CONTEXT MODIFICATION IN ACTION

ALBERT VISSER

Abstract. In this paper we develop the positive fragment of Context Modification Logic. This logical system is a variant of Dynamic Predicate Logic that employs multiple-access variables and that treats argument places in the same way as Latin. The positive fragment is purely incremental.

Beste Martin,

Het boek over Bronnen en Stromen van Informatie *is er helaas nooit gekomen. Hier is, een beetje verlaat, mijn geplande bijdrage aan dat boek. Het was vast geweldig geworden.*

Jij bent één van de mensen van wie ik veel geleerd heb en van wie ik ongetwijfeld nog veel zal leren.

Ik wens je veel wetenschappelijke en filosofische inspiratie toe in de komende jaren!

Het beste,
Albert

Date: August 24, 2011.
2000 *Mathematics Subject Classification.* 03B65, 68T50, 91F20.
Key words and phrases. Discourse, Discourse Structure, DRT, Dynamic Predicate Logic, Context, Natural Language, Meaning, Information.

Contents

1. Introduction

Context Modification Logic (CML) can be best viewed as an experimental domain to try out certain philosophical views about language. We briefly review some of these views in the next subsection. Honesty requires us to warn the reader that, even if the aims formulated below are high and ambitious, the fragment developed in the paper is of very restricted scope.

1.1. **Themes.** We present a quick tour around the various design ideas of Context Modification Logic, CML.

1.1.1. *Dynamics.* Meanings are actions. These actions modify the information state of the hearer.[1] More precisely, a meaning of an expression is an *action-type*: it is what is in common to how attending to utterances of the expression modifies the states of hearers.

It would seem that such an activistic semantics diverges sharply from the classical truth-conditional idea. In one sense that is so. E.g. the meanings for which we give a compositional account are actions, not truth conditions. In another sense this is not so. Actions have success-conditions. These success conditions contain the truth-conditions. Thus, classical meanings can be regained as projections of dynamic meanings.

In Dynamic Semantics, actions are modeled as input-output relations —in the venerable tradition of mathematics where a function is considered to be a set of pairs. In this paper we will follow this style of modeling.[2]

1.1.2. *Alternative View of Syntax.* Here is a standard picture of interpretation. We are offered a piece of text in the form of a string (an element of the free monoid). The first stage of interpretation is parsing in which we produce a syntactic analysis of the string, transforming it into a structured object, an element of an appropriate free algebra: the syntactic algebra. Next we assign meanings to the atomic constituents (the generators) and compute the meanings of the complex constituents inside-out.

Typical for the usual functional architecture is that the basic operation of our meaning algebras is function-application and that, thus, (i) the interaction of meanings is always directed and (ii) both the syntactic and the meaning algebras can have very little further mathematical structure. Since the functional architecture does not always fit natural language, higher types are introduced to make things fitting —as in the analysis of: *a dog sees a cat*.[3]

The approach proposed in this paper is different. Here parsing serves to reveal the discourse actions that are not explicitly written down. However, we still consider the product of the parsing process as a string of symbols (an element of the free monoid) rather than as a structured object (an element of a free algebra). We compute the meanings incrementally moving from left to right. The syntactical elements, like brackets and agreement morphemes, will guide the interpretation

[1] This perspective is undoubtedly too restricted. What we really want is a dialogical model. We consider the one-sided sender-receiver model as a useful first abstraction.

[2] For a full account, we would need a bit more: to model meanings as natural transformations. See also the remarks in Section 9.

[3] We would rather like to follow the methodology that higher types are only introduced when there are 'ontological reasons' —as in the case of generalized quantifiers.

process on a deeper level where both semantical and syntactical actions are interwoven.[4] From the standpoint of our semantics of actions these syntactical elements are not treated as *syncategorematical*, but as first class citizens of the world of interpretable expressions.

Here are two arguments for our treatment of the syntax-semantics interface. (1) Our treatment makes incremental processing of meanings possible. (2) Treating brackets as discourse actions may help us escape from (some of) the rigidities of classical syntax. E.g. in ordinary predicate logic all quantifiers are on a single stack: the last quantifier introduced must end its scope before the earlier ones do so. In our kind of syntax we may have: $\exists x \ldots \exists y \ldots x\mathsf{E} \ldots y\mathsf{E}$, where '$x\mathsf{E}$' represents the end of scope of x. A second example: we may develop brackets of several different 'weights' as is done in Section 8.

Our main semantical operation will be composition rather than application. Thus, e.g. in our analysis of: *Jane sees Mary*, no choices about what applies to what are needed.

1.1.3. *Discourse Referents.* In the standard semantics for predicate logic meanings are sets of assignments. Since assignments essentially contain variables, variables are part of meanings. Yet, how can that be? Syntax surely cannot be made to depend on conventional choices? These cannot have a place at the level of meaning.[5] The solution to this question is that we have to make a distinction. We distinguish between the interactive surface level, the domain of conventional notational choices, and the underlying level of discourse structure. For example, consider the predicate-logical formulas $P(x)$ and $P(y)$. From one point of view, the formulas are different. E.g. there is all the difference in the world between $P(x) \wedge \neg P(y)$ and $P(x) \wedge \neg P(x)$. From another point of view these formulas are the same. The choice of x or y as variables is immaterial. It does not correspond to an 'underlying' meaning.

The variables of our semantics will be called: *discourse referents* or *discourse objects*. Their treatment will differ a bit from the usual treatment in predicate logic. First, we implement the distinction discussed above between the interactive labels (variable-names, addresses) and the underlying objects. Secondly, we exploit this distinction to create extra possibilities: the same underlying variable may have several or no names. Moreover, the names of the variable may be changed. Thirdly, discourse referents are not given beforehand as a fixed collection, but are created run-time in the unfolding of discourse.

Fusing discourse referents will be a central theme of this paper. We study only the most simple minded mechanism: two referents will be fused / identified because they share some label. Labels signal sameness. It would, of course, be more realistic to also develop fusing for contentual reasons. However, the implementation of contentual fusing is beyond the scope of the methods developed in this paper.

1.2. **Contents of the Paper.** In this paper we develop the positive fragment or, to be more precise, the $\wedge\exists$-fragment of Context Modification Logic.

In Section 2 we develop labeled sets. Labeled sets will form our machinery to implement multi-access variables. In Section 3 we will show how precisely one can

[4]Our alternative view is not unlike the way Reverse Polish Notation is interpreted in Hewlett-Packard calculators. See [Vis01].

[5]See [Fre75] and [Kap90] for comments on this point.

use labeled sets as multi-access variables. Section 4, will provide the definition of a model. In Section 5, we provide the final ingredient: relabeling. In Section 6, we put everything together and develop an example of a full positive fragment. A disadvantage of the fragment of Section 6 is that the articles are still treated in a syncategorematic way. In Section 7, we remedy this situation, making *a* and *the* first class citizens of the empire of meaning. Finally, in Section 8, we present a variation that allows us to use heavy brackets. This variation will make a reasonable treatment of the word *self* possible.

1.3. **Methods & Prerequisites.** In [VV96], Kees Vermeulen[6] and the author developed a dynamic semantics using category theoretic methods. These methods are philosophically superior, because they make the nature of the objects constructed perspicuous. However, in that paper things looked rather complicated. For example, we had a rather unwieldy garbage collection mechanism.

In the present paper, I am sidestepping a number of presentational problems of [VV96] by constructing the basic objects as normal forms modulo commutativity of term rewrite systems. The idea to use term rewrite systems emerged as a didactical device during a course on dynamic semantics for AI students.

The prerequisites for this paper are rather modest. Some knowledge of DPL or DRT would be helpful to put what is happening here into perspective. See e.g. the overview papers [MBV97] and [KvE97]. No *specific* knowledge of either DPL or DRT is needed, however. The reader needs to know a few basic facts about first order term rewriting. See [Ter03], chapter 2, by Jan Willem Klop and Roel de Vrijer.

Acknowledgements. I am grateful to Sander Bruggink, Jan van Eijck, Marcus Kracht, Kees Vermeulen, Derk Pietersen and Henk Zeevat for enlightening discussions. I am grateful to the students of various classes and of one project for their comments and critical questions. I thank Vincent van Oostrom for introducing me to term rewriting and for his keen understanding of the essentials of this project. I am grateful to Jan Bergstra for suggesting the name 'Context Modification Logic'.

2. Labeled Sets

Labeled sets or ℓ-sets are a basic tool of this paper. They realize in its simplest form the idea of an 'underlying object' that is accessible only via addresses or labels. Think of it this way. Our basic discourse objects are 'registers' or 'pegs' or 'underlying variables'. They're things with no other function than storing information in or on. As such, pegs have no further identity. Of course, in real life, these objects have to be realized somehow, but the point is that the specific realization is immaterial. We abstract away from it. This story of featureless but distinct pegs would be the whole story if we were only interested in the static perspective. However, we are interested in discourse as a process, so things should be able to interact. Thus we will equip our pegs with addresses or labels. All interaction will be mediated via these labels. The basic operation will be that two discourse objects are merged or fused or identified if they share some label.

[6]Kees Vermeulen died in 2004. His tragic death is a great loss for all who knew him.

2.1. **Labeled Sets Defined.** Here is the definition of ℓ-sets. We first give the definition and then the informal explanation. Let a set V be given. The set V will be considered as the 'space' of labels. The set V will be a hidden parameter in our definition of ℓ-set. In case we have to make it explicit, we will speak of ℓ, V-sets.

- ▷ An ℓ-set is a triple $\langle U, f, X\rangle$, where $U \subseteq V$, where X is an arbitrary set and where f is a function from U to X.
- ▷ Two ℓ-sets, $\langle U, f, X\rangle$ and $\langle W, g, Y\rangle$ are the same (isomorphic) if $U = W$ and there is a bijection $\phi : X \to Y$ such that, for all $u \in U$, $\phi(fu) = gu$, or, in other words, $\phi \circ f = g$.
- ▷ We define: $\langle U, f, X\rangle \sqsubseteq \langle W, g, Y\rangle$ iff $U \subseteq W$ and there is an injection $\phi : X \to Y$ such that, for all $u \in U$, $\phi(fu) = gu$, or, in other words, $\phi \circ f = g \circ \mathsf{emb}_{UW}$. (Here emb_{UW} is the identical embedding witnessing that $U \subseteq W$.)

