.

Kees F. M. Vermeulen. 1995. Merging without mystery or: Variables in dynamic semantics. *Journal of Philosophical Logic*, 24:405–450.

David J. Weir. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania, PA.