The set U is the set of labels and the set X is the 'underlying' set of discourse objects. A discourse object $x \in X$ has associated to it a set of labels $f^{-1}[x] := \{u{\in}U \mid fu = x\}$. This set might be empty. We allow unlabeled elements. There are several reasons for that: (i) the theory is easier with the possibility of unlabeled elements, (ii) unlabeled elements may still store information. Note that if $x \neq x'$, then the sets of labels of x and x' are disjoint.

Our notion of sameness reflects the fact that the precise 'embodiment' of the x's is immaterial —this in contrast to the labels.

Fact 2.1. *(1)* Sameness *as defined above is an equivalence relation.*
(2) The relation $\sqsubseteq$ is a preorder, i.e. it is transitive and reflexive.
(3) Sameness (isomorphism) is the induced equivalence relation of $\sqsubseteq$.

Par abus de langage, we will use the identity sign for sameness of ℓ-sets. From the context it will always be clear whether we intend equality of tuples *qua* ℓ-set or identity *qua* underlying set-theoretical realization.

Remark 2.2. An alternative representation of ℓ-sets is as follows. We represent an ℓ-set as a pair $\langle X, F\rangle$, where $F : X \to \wp V$. We demand: $x \neq x' \Rightarrow F(x) \cap F(x') = \emptyset$.

We can get the new representation from the old one by taking (new) X to be (old) X, and by taking $F(x) := f^{-1}[x]$. Conversely, we can translate old to new by taking:

- ▷ $U := \bigcup F[X] := \{v{\in}V \mid \exists x{\in}X\ v \in F(x)\}$,
- ▷ $f(u) = x :\Leftrightarrow u \in F(x)$,
- ▷ (old) $X :=$ (new) X.

Note that we get, in the alternative format:
$\langle X, F\rangle \sqsubseteq \langle Y, G\rangle \Leftrightarrow \exists \phi{:}X{\to}Y$ (ϕ is injective and $\forall x{\in}X\ Fx \subseteq G(\phi x)$). ❑

2.2. **Term Rewrite Systems.** In this section we study finite ℓ-sets from a term rewriting perspective. To compute with these objects with some ease we will represent these sets employing a rewrite system. The basic facts on term rewriting, can be found in [Ter03].

We can build rewrite systems for finite ℓ-sets in two ways. The first way is in, what I call, *stack style* or —following a suggestion of Vincent van Oostrom— *soroidal style* ($\sigma\omega\rho\acute{o}\sigma$, *heap* or *stack*, not to be confused with $\sigma o\rho\acute{o}\sigma$, *coffin*). It is analogous to the usual treatment of the natural numbers as built up inductively

from zero and successor, and the treatment of lists in functional languages. First the definition of the terms. We have two kinds of terms: s-terms and ℓ-terms.

- ▷ $s ::= \varepsilon \mid vs \mid [s + s]$.
  Here v represents any element from V.
- ▷ $\ell ::= \Box \mid (s)\ell \mid [\ell + \ell]$.

The s's stand for sets of labels. The ℓ's stand for the ℓ-sets: each matching pair of round brackets corresponds to an element of the underlying set; the s between these brackets is the set of labels associated to the underlying element. E.g. $(xy\varepsilon)(\varepsilon)(uvw\epsilon)\Box$ is an ℓ-term. It stands for the ℓ-set: $\langle\{x, y, u, v, w\}, f, \{0, 1, 2\}\rangle$, where $f(x) = f(y) = 0$ and $f(u) = f(v) = f(w) = 2$.

Here are the rewrite rules. The variables x and y range over labels from V.

1.	$xys \to yxs$	5.	$(s)(s')\ell \to (s')(s)\ell$
2.	$xxs \to xs$	6.	$(xs)(xs')\ell \to (xs + xs')\ell$
3.	$[\varepsilon + s] \to s$	7.	$[\Box + \ell] \to \ell$
4.	$[xs + s'] \to x[s + s']$	8.	$[(s)\ell + \ell'] \to (s)[\ell + \ell']$

There is a second style of rewrite system that we may use. It is in the *monoidal* or *string style*. Here is the language.

- ▷ $s ::= \varepsilon \mid v \mid ss$.
  Here v ranges over elements of V.
- ▷ $\ell ::= \Box \mid (s) \mid \ell\ell$.

So e.g. $(xy)(\varepsilon)(uvw)$ would be an ℓ-term. It is supposed to stand for the ℓ-set of our previous example: $\langle\{x, y, u, v, w\}, f, \{0, 1, 2\}\rangle$, where $f(x) = f(y) = 0$ and $f(u) = f(v) = f(w) = 2$. These are the rewrite rules. The variables x and y range over elements of V.

1.	$xy \to yx$	5.	$(s)(s') \to (s')(s)$
2.	$xx \to x$	6.	$(xs)(xs') \to (xsxs')$
3.	$\varepsilon s \to s$	7.	$\Box\ell \to \ell$
4.	$s\varepsilon \to s$	8.	$\ell\Box \to \ell$

We will follow our second style of notations for ℓ-sets. We will write $\mathbb{I}$ for (ε). We employ the following convention: *if we construct a set theoretical representation from a normal form, we use as underlying objects 0,1,2,.... Here 0 will correspond to the first object of the normal form, etcetera.*

We will use $A, B, \ldots$ as ranging over arbitrary terms. Let's use '$\approx$' for the transitive, commutative, reflexive closure of $\to$. We can now prove, for both rewrite systems, that any term A reduces to a normal form B 'up to permutations' in the following sense: the only rules that can be applied to B are (1) or (5). Such normal forms are unique in the following sense: for any other normal form B' of A, we have: $B \approx B'$. In fact B and B' are just permutation variants of each other. It is the normal forms up to permutations of type ℓ that can be considered as direct descriptions of finite ℓ-sets. Each such form represents an ℓ-set. Conversely, for any finite ℓ-set there is a normal form up to permutations representing this ℓ-set modulo isomorphism. *From now on, we will simply use 'normal form' for: normal form up to permutations.*

We can now define, modulo isomorphism, addition of ℓ-sets. consider ℓ-sets ξ and η. Let A and B be normal forms corresponding to these sets. Then $\xi \oplus \eta$ is the ℓ-set which corresponds to the normal form of AB.

From this point on we will confuse finite ℓ-sets with their notations as normal forms in the second rewrite system.

Example 2.3. We have: $(xy)\mathbb{I}(uv)(z) \oplus (yu)(zw)\mathbb{I} = (xyuv)\mathbb{I}(zw)\mathbb{I}$. ❑

The intuition behind addition is as follows. Underlying sets X behave in an inert way under addition. Their default behavior under addition is simply that two elements are always counted different even if their accidental realizations are equal. So, if there are no labels, addition will behave like *disjoint union*. On the other hand, labels will merge whenever they have the same realization. One could say that labels *are* their realizations or that labels have *their identities on the surface*. Thus, addition on labels is ordinary union. The default behavior of underlying elements is *overruled* when their labels instruct them to fuse. This intuitive story can be made more concrete by considering how sets of underlying elements and sets of labels can be naturally embedded into ℓ-sets. This will be done in the next subsection.

Remark 2.4. Here are some notes on our term rewrite systems. First, our systems are not, strictly speaking, systems of two sorted, first order term rewriting. This is caused by our exploitation of 'notational shortcuts'. In the stack style system, we write vs instead of $\mathsf{S}_v(s)$. In this way, we can write e.g. $vwu\varepsilon$ instead of $\mathsf{S}_v(\mathsf{S}_w(\mathsf{S}_u(\varepsilon)))$. Similarly, we have written $(s)\ell$ instad of $\mathsf{S}(s,\ell)$.

In the monoidal style, we write ss', (s) and $\ell\ell'$ instead of respectively $\mathsf{c}(s,s')$, $\mathsf{i}(s)$ and $\mathsf{C}(\ell,\ell')$. We exploit the associativity of concatenation to diagrammatically represent the associativity of c and C. Thus, we do not need a rewrite rule like:

$$\mathsf{c}(s,\mathsf{c}(s',s'')) \to \mathsf{c}(\mathsf{c}(s,s'),s'').$$

Note that if we had set up our systems in the strict first order style, we should have taken our normal forms modulo commutativity *and associativity*.

The proper way of looking at our monoidal style system is as term rewriting modulo an equivalence relation. We could take this equivalence relation as generated by A (associativity) and C (commutativity) for both sorts or even as generated by A, C, U (units), I (idempotency) for both sorts. If we would use the last option, the only rewrite rule we would need to explicitely postulate would be rule nr. (6). So (6) is where the real action is. ❑

2.3. **Embeddings.** First we can consider sets of only inert underlying elements. These ℓ-sets *are*, in a sense, just their underlying set. Consider any set X. We define the ℓ-set $X^\heartsuit := \langle \emptyset, \emptyset, X\rangle$. Clearly, for any X and Y of the same cardinality, $X^\heartsuit$ and $Y^\heartsuit$ are equal. So these ℓ-sets will behave like numbers. Following the convention that $n := \{0, \cdots, n-1\}$, we have the following.

- $0^\heartsuit = \Box$, $(n+1)^\heartsuit = \overbrace{\mathbb{I}\cdots\mathbb{I}}^{n+1}$.
- $n \leq m \Leftrightarrow n^\heartsuit \sqsubseteq m^\heartsuit$.
- $(n+m)^\heartsuit = n^\heartsuit \oplus m^\heartsuit$.

Secondly, we may consider ℓ-sets posing as ordinary sets of labels. Consider any set of labels U. So, $U \subseteq V$. We define $U^\diamond := \langle U, \mathsf{id}_U, U\rangle$. Note that, modulo equality of ℓ-sets, the elements in the range of $(\cdot)^\diamond$ are those ℓ-sets $\langle U, f, X\rangle$, where f is a bijection of sets. E.g., modulo isomorphism, we have $\{u,v,w\}^\diamond = (u)(v)(w)$. We have:

- $\emptyset^\diamond = \square$.
- $U \subseteq W \Leftrightarrow U^\diamond \sqsubseteq W^\diamond$.
- $(U \cup W)^\diamond = U^\diamond \oplus W^\diamond$.

Finally, we can send all labels to one single underlying element. Let $U \subseteq V$. Define $U^\triangle := \langle U, \lambda u{\in}U \cdot 0, \{0\}\rangle$. We have e.g. $\{u, v, w\}^\triangle = (uvw)$. Note that:

- $U \subseteq W \Leftrightarrow U^\triangle \sqsubseteq W^\triangle$.
- We *do not* generally have: $(U \cup W)^\triangle = U^\triangle \oplus W^\triangle$.

3. Sorts and Contents

We want ℓ-sets as carriers of information. In this section, we will show how to store information 'on' an ℓ-set. We want our discourse referents, i.e. the underlying elements from X, to function as sorted variables. Our assignments (of objects to discourse referents) will have to respect these sorts.

Consider an ℓ-set $\xi = \langle U, h, X\rangle$. Suppose we are given a set SORT of sorts. We assign sorts to the referents of X via a function $\tau : X \to \wp(\mathsf{SORT})$. Let an inclusive domain D and a function $\delta : \mathsf{SORT} \to (\wp D \setminus \{\emptyset\})$ be given. A ξ-assignment f is a function from X to D. We say that f *respects* τ iff, for every $x \in X$, and, for every $\mathfrak{s} \in \tau(x)$, $f(x) \in \delta(\mathfrak{s})$. We will also say: f is a ξ, τ-assignment.

For example, one of the sorts could be $\mathfrak{time}$. Then $\delta(\mathfrak{time})$ would be the set of all moments. Suppose, for $x_0 \in X$, $\tau(x_0) = \{\mathfrak{time}\}$. Then ξ, τ-assignments are constrained to assign only times to x_0.

We allow overlapping sorts. This is the reason that we assign a set of sorts rather than a single sort to a referent. In discourse, a referent may acquire several sorts. E.g. in *John hates him. He throws him into the water*, the *him* would acquire the sort $\mathfrak{person}$ by being the object of *hates* and it would acquire the sort $\mathfrak{physobj}$ by being the object of *throws*.

Remark 3.1. A more beautiful approach would be to make SORT into a partial order with the following property: if two elements have a lower bound, then they have an infimum. We demand that $\mathfrak{s} \leq \mathfrak{t} \Rightarrow \delta(\mathfrak{s}) \subseteq \delta(\mathfrak{t})$. Now we may assign a single sort to a referent. If, in discourse, we are moved to assign two sorts $\mathfrak{s}$ and $\mathfrak{t}$ to the same referent, we assign the infimum of $\mathfrak{s}$ and $\mathfrak{t}$, if it exists. If the infimum does not exist —e.g. if the sorts would be $\mathfrak{time}$ and $\mathfrak{physobj}$—, the interpretation process enters the error state. ❑

Suppose τ assigns sorts to ξ and ρ assigns sorts to η. We can compute the sum of the sorted labeled sets, using the following rewrite system. First we define the language.

- $s ::= \varepsilon \mid v \mid ss$,
  where v ranges over labels,
- $u ::= \varepsilon \mid \mathfrak{s} \mid uu$,
  where $\mathfrak{s}$ ranges over sorts,
- $A ::= \varepsilon \mid (s : u) \mid AA$.

These are the rewrite rules. Let x, y range over labels and let $\mathfrak{s}, \mathfrak{t}$ range over sorts.

1.	$xy \to yx$	7.	$\varepsilon u \to u$
2.	$xx \to x$	8.	$u\varepsilon \to u$
3.	$\varepsilon s \to s$	9.	$(s : u)\,(s' : u') \to (s' : u')\,(s : u)$
4.	$s\varepsilon \to s$	10.	$(xs : u)\,(xs' : u') \to (xsxs' : uu')$
5.	$\mathfrak{st} \to \mathfrak{ts}$	11.	$\varepsilon A \to A$
6.	$\mathfrak{ss} \to \mathfrak{s}$	12.	$A\varepsilon \to A$

We can rewrite our terms to normal forms up to permutations. These normal forms represent, modulo isomorphism a pair of an ℓ-set and an assignment of sorts. We will confuse the normal forms with such pairs. To compute the sum of $\langle \xi, \tau \rangle$ and $\langle \eta, \rho \rangle$ (modulo isomorphism), pick normal forms A and B, representing sorted labeled sets isomorphic to respectively $\langle \xi, \tau \rangle$ and $\langle \eta, \rho \rangle$, compute the normal form C of AB and take the sorted ℓ-set corresponding to C.

In a similar way, we treat assignments. We employ an 'error assignment' $\surd$ to represent the fact that some terms carry conflicting information. Here is our rewrite system. First we define the language.

- $s ::= \varepsilon \mid v \mid ss$,
  where v ranges over labels,
- $u ::= \varepsilon \mid \mathfrak{s} \mid uu$,
  where $\mathfrak{s}$ ranges over sorts,
- $f ::= \varepsilon \mid \surd \mid (s : u : d) \mid ff$,
  where d ranges over objects in D.

These are the rewrite rules. Let d, d', e range over our inclusive domain; let x, y range over labels; let $\mathfrak{s}, \mathfrak{t}, \mathfrak{u}$ range over sorts. We assume $d \neq e$ and $d \notin \delta(\mathfrak{u})$.

1.	$xy \to yx$	9.	$(s : u : d)\,(s' : u' : d') \to (s' : u' : d')\,(s : u : d)$
2.	$xx \to x$	10.	$(xs : u : d)\,(xs' : u' : d) \to (xsxs' : uu' : d)$
3.	$\varepsilon s \to s$	11.	$(xs : u : d)\,(xs' : u' : e) \to \surd$
4.	$s\varepsilon \to s$	12.	$(s : \mathfrak{u}u : d) \to \surd$
5.	$\mathfrak{st} \to \mathfrak{ts}$	13.	$\varepsilon f \to f$
6.	$\mathfrak{ss} \to \mathfrak{s}$	14.	$f\varepsilon \to f$
7.	$\varepsilon u \to u$	15.	$\surd f \to \surd$
8.	$u\varepsilon \to u$	16.	$f\surd \to \surd$

As before each f-term will reduce to a normal form up to permutations. Moreover all normal forms of a given f-term are equivalent. We will confuse assignments and normal forms. We count the sorted ℓ-set $\langle \xi, \tau \rangle$ as part of the data for an assignment on ξ, τ. We define the 'sum' $f \circledast g$ as the assignment corresponding to the normal form obtained from normalizing the term fg. (Here we confuse f and g with corresponding normal forms. The result will be independent of the specific choices.) Note that clause 12 will never be used in this computation as long as we start with two non-error assignments.

Example 3.2. Let our domain consist of three objects: Sheeba, Felix, Bluto, 0 and 1. We have labels $\mathtt{sub}$, $\mathtt{ob}$ and $\mathtt{time}$. We have sorts $\mathfrak{organism}$, $\mathfrak{physobj}$ and $\mathfrak{time}$. Sheeba, Felix and Bluto are both organisms and physical objects. 0 and 1 are times. We define an assignment as follows:

- $X := \{0, 1, 2\}$,
- $U := \{\mathtt{sub}, \mathtt{ob}, \mathtt{time}\}$,

- ▷ $h(\mathtt{sub}) := 0$, $h(\mathtt{ob}) := 1$, $h(\mathtt{time}) := 2$,
- ▷ $\xi := \langle U, h, X\rangle$,
- ▷ $\tau(0) := \{\mathfrak{organism}\}$, $\tau(1) := \{\mathfrak{physobj}\}$, $\tau(2) := \{\mathfrak{time}\}$,
- ▷ $f(0) :=$ Sheeba, $f(1) :=$ Felix, $f(2) := 0$.

In rewrite notation this assignment (or, more precisely: the triple $\langle \xi, \tau, f\rangle$) would be written as:

- ▷ $(\mathtt{sub} : \mathfrak{organism} :$ Sheeba$)(\mathtt{ob} : \mathfrak{physobj} :$ Felix$)(\mathtt{time} : \mathfrak{time} : 0)$.

We will often replace the rewrite notation by a tabular notation that looks as follows.

lab	sub	ob	time
sort	organism	physobj	time
	Sheeba	Felix	0

Here the columns represent the discourse referents. ❑

A ξ, τ-content is a triple $\langle \xi, \tau, F\rangle$, where F is a set of (non-error) ξ, τ-assignments. The 'product' $\langle \xi, \tau, F\rangle \circledast \langle \eta, \rho, G\rangle$ of $\langle \xi, \tau, F\rangle$ and $\langle \eta, \rho, G\rangle$ is $\langle \theta, \sigma, H\rangle$, where θ is $\xi \oplus \eta$, where σ is the sum of τ and ρ and where:

- ▷ $H := \{f \circledast g \mid f \in F,\ g \in G$ and $f \circledast g \neq \surd\ \}$.

Note that this 'product' behaves as an intersection in case $\xi = \eta$, $\tau = \rho$ and there are no unlabeled referents. In this case we have $H = F \cap G$. In case no referent from ξ shares a label with a referent from η, the set H has an obvious 1-1 correspondence with $F \times G$.

Example 3.3. Here is an example, using an obvious tabular representation for the computation. Suppose we have a domain of five objects: Sheeba, Felix, Bluto, 0 and 1. We have three sorts $\mathfrak{organism}$, $\mathfrak{physobj}$ and $\mathfrak{time}$. We have an ℓ-content, corresponding to *sees* and an ℓ-content corresponding to *cat*. These contents are given by the following tables, say S and C. (Here these contents are simply stipulated. Later we will see how contents are assigned over structures.)

lab	sub	ob	time
sort	organism	physobj	time
	Sheeba	Felix	0
	Sheeba	Bluto	1
	Bluto	Sheeba	1

lab	ob it
sort	organism
	Sheeba
	Felix

Now we take the sum of S and C by making the table, say A, for comparing each object-row of S with each object row of C. So, A will have $6 = 3 \times 2$ object rows. Now we commute and contract the columns of the new table by following the rules of the rewrite system, eliminating rows on which we have a clash when contracting. The final result is B. Here is the table A. The yes and no's, tell us where contraction

works and where contraction produces a clash.

lab	sub	ob	time	ob it
sort	organism	physobj	time	organism
no	Sheeba	Felix	0	Sheeba
yes	Sheeba	Felix	0	Felix
no	Sheeba	Bluto	1	Sheeba
no	Sheeba	Bluto	1	Felix
yes	Bluto	Sheeba	1	Sheeba
no	Bluto	Sheeba	1	Felix

The final table B is as follows.

lab	sub	ob it	time
sort	organism	physobj organism	time
	Sheeba	Felix	0
	Bluto	Sheeba	1

Note the attractiveness of the approach sketched in remark 3.1. We would, in this approach, be allowed to contract physobj organism to organism. ❑

We end this section, by looking at the ℓ-contents on the empty ℓ-set $\Box$. There are two such contents: $\langle \Box, \emptyset, \emptyset \rangle$ and $\langle \Box, \emptyset, \{\varepsilon\} \rangle$. We trace the meanings of these mysterious looking objects. Let me remind you of the difference between the set-theoretical modeling, which provides implementation details, and *what the intended objects really are*.

The first component of our two contents is $\Box$, the empty ℓ-set. This empty ℓ-set is modeled as $\langle \emptyset, \emptyset, \emptyset \rangle$. The first component of $\Box$ stands for the empty subset of V. The third component is the empty set of referents. The intended meanings of these 'empty sets' are different, even if their modeling is the same. In a typeful language, like Haskell, these objects would be differently typed. The second component of $\Box$ is the empty function from the empty subset of V to the empty arbitrary set. The empty function really *is* a function since it assigns to *every* element of its domain a unique value: *there are no elements in its domain on which the function is either under- or over-defined*.

The second component of our two ℓ-contents is the empty assignment of sorts on the referents of $\Box$, i.e. on the empty set of arbitrary elements.

The third component of $\langle \Box, \emptyset, \emptyset \rangle$ is the empty set of assignments. Remember that a more informative piece of information yields a greater constraint on assignments and, hence, a smaller set of assignments. So, the empty set of assignments corresponds with *the most informative piece of information*, in fact with a piece that is *too* informative: absurdum. Thus $\langle \Box, \emptyset, \emptyset \rangle$ corresponds with the boolean truthvalue $\bot$. Similarly, $\langle \Box, \emptyset, \{\varepsilon\} \rangle$ is *the least informative piece of information*: *tabula rasa* or the boolean truthvalue $\top$. Note that, in our modeling, ε, the empty assignment, is equal to $\emptyset$. Note also that $\{\varepsilon\} = D^{\emptyset}$, the set of all assignments on the empty context.

For any $\xi = \langle U, h, X \rangle$, and for any assignment of sorts τ on X, we have elements $\bot_{\xi,\tau} := \langle \xi, \tau, \emptyset \rangle$ and $\top_{\xi,\tau} := \langle \xi, \tau, \prod_{x \in X} \bigcap_{\mathfrak{t} \in \tau(x)} \delta(\mathfrak{t}) \rangle$. Note that:

$$\prod_{x \in X} \bigcap_{\mathfrak{t} \in \tau(x)} \delta(\mathfrak{t}) = \{f \in D^X \mid f \text{ respects } \tau\} = \{f \mid f \text{ is a } \xi, \tau\text{-assignment}\}.$$

The miraculous multiplication of $\bot$'s and $\top$'s is due to the fact that we pursue a *more dimensional model of information growth*. E.g. $\langle \xi, \lambda x{\in}X \cdot \varepsilon, D^X \rangle$, for non-empty ξ, is, in a sense, more informed than the tabula 'rasissima' $\langle \Box, \emptyset, \{\varepsilon\} \rangle$. In terms of *contentual* information $\langle \xi, \lambda x{\in}X \cdot \varepsilon, D^X \rangle$ and $\langle \Box, \emptyset, \{\varepsilon\} \rangle$ are the same. However, $\langle \xi, \lambda x{\in}X \cdot \varepsilon, D^X \rangle$ has more storage capacity. It is more ready to receive certain contentual information than $\langle \Box, \emptyset, \{\varepsilon\} \rangle$ is.

4. Structures

In this section we introduce a modified notion of structure that corresponds naturally to the use of ℓ-sets of arguments and variables. We will first give the definition without much explanation. Then, we will illustrate the definition with an example. The best reading strategy is to first read Subsections 4.1 and 4.2 superficially. Then read Subsection 4.3, returning when needed to the definitions.

4.1. **Signatures.** A signature is a tuple $\langle \mathsf{PRED}, \mathcal{I}, \mathsf{LAB}, \mathsf{SORT}, \mathsf{ind}, \mathsf{lab}, \mathsf{sort} \rangle$. Here PRED is the set of predicate symbols. $\mathcal{I}$ is partial ordering $\langle \mathsf{IND}, \leq \rangle$.[7] The set IND is a set of indices. LAB is a function that assigns to indices sets of labels, sometimes called *label spaces*. We demand that $i \leq j \Rightarrow \mathsf{LAB}_i \subseteq \mathsf{LAB}_j$. SORT is a set of sorts. ind is a function that assigns to each predicate P in PRED an index ind_P. We write $\mathsf{Lab}_P := \mathsf{LAB}_{\mathsf{ind}(P)}$. lab is a function that assigns to each P a finite ℓ,Lab_P-set. The function sort assigns to each predicate P a function sort_P. The function sort_P assigns to each referent in lab_P a finite subset of SORT.

4.2. **Structures Defined.** A structure $\mathcal{M}$ of signature $\langle \mathsf{PRED}, \mathcal{I}, \mathsf{LAB}, \mathsf{SORT}, \mathsf{ind}, \mathsf{lab}, \mathsf{sort} \rangle$. is a triple $\langle D, \delta, I \rangle$. Here D is a non-empty set. δ assigns to every sort a subset of D. We put $D_{\mathfrak{s}} := \delta(\mathfrak{s})$. I assigns to each predicate P a set of functions on the referents of $\mathsf{lab}_P =: \langle U, h, X \rangle$ that respects the sort-information of sort_P. So:

$$I(P) \subseteq \{ f{\in}D^X \mid \forall x{\in}X \, \forall \mathfrak{s} \in \mathsf{sort}_P \; fx \in D_{\mathfrak{s}} \}.$$

Note that $\mathsf{cont}_P := \langle \mathsf{Lab}_P, \mathsf{sort}_P, I_P \rangle$ is a $\mathsf{Lab}_P, \mathsf{sort}_P$-content.

In the current treatment, we assign only 'semi-static' meanings to the predicates, and postpone the specification of relabelings —our main dynamic device— to the next stage (see Section 5). This choice is merely for the purpose of presentation. We could choose to specify directly the meanings of e.g. common nouns including their dynamic aspects. It is important to realize that in the present set-up e.g. the predicate father is not yet the common noun *father*.

4.3. **An Example.** We first specify the signature of our example. We call it $\mathcal{S}_0$. We first define $\mathcal{I}$ and LAB. All label spaces of $\mathcal{S}_0$ are subsets of a specific set of labels which is specified in terms of two other sets, viz. ARG, the set of (local) arguments (or: argument roles) and VAR, the set of (global) variables. We take:

$$\triangleright \; \Lambda_{\mathsf{ARG},\mathsf{VAR}} := \{ \langle \mathtt{arg}, \mathsf{a}, i \rangle \mid \mathsf{a} \in \mathsf{ARG}, \; i \in \omega \} \cup \{ \langle \mathtt{v}, *, i \rangle \mid \mathtt{v} \in \mathsf{VAR}, \; i \in \omega \}.$$

We assume that $\mathtt{arg} \notin \mathsf{VAR}$. We write $\mathsf{VAR}^+ := \mathsf{VAR} \cup \{\mathtt{arg}\}$. The elements of VAR^+ as first components of our labels represent the *names of the syntactic stacks* that we will employ. The second components represent the *objects present on a level of the stack*. Finally the third components represent *the levels of the stack*. Here we count the levels, starting with 0, *from top to bottom*. So level 0 is the

[7] More generally, we could take $\mathcal{I}$ to be a category.

top level, etc. The stack architecture give us the possibility to work with 'different copies of the same label', yet avoiding the danger of undesirable identifications.

Here are two convenient abbreviations: $\mathtt{a}^i := \langle \mathtt{arg}, \mathtt{a}, i\rangle$ and $\mathtt{v}^i := \langle \mathtt{v}, *, i\rangle$.

Our indices are defined as follows. Let ν be a function from VAR^+ to the natural numbers, that is 0 almost everywhere (i.e. for all but finitely many elements of the domain). Such a function can be considered as a finite multiset of elements of VAR^+. We will write '$\varnothing$' for the empty multiset and e.g. '$\lceil \mathtt{a}, \mathtt{a}, \mathtt{b}\rceil$' or '$\lceil \mathtt{a}^{(2)}, \mathtt{b}\rceil$' for the function that sends $\mathtt{a}$ to 2, $\mathtt{b}$ to 1 and the rest to 0. Now we define:

- $\Lambda^\nu := \Lambda^\nu_{\mathsf{ARG},\mathsf{VAR}} := \{\mathtt{a}^i \mid \mathtt{a} \in \mathsf{ARG},\ \mathtt{i} < \nu(\mathtt{arg})\} \cup \{\mathtt{v}^i \mid \mathtt{v} \in \mathsf{VAR},\ i < \nu(\mathtt{v})\}$.

A very compact definition would be: $\Lambda^\nu := \{\lambda{\in}\Lambda \mid (\lambda)_2 < \nu((\lambda)_0)\}$, where $(\cdot)_i$ stands for the ith projection. We take: $\mathsf{LAB}_\nu := \Lambda^\nu$.

The ordering on our indices is the multi-subset ordering:

- $\nu \leq \mu :\Leftrightarrow \forall n{\in}\omega\ \nu(n) \leq \mu(n)$.

The finite multisets over a given set X under the multi-subset ordering will be called $\mathfrak{M}_{\mathsf{fin}}X$.

We define PRED, ARG, VAR, SORT. We take:

- $\mathsf{PRED} := \{\mathsf{sees}, \mathsf{dog}, \mathsf{cat}, \mathsf{boss}, \mathsf{it}, \cdots\}$,
- $\mathsf{ARG} := \{\mathtt{sub}, \mathtt{ob}, \mathtt{in}, \cdots\}$,
- $\mathsf{VAR} := \{\mathtt{dog}, \mathtt{cat}, \mathtt{boss}, \mathtt{it}, \cdots\}$,
- $\mathsf{SORT} := \{\mathfrak{time}, \mathfrak{place}, \mathfrak{organism}, \mathfrak{physobj}, \cdots\}$,

Our domain contains such things as times, places, physical objects, organisms, etc. corresponding to these sorts. We first treat signature and semantics of the predicate sees. We take:

- $\mathsf{ind}(\mathsf{sees}) := \lceil \mathtt{arg}\rceil$.
- $\xi := \mathsf{lab}(\mathsf{sees}) := (\mathtt{sub}^0)(\mathtt{ob}^0)(\mathtt{time}^0)(\mathtt{place}^0)$,
- Let $\sigma := \mathsf{sort}_{\mathsf{sees}}$. Remember our convention according to which 0 will correspond to the first underlying object of the normal form we employed to specify ξ, etcetera.
  $\sigma_0 := \{\mathfrak{organism}\}$,
  $\sigma_1 := \{\mathfrak{physobj}\}$,
  $\sigma_2 := \{\mathfrak{time}\}$,
  $\sigma_3 := \{\mathfrak{place}\}$,
- $\delta(\mathfrak{time})$ is the set of all moments,
  $\delta(\mathfrak{place})$ is the set of all locations,
  $\delta(\mathfrak{organism})$ is the set of all organisms,
  $\delta(\mathfrak{physobj})$ is the set of al physical objects,
- $I(\mathsf{sees}) := F$, where F is the set of functions f such that $f(0)$ is an organism that sees the physical object $f(1)$ at time $f(2)$ on location $f(3)$.

So our signature information for sees can be represented as follows.

lab	$\mathtt{sub}^0$	$\mathtt{ob}^0$	$\mathtt{time}^0$	$\mathtt{place}^0$
sort	$\mathfrak{organism}$	$\mathfrak{physobj}$	$\mathfrak{time}$	$\mathfrak{place}$

We turn to the semantics of sees. Here is a concrete model DOCA, just for sees. The domain contains two organisms, Bluto, or b, and Sheeba, or s. These organisms coincide with the physical objects. The universe contains two places A and B, and two times, 0 and 1. In our model, Bluto sees Sheeba at A and 0 and Bluto and

Sheeba see each other at A and 1. So the total value of sees in this model will be as given by the table below.

lab	$\mathtt{sub}^0$	$\mathtt{ob}^0$	$\mathtt{time}^0$	$\mathtt{place}^0$
sort	organism	physobj	time	place
	b	s	0	A
	b	s	1	A
	s	b	1	A

We turn to the treatment of the predicate father.

- ▷ $\mathsf{ind}(\mathsf{father}) := \{\mathtt{arg}, \mathtt{father}, \mathtt{he}\}$.
- ▷ $\xi := \mathsf{lab}(\mathsf{father}) := (\mathtt{val}^0\, \mathtt{father}^0\, \mathtt{he}^0)(\mathtt{of}^0)$,
- ▷ Let $\sigma := \mathsf{sort}_{\mathsf{father}}$.
  $\sigma_0 := \{\mathtt{organism}\}$,
  $\sigma_1 := \{\mathtt{organism}\}$.
- ▷ $I(\mathsf{father}) := F$, where F is the set of functions f such that $f(0)$ is an organism that is father of the organism $f(1)$.

The argument val is present to implement the role father will play to form *terms*. A term has a value. How do we know that the father is that value and not the implicitly present child? Well, this is because the father is assigned the value role val.

Following a suggestion of Bernhard Fisseni we could make the presence of the variable child implicit in the signature of father. This would make it possible to mimic a discourse in which, after a father is introduced, one refers back to 'the child' without any child being explicitly mentioned before. So father à la Fisseni would look as follows:

- ▷ $\mathsf{ind}(\mathsf{father}) := \{\mathtt{arg}, \mathtt{father}, \mathtt{he}, \mathtt{child}\}$.
- ▷ $\mathsf{lab}(\mathsf{father}) := (\mathtt{val}^0\, \mathtt{father}^0\, \mathtt{he}^0)(\mathtt{of}^0\, \mathtt{child}^0)$.

As Fisseni points out, many uses of *child* cannot be given the purely relational meaning, since they carry the extra information *young*. We will not go into this interesting problem here.

One way to get more control of anaphoric reference using the definite description *the father* is to employ predicates father^i, where:

- ▷ $\mathsf{ind}(\mathsf{father}^i) := \{\mathtt{arg}, \mathtt{father}^{(i+1)}, \mathtt{he}\}$,
- ▷ $\mathsf{lab}(\mathsf{father}^i) := (\mathtt{val}^0\, \mathtt{father}^i\, \mathtt{he}^0)(\mathtt{of}^0)$.

Our next example is the predicate angry. We take:

- ▷ $\mathsf{ind}(\mathsf{angry}) := \{\mathtt{arg}\}$.
- ▷ $\xi := \mathsf{lab}(\mathsf{angry}) := (\mathtt{val}^0)$,
- ▷ Let $\sigma := \mathsf{sort}_{\mathsf{angry}}$.
  $\sigma_0 := \{\mathtt{organism}\}$.
- ▷ $I(\mathsf{angry}) := F$, where F is the set of functions f such that $f(0)$ is an angry organism.

Next we treat the predicates he and he^i. These predicates will, of course, be used for anaphoric reference.

- ▷ $\mathsf{ind}(\mathsf{he}^i) := \{\mathtt{arg}, \mathtt{he}^{(i+1)}\}$.
- ▷ $\xi := \mathsf{lab}(\mathsf{he}^i) := (\mathtt{val}^0\, \mathtt{he}^i)$,
- ▷ Let $\sigma := \mathsf{sort}_{\mathsf{he}^i}$.
  $\sigma_0 := \{\mathtt{organism}\}$.

- ▷ $I(\mathsf{he}^i) := F$, where F is the set of functions f such that $f(0)$ is an organism.
- ▷ $\mathsf{he} := \mathsf{he}^0$.

The choice of organisms for the range of he seems rather doubtful. Maybe we should have chosen simply all entities. For she and it we can give similar definitions. Note that we only implement the idea of 'syntactical gender'. E.g. a referent introduced by a use of *father* can be picked up by a use of *he* rather than *she* or *it*. We could also add constraints for 'semantical gender', e.g. by demanding that the possible values of he are male organisms.

Next we treat the predicate john. We will treat john as a free variable, very much analogous to he.

- ▷ $\mathsf{ind}(\mathsf{john}) := \lceil \mathtt{arg}, \mathtt{john}, \mathtt{he} \rceil$.
- ▷ $\xi := \mathsf{lab}(\mathsf{john}) := (\mathtt{val}^0\, \mathtt{john}^0\, \mathtt{he}^0)$,
- ▷ Let $\sigma := \mathsf{sort}_{\mathsf{john}}$.
  $\sigma_0 := \{\mathfrak{person}\}$.
- ▷ $I(\mathsf{john}) := F$, where F is a set of functions f such that $f(0)$ is a person.

We could also make predicates john^i analogous to he^i. We might wish to add the constraint that John must be male.

Prepositions play a rather special role in our system. They function as the link between values of terms and arguments of terms and verbs. sub is such a preposition, which functions in a way analogous to the nominative case.

- ▷ $\mathsf{ind}(\mathsf{sub}) := \lceil \mathtt{arg}^{(2)} \rceil$.
- ▷ $\xi := \mathsf{lab}(\mathsf{sub}) := (\mathtt{val}^0\, \mathtt{sub}^1)$,
- ▷ Let $\sigma := \mathsf{sort}_{\mathsf{sub}}$.
  $\sigma_0 := \emptyset$.
- ▷ $I(\mathsf{sub}) := F$, where F is the set of functions f such that $f(0)$ is an entity in D.

The idea is that sub links the *value* of the current term level with the *subject* occurring on the underlying sentence level. Similar definitions can be given for ob, with, etc. ....

A special predicate is who that will be used to interpret such sentences as *the man, who ...*. This will be parsed somewhat like $\langle\langle$*the-man* sub $\langle\langle$*who*. Here the third bracket signals a sentence embedded in the term level. So we have the term level 0 on which *who* occurs, the underlying sentence level 1 and the term level 2 in which the sentence of level 1 is embedded. who's business is to link the value of level 0 to the value of level 2. Thus we define:

- ▷ $\mathsf{ind}(\mathsf{who}) := \lceil \mathtt{arg}^{(3)} \rceil$.
- ▷ $\xi := \mathsf{lab}(\mathsf{who}) := (\mathtt{val}^0\, \mathtt{val}^2)$,
- ▷ Let $\sigma := \mathsf{sort}_{\mathsf{val}}$.
  $\sigma_0 := \{\mathfrak{organism}\}$.
- ▷ $I(\mathsf{sub}) := F$, where F is the set of functions f such that $f(0)$ is an organism.

Perhaps we should have chosen the empty set of sorts for who.

5. Relabeling

Syntactic structure and, more generally, discourse structure will exercise their effects via *relabeling*. In this section we develop the theory of relabeling.

5.1. **Injective Relabeling.** There are good reasons to consider *injective* relabeling first. Nearly all relabelings we are interested in are injective. Moreover, injective relabeling is much simpler than the general case.

Consider two spaces of labels V and V'. Let $\xi = \langle U, h, X\rangle$ be an ℓ,V-set. Let E be a partial injective function from V to V'. We can relabel ξ via E to the ℓ, V'-set $\xi' =: \xi \star E$ as follows:

- ▷ $\xi' := \langle E[U], h \circ E^{-1}, X\rangle$.
  Here $E[U] = \{E(u) \mid u \in U\}$.

Example 5.1. Suppose $V = \{a, b, c\}$ and $V' = \{d, e\}$. Let $E(a) = d$, $E(b) = e$ and let $E(c)$ be undefined. We have: $(ab)(c)\mathbb{I} \star E = (de)(\varepsilon)\mathbb{I} = (de)\mathbb{III}$ and $(ac)(b) \star E = (d)(e)$.

Suppose that $W = \{a\}$, $W' = \emptyset$ and that E is the empty mapping from W to W'. We obviously have: $(a) \star E = \mathbb{I}$. Note that we do *not* have: $(a)(a) \star E = \mathbb{III}$. We can only use E as instruction to replace labels when we have a normal form. ❑

Let E be a partial injection from V to V' and let E' be a partial injection from V' to V''. Note that:

- ▷ $\xi \star (E; E') = (\xi \star E) \star E'$.
  Here ';' stands for composition in the order of writing.
- ▷ The ℓ-set $(\xi \oplus \eta) \star E$ need not be equal to $(\xi \star E) \oplus (\eta \star E)$.
- ▷ Suppose that ξ has n referents. Then $\xi \star \emptyset = n^{\heartsuit}$.
- ▷ $n^{\heartsuit} \star E = n^{\heartsuit}$.

We can lift the operation $(\cdot)\star E$ in the obvious way to sorted ℓ-sets and to ℓ-contents: we take $\langle \xi, \tau\rangle \star E := \langle \xi \star E, \tau\rangle$ and $\langle \xi, \tau, F\rangle \star E := \langle \xi \star E, \tau, F\rangle$.

5.2. **Special Injective Relabelings.** We will employ special injective relabelings on the Λ^{ν}. Let the sum of two multisets (on a given domain) be given as follows: $(\nu + \mu)(a) := \nu(a) + \mu(a)$.

We define, for a in VAR^{+}, the partial bijection $\mathsf{S}_{\mathsf{a}} : \Lambda^{\nu} \to \Lambda^{\nu + \lceil \mathsf{a} \rceil}$, as follows:

- ▷ $\mathsf{S}_{\mathsf{a}}(\langle \mathsf{b}, \mathsf{c}, i\rangle) := \begin{cases} \langle \mathsf{b}, \mathsf{c}, i+1\rangle & \text{if } \mathsf{b} = \mathsf{a} \\ \langle \mathsf{b}, \mathsf{c}, i\rangle & \text{if } \mathsf{b} \neq \mathsf{a} \end{cases}$

Here c is in ARG or $\mathsf{c} = *$, depending on what b is. The relabelings we employ in our simplest fragment will be generated by relabelings of the form S_{a} and $\mathsf{S}_{\mathsf{a}}^{-1}$. Here, we take, $\mathsf{S}_{\mathsf{a}}^{-1} : \Lambda^{\nu + \lceil \mathsf{a} \rceil} \to \Lambda^{\nu}$.

There is an aspect of 'polymorphism' to S_{a}, since it is defined uniformly for different ν.[8]

We can directly compute the effect of repeated applications of the S_{a} and $\mathsf{S}_{\mathsf{b}}^{-1}$. See appendix A.1.

5.3. **Arbitrary Relabeling.** Arbitrary relabelings will be only used in Section 7. So temporarily skipping this subsection is a reasonable option.

Consider two spaces of labels V and V'. Let $\xi = \langle U, h, X\rangle$ be an ℓ,V-set. Let R be a relation between V to V'. We say that R is *of finite degree* iff, for each $v \in V$, $\{v' \in V' \mid vRv'\}$ is finite and, for each $v' \in V$, $\{v \in V \mid vRv'\}$ is finite.

[8]This aspect can be captured using the notion of (lax) natural transformation. Closer study of this point is beyond the scope if these notes.

Suppose that R is of finite degree. We can relabel ξ via R to the ℓ, V'-set $\xi' =: \xi \star R$ as follows. Pick a normal form representation $\breve{\xi}$ of ξ. For every finite subset X of V', we introduce a label $\overline{X}$. We arrange that the elements of V' are disjoint from the $\overline{X}$'s. Now we form a term $\tilde{\xi}$ by replacing every label v in $\breve{\xi}$ by $\overline{\{v' \in V' \mid vRv'\}}$. We reduce $\tilde{\xi}$ in the term rewriting system for ℓ-sets with as labels the elements of V' plus the $\overline{X}$'s. We restrict rule (10) concerning the identification of referents to labels from V'. We add the extra rules:

$$\boxed{\begin{array}{l}\overline{X} \to x\,\overline{X \setminus \{x\}}, \text{ for } x \in X \\ \overline{\emptyset} \to \varepsilon\end{array}}$$

We reduce $\tilde{\xi}$ to normal form modulo permutations. It is not hard to see that, modulo permutations, this normal form is independent of the choice of $\breve{\xi}$. the normal form we obtain represents an ℓ, V'-set, which is $\xi \star R$.

Example 5.2. Suppose $V = \{a, b, c\}$ and $V' = \{d, e, j\}$ and that R is given by: aRd, aRe, bRe, cRe, cRf. We have: $(ab)(c)\mathbb{I} \star R = (de)(ef)\mathbb{I} = (def)\mathbb{I}$ and $(ac)(b) \star R = (def)(e) = (def)$. ❑

Example 5.3. Suppose $V = \{a, b\}$, $V' = \{c\}$ and $V'' = \emptyset$. Suppose R between V and V' is given by: aRc, bRc. Let R' be the empty relation from V' to V''. We have:

- $((a)(b) \star R) \star R' = (c)(c) \star R' = (c) \star R' = \mathbb{I}$,
- $(a)(b) \star (R; R') = (a)(b) \star \emptyset = \mathbb{III}$.

The example illustrates that we do not always have $(\xi \star R) \star R' = \xi \star (R; R')$. ❑

We can lift the operation $(\cdot)\star R$ in the obvious way to sorted ℓ-sets and to ℓ-contents by using a fully analogous procedure.

6. Transitions over a Model

Finally, we are at the point where we can define our dynamic semantics. A coherent package of such assignments of transition systems to models will be called a context modification logic.

6.1. **What Meanings Are.** A dynamic meaning will be a transition between information states. A state will have the following form: $\langle i, \xi, \tau, F\rangle$, where $i \in \mathsf{IND}$ and $\alpha := \langle \xi, \tau, F\rangle$ is an ℓ, LAB_i-content. We will also write states as: $\langle i, \alpha\rangle$.

A special and very important state is $\mho := \langle \varnothing, \Box, \emptyset, \{\varepsilon\}\rangle$. Here the first component is the empty multiset of elements of VAR^+. Note that $\Lambda^{\varnothing} = \emptyset$. Moreover, $\langle \Box, \emptyset, \{\varepsilon\}\rangle$ is the $\ell, \emptyset$-content tabula rasa or $\top$. Thus $\mho$ is the tabula rasa state. $\mho$ is, so to speak, the mere possibility of discourse, the state obtaining before the first word is spoken.

6.2. **Description of a Context Modification Logic.** In this subsection, we will describe context modification logic and simultaneously provide the example CML_0. The signature of CML_0 is the signature $\mathcal{S}_0$, that was introduced in Subsection 4.3.

Consider a state $\sigma = \langle i, \alpha\rangle$ and consider a predicate P of the signature of the model. We define:

- $\sigma[P] := \begin{cases} \langle i, \alpha \circledast \mathsf{cont}_P\rangle & \text{if } \mathsf{ind}_P \leq i \\ \text{undefined} & \text{otherwise} \end{cases}$

Here $\mathsf{cont}_P = \langle \mathsf{lab}_P, \mathsf{sort}_P, I_P \rangle$, the ℓ, Lab_P-content assigned to P by the model. We 'lift' this content to the, possibly, wider label space LAB_i by simply considering the same $\langle \mathsf{lab}_P, \mathsf{sort}_P, I_P \rangle$ as an ℓ, LAB_i-content.[9] If the index of the state does not extend ind_P, which provides *the minimum label space necessary for the transition*, then our action becomes undefined. Thus, ind_P functions as a *presupposition* for updating. (Note that our indices may carry more 'desiderata for definedness' than just those given by the associated label spaces.)

There is a second set of possible transitions given by a set of *relabeling symbols* RELAB. In the case of CML_0, we take:

▷ $\mathsf{RELAB} = \{\langle \mathtt{a}|, |\mathtt{a}\rangle \mid \mathtt{a} \in \mathsf{VAR}^+\}$.

We will often write "⟨" for: $\langle \mathtt{arg}|$, and "⟩" for: $|\mathtt{arg}\rangle$. For $\mathtt{v} \in \mathsf{VAR}$, we write "$\lceil_{\mathtt{v}}$" for: $\langle \mathtt{v}|$, and "$_{\mathtt{v}}\rceil$" for: $|\mathtt{v}\rangle$.

The logic will assign to any symbol $A \in \mathsf{RELAB}$ an indexed set of relabelings, J_A, in the following way. For $i \in I$, then $J_A(i)$, if defined, is an index j and $J_{A,i} : \mathsf{LAB}_i \to \mathsf{LAB}_j$ is a relabeling. We ask that, if $i \leq i'$ and if $J_A(i)$ is defined, then $J_A(i')$ is defined, $J_A(i) \leq J_A(i')$ and $J_{A,i'}$ extends $J_{A,i}$.

In the case of CML_0, we take:

▷ $J_{\langle \mathtt{a}|}(\nu) := \nu + \lceil \mathtt{a} \rceil$,

▷ $J_{\langle \mathtt{a}|,\nu} := \mathsf{S}_{\mathtt{a}} : \Lambda^{\nu} \to \Lambda^{\nu + \lceil \mathtt{a} \rceil}$,

▷ $J_{|\mathtt{a}\rangle}(\mu) := \begin{cases} \nu & \text{if } \mu = \nu + \lceil \mathtt{a} \rceil \\ \text{undefined} & \text{otherwise} \end{cases}$,

▷ $J_{|\mathtt{a}\rangle, \nu + \lceil \mathtt{a} \rceil} := \mathsf{S}_{\mathtt{a}}^{-1} : \Lambda^{\nu + \lceil \mathtt{a} \rceil} \to \Lambda^{\nu}$.

Let σ be $\langle i, \alpha \rangle$ and let $\alpha := \langle \xi, \tau, F \rangle$. With a relabeling symbol A, we associate a transition as follows:

▷ $\sigma[A] := \begin{cases} \langle J_A(i), \alpha \star J_{A,i} \rangle & \text{if } J_A(i) \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases}$

We assume that PRED and RELAB are disjoint. Let ϕ and ψ be strings of elements of $\mathsf{PRED} \cup \mathsf{RELAB}$. We define:

▷ $\sigma[\varepsilon] = \sigma$, where ε is used for the the empty string,

▷ $\sigma[\phi\,\psi] := \sigma[\phi][\psi]$.

Remark 6.1. Any of the transitions $[\phi]$ defined above maps a state $\langle i, \langle \xi, \tau, F \rangle \rangle$ to a state $\langle j, \langle \eta, \nu, G \rangle \rangle$. Note that j, η and ν only depend on i, ξ and τ. Thus, if we keep i, ξ, τ fixed, $[\phi]$ gives us a map $\Phi = \Phi_{i,\xi,\tau}$ which sends the F's to the G's. It is not difficult to see that our semantics is *distributive* in the following sense. Consider any i, ξ, τ. Let $\Phi := \Phi_{i,\xi,\tau}$. Let $\mathcal{F}$ be an set of sets of assignments for ξ. We have:

▷ $\Phi(\bigcup \mathcal{F}) = \bigcup \Phi[\mathcal{F}]$ $\;(= \bigcup\{\Phi(F) \mid F \in \mathcal{F}\})$.

Using the fact that our semantics is distributive, we can associate a relational semantics to it. This means that all the information of Φ is contained in the following relation $R := R_{\Phi}$:

▷ $f R g :\Leftrightarrow g \in \Phi(\{f\})$.

We can recover Φ from R, since $\Phi(F) = [F]R := \{g \mid \exists f \in F\ f R g\}$. Moreover, if Ψ is the map associated with $[\psi]$ on j, η, ν, we have: $R_{\Phi;\Psi} = R_{\Phi}; R_{\Psi}$.

[9] This means that there is a notationally suppressed embedding functor here.

Consider the transition $[\phi]$. Let R be the associated relation for i, ξ, τ . Consider $\sigma := \langle i, \langle \xi, \tau, F \rangle \rangle$. We can define $\sigma \models \phi$ as: $F \subseteq \mathsf{dom}(R)$. The disadvantage of this definition is that it only works for our restricted fragment where all transitions are distributive. For further comments on this point, see Section 9. ❑

6.3. **An Example.** Natural language expressions will be assigned certain translations in our logic. Before discussing these translations in a more comprehensive way, we first treat an example in some detail. Here are three translations.

- ▷ *a-father* := $\lceil_{\mathtt{father}} \lceil_{\mathtt{he}} \mathsf{father}$,
- ▷ *a-dog* := $\lceil_{\mathtt{dog}} \lceil_{\mathtt{it}} \mathsf{dog}$,
- ▷ *sees* := sees.

Note that we treat the indefinite article syncategorematically. In Section 7, we will discuss how to switch to a categorematic treatment.

Let us consider a structure. The domain consists of John, Dan, Paula, Peter, Eve, Pluto, Bonzo, Felix, 0, 1, A, B. The numbers 0, 1 stand for times and A, B stand for locations. Let's suppose the only fathers in the domain are John, father of Paula and Peter, and Dan, father of Eve. The dogs are Pluto and Bonzo. The seeing relation is completely specified by: John's sees Pluto at time 0 at place A and Eve at time 1 and place B; Dan sees Bonzo and Felix both at time 0 and location B; Felix sees every object except himself at every time and every location.

We compute the action of $\langle \langle$*a-father* $\mathsf{sub} \rangle$ *sees* $\langle$*a-dog* $\mathsf{ob} \rangle \rangle$ on the tabula rasa state $\mathfrak{V}$.

$\mathfrak{V}[\,\langle\,] = \langle\, \lceil \mathtt{arg} \rfloor, \Box, \emptyset, \{\varepsilon\} \rangle$
$\mathfrak{V}[\,\langle\langle\,] = \langle\, \lceil \mathtt{arg}^{(2)} \rfloor, \Box, \emptyset, \{\varepsilon\} \rangle$
$\mathfrak{V}[\,\langle\langle \lceil_{\mathtt{father}}\,] = \langle\, \lceil \mathtt{arg}^{(2)}, \mathtt{father} \rfloor, \Box, \emptyset, \{\varepsilon\} \rangle$
$\mathfrak{V}[\,\langle\langle \lceil_{\mathtt{father}} \lceil_{\mathtt{he}}\,] = \langle\, \lceil \mathtt{arg}^{(2)}, \mathtt{father}, \mathtt{he} \rfloor, \Box, \emptyset, \{\varepsilon\} \rangle$

So nothing really happens. We just 'load' the context, so that the new files or referents can be meaningfully created.

$\mathfrak{V}[\,\langle\langle \lceil_{\mathtt{father}} \lceil_{\mathtt{he}} \mathsf{father}\,] = \langle\, \lceil \mathtt{arg}^{(2)}, \mathtt{father}, \mathtt{he} \rfloor, \alpha_0 \rangle$.
Here α_0 is given by the following table:

$\mathtt{val}^0\,\mathtt{father}^0\,\mathtt{he}^0$	$\mathtt{of}^0$
organism	organism
John	Paula
John	Peter
Dan	Eve

Note that it is the combined effect of the relabelings and the predicate that causes *new referents* for father and child to be created.

$\mathfrak{V}[\,\langle\langle \lceil_{\mathtt{father}} \lceil_{\mathtt{he}} \mathsf{father}\ \mathsf{sub}\,] = \langle\, \lceil \mathtt{arg}^{(2)}, \mathtt{father}, \mathtt{he} \rfloor, \alpha_1 \rangle$.
Here α_1 is given by the following table:

$\mathtt{val}^0\,\mathtt{sub}^1\,\mathtt{father}^0\,\mathtt{he}^0$	$\mathtt{of}^0$
organism	organism
John	Paula
John	Peter
Dan	Eve

The role val has done its work. It is thrown away in the next step.

$\mathfrak{V}[\,\langle\langle\lceil_{\texttt{father}}\lceil_{\texttt{he}}\mathsf{father\ sub}\rangle\,] = \langle\,[\texttt{arg}, \texttt{father}, \texttt{he}], \alpha_2\rangle$.
Here α_2 is given by the following table:

$\texttt{sub}^0\,\texttt{father}^0\,\texttt{he}^0$	
organism	organism
John	Paula
John	Peter
Dan	Eve

$\mathfrak{V}[\,\langle\langle\lceil_{\texttt{father}}\lceil_{\texttt{he}}\mathsf{father\ sub}\rangle\ \mathsf{sees}\,] = \langle\,[\texttt{arg}, \texttt{father}, \texttt{he}], \alpha_3\rangle$.
Here α_3 is given by the following table:

$\texttt{sub}^0\,\texttt{father}^0\,\texttt{he}^0$		$\texttt{ob}^0$	$\texttt{time}^0$	$\texttt{place}^0$
organism	organism	physobj	time	place
John	Paula	Pluto	0	A
John	Paula	Eve	1	B
John	Peter	Pluto	0	A
John	Peter	Eve	1	B
Dan	Eve	Bonzo	0	B
Dan	Eve	Felix	0	B

We skip a few steps.

$\mathfrak{V}[\,\langle\langle\lceil_{\texttt{father}}\lceil_{\texttt{he}}\mathsf{father\ sub}\rangle\ \mathsf{sees}\ \langle\,\lceil_{\texttt{dog}}\lceil_{\texttt{it}}\mathsf{dog}\,] =$
$\langle\,[\texttt{arg}^{(2)}, \texttt{father}, \texttt{dog}, \texttt{he}, \texttt{it}], \alpha_4\rangle$.
Here α_4 is given by the following table:

$\texttt{sub}^1\,\texttt{father}^0\,\texttt{he}^0$		$\texttt{ob}^1$	$\texttt{time}^1$	$\texttt{place}^1$	$\texttt{val}^0\,\texttt{dog}^0\,\texttt{it}^0$
organism	organism	physobj	time	place	organism
John	Paula	Pluto	0	A	Pluto
John	Paula	Pluto	0	A	Bonzo
John	Paula	Eve	1	B	Pluto
John	Paula	Eve	1	B	Bonzo
John	Peter	Pluto	0	A	Pluto
John	Peter	Pluto	0	A	Bonzo
John	Peter	Eve	1	B	Pluto
John	Peter	Eve	1	B	Bonzo
Dan	Eve	Bonzo	0	B	Pluto
Dan	Eve	Bonzo	0	B	Bonzo
Dan	Eve	Felix	0	B	Pluto
Dan	Eve	Felix	0	B	Bonzo

$\mathfrak{V}[\,\langle\langle\lceil_{\texttt{father}}\lceil_{\texttt{he}}\mathsf{father\ sub}\rangle\ \mathsf{sees}\ \langle\,\lceil_{\texttt{dog}}\lceil_{\texttt{it}}\mathsf{dog\ ob}\,] =$
$\langle\,[\texttt{arg}^{(2)}, \texttt{father}, \texttt{dog}, \texttt{he}, \texttt{it}], \alpha_5\rangle$.

Here α_5 is given by the following table:

$\mathtt{sub}^1\,\mathtt{father}^0\,\mathtt{he}^0$		$\mathtt{ob}^1\,\mathtt{val}^0\,\mathtt{dog}^0\,\mathtt{it}^0$	$\mathtt{time}^1$	$\mathtt{place}^1$
organism	organism	physobj	time	place
John	Paula	Pluto	0	A
John	Peter	Pluto	0	A
Dan	Eve	Bonzo	0	B

$\mathcal{U}[\,\langle\langle\lceil_{\mathtt{father}}\lceil_{\mathtt{he}}\mathsf{father\ sub}\rangle\ \mathsf{sees}\ \langle\,\lceil_{\mathtt{dog}}\lceil_{\mathtt{it}}\mathsf{dog\ ob}\rangle\,]\,=$
$\langle\,[\mathtt{arg},\mathtt{father},\mathtt{dog},\mathtt{he},\mathtt{it}],\alpha_6\rangle$.
Here α_6 is given by the following table:

$\mathtt{sub}^0\,\mathtt{father}^0\,\mathtt{he}^0$		$\mathtt{ob}^0\,\mathtt{dog}^0\,\mathtt{it}^0$	$\mathtt{time}^0$	$\mathtt{place}^0$
organism	organism	physobj	time	place
John	Paula	Pluto	0	A
John	Peter	Pluto	0	A
Dan	Eve	Bonzo	0	B

$\mathcal{U}[\,\langle\langle\lceil_{\mathtt{father}}\lceil_{\mathtt{he}}\mathsf{father\ sub}\rangle\ \mathsf{sees}\ \langle\,\lceil_{\mathtt{dog}}\lceil_{\mathtt{it}}\mathsf{dog\ ob}\rangle\rangle\,]\,=\,\langle\,[\mathtt{father},\mathtt{dog},\mathtt{he},\mathtt{it}],\alpha_7\rangle$.
Here α_7 is given by the following table:

$\mathtt{father}^0\,\mathtt{he}^0$		$\mathtt{dog}^0\,\mathtt{it}^0$		
organism	organism	physobj	time	place
John	Paula	Pluto	0	A
John	Peter	Pluto	0	A
Dan	Eve	Bonzo	0	B

So, after processing the sentence we are left with the long-range 'variables' $\mathtt{father}^0$, $\mathtt{he}^0$, $\mathtt{dog}^0$ and $\mathtt{it}^0$.

Note that it would be an option to delete the unlabeled columns. However, we then loose the feature of steady information growth.

6.4. **Translations.** In this subsection we provide some translations of natural language expressions. It is important to realize that the language we are translating natural language expressions into is a language *without syntax*: every string of atomic symbols is well-formed. Moreover, every such string will correspond to a transition. Of course, not every transition is interesting or sensible ....

6.4.1. *Indefinite Articles.* The indefinite article was already discussed in Subsection 6.3. We will return to it in Section 7.

6.4.2. *Personal Pronouns.* The simplest approach to translate the pronoun *he* is as follows.

- ▷ $he := \mathsf{he}$,
- ▷ $he^i := \mathsf{he}^i$,
- ▷ $his := \langle \mathsf{of}\ he\rangle$.

We treat *she* and *it* in a similar way. Here is the paraphrase of: *A father sees a dog. He kicks it.*

$$\langle\langle a\text{-}father\ \mathsf{sub}\rangle\ sees\ \langle a\text{-}dog\ \mathsf{ob}\rangle\rangle\ \langle\langle he\ \mathsf{sub}\rangle\ kicks\ \langle it\ \mathsf{ob}\rangle\rangle.$$

In the paraphrase *he* and *it* will co-refer with respectively *a-father* and *a-dog*. Here is another example: *A father shouts at his dog*. We may paraphrase this discourse as:

$$\langle\langle \textit{a-father}\ \mathsf{sub}\rangle\ \textit{shouts}\ \langle \mathsf{at}\ \textit{his a-dog}\rangle\rangle.$$

In our example we have to give dog an 'owner argument' `of`. This state of affairs is somewhat unsatisfactory, since it would make the proposition *that a dog has an owner*, a logical truth. Clearly, we need some notion of 'optional argument'. We will not pursue this matter further in this paper. Secondly, we treated the utterance *his dog* as introducing a dog. My intuition is that it is in reality more like an anaphor. One could try to give it a more anaphoric effect by assuming that every father has a dog and by introducing a referent for a dog with an utterance of *father*. That does not seem to be such a plausible solution. It is more as if *a father* introduces the possibility of dog-ownership in an optional way. The option is activated as soon as we say *his dog*. Again I will not pursue this matter in this paper.

It may be argued that uses of *he* affect the salience pattern. After we picked up a referent using he, that referent is a more likely candidate to be picked up by a subsequent use of *he*. We could model this idea using an alternative translation and a new predicate he^i_* as follows.

- ▷ $\mathsf{ind}(\mathsf{he}^i_*) := \{\mathtt{arg}, \mathtt{he}^{(i+2)}\}$.
- ▷ $\xi := \mathsf{lab}(\mathsf{he}^i_*) := (\mathtt{val}^0\,\mathtt{he}^0\,\mathtt{he}^{i+1})$,
- ▷ Let $\sigma := \mathsf{sort}_{\mathsf{he}^i_*}$.
  $\sigma_0 := \{\mathfrak{organism}\}$.
- ▷ $I(\mathsf{he}^i_*) := F$, where F is the set of functions f such that $f(0)$ is an organism.
- ▷ $he^i := \lceil_{\mathtt{he}}\,\mathsf{he}^i_*$,
- ▷ $he := he^0$.

6.4.3. *Names.* The treatment of names is analogous to the treatment of pronouns. One could say that we treat names as a special kind of pronouns. The simplest approach is as follows.

- ▷ $John^i := \lceil_{\mathtt{he}}\,\mathsf{john}^i$,
- ▷ $John := John^0$.

Again, me may wish to model the fact that a use of a name makes it more salient, not only for being picked up by a pronoun but also for being picked up by a repeated use of the name. Here is the modified theory.

- ▷ $\mathsf{ind}(\mathsf{john}^i_*) := \{\mathtt{arg}, \mathtt{john}^{i+1}, \mathtt{he}\}$.
- ▷ $\xi := \mathsf{lab}(\mathsf{john}^i_*) := (\mathtt{val}^0\,\mathtt{john}^0\,\mathtt{john}^{i+1}\,\mathtt{he}^0)$,
- ▷ Let $\sigma := \mathsf{sort}_{\mathsf{he}^i_*}$.
  $\sigma_0 := \{\mathfrak{male\ person}\}$.
- ▷ $I(\mathsf{john}^i_*) := F$, where F is the set of functions f such that $f(0)$ is an male person.
- ▷ $John^i := \lceil_{\mathtt{john}}\lceil_{\mathtt{he}}\,\mathsf{john}^i_*$,
- ▷ $John := John^0$.

6.4.4. *Relative Pronouns.* Our paraphrase for *who* is as follows.

- ▷ *who* := who.

Here is a sample paraphrase.

$$\langle\langle \textit{a-father}\ \mathsf{sub}\ \langle\langle \textit{who}\ \mathsf{sub}\rangle\ \textit{sees}\ \langle \textit{a-dog}\ \mathsf{ob}\rangle\rangle\rangle\ \textit{kicks}\ \langle \textit{it}\ \mathsf{ob}\rangle\rangle.$$

Note that we can only mimic *nonrestrictive relative clauses* in this way. To get the effect of a *restrictive relative clause* we would need to have a fusing mechanism for referents that also can employ contentual information.

Our semantics of *who* does not work too well, since the term *who* can be a strict subterm of a term of the embedded sentence. Consider for example: *A man, who's mother enters the room, sees a dog* and *John, a book about whom I read, has a nice house*. The semantics for *who* of this subsection would get the linking wrong. In Section 8, we develop a non-ad-hoc solution to this problem.

6.4.5. *Prepositions.* We treat e.g. the preposition *with*, as follows.

- ▷ $\mathsf{ind}(\mathsf{with}) := [\mathtt{arg}^{(2)}]$.
- ▷ $\xi := \mathsf{lab}(\mathsf{with}) := (\mathtt{val}^0\,\mathtt{with}^1)$,
- ▷ Let $\sigma := \mathsf{sort}_{\mathsf{with}}$.
  $\sigma_0 := \emptyset$.
- ▷ $I(\mathsf{with}) := F$, where F is the set of functions f such that $f(0)$ is an entity in D.
- ▷ *with* := with.

To demonstrate its use let's consider the verb *cuts*.

- ▷ $\mathsf{ind}(\mathsf{cuts}) := [\mathtt{arg}]$.
- ▷ $\xi := \mathsf{lab}(\mathsf{cuts}) := (\mathtt{sub}^0)(\mathtt{ob}^0)(\mathtt{with}^0)(\mathtt{time}^0)(\mathtt{place}^0)$,
- ▷ Let $\sigma := \mathsf{sort}_{\mathsf{cuts}}$. Remember our convention according to which 0 will correspond to the first underlying object of the normal form we employed to specify ξ, etcetera.
  $\sigma_0 := \{\mathfrak{organism}\}$,
  $\sigma_1 := \{\mathfrak{physobj}\}$,
  $\sigma_2 := \{\mathfrak{physobj}\}$,
  $\sigma_3 := \{\mathfrak{time}\}$,
  $\sigma_4 := \{\mathfrak{place}\}$,
- ▷ $\delta(\mathfrak{time})$ is the set of all moments,
  $\delta(\mathfrak{place})$ is the set of all locations,
  $\delta(\mathfrak{organism})$ is the set of all organisms,
  $\delta(\mathfrak{physobj})$ is the set of al physical objects,
- ▷ $I(\mathsf{cuts}) := F$, where F is the set of functions f such that $f(0)$ is an organism that cuts the physical object $f(1)$ with the fysical object $f(2)$ at time $f(3)$ on location $f(4)$.
- ▷ *cuts* := cuts.

We can make the following paraphrases.

(1) $\langle\langle \textit{John}\ \mathsf{sub}\rangle\ \textit{cuts}\ \langle \textit{a hamburger}\ \mathsf{ob}\rangle\rangle$.
(2) $\langle\langle \textit{John}\ \mathsf{sub}\rangle\ \textit{cuts}\ \langle \textit{a hamburger}\ \mathsf{ob}\rangle\ \langle \textit{with a knife}\rangle\rangle$.

Arguments that are omitted, will be implicitly existentially quantified. So the first paraphrase will mean roughly: John cuts a hamburger with something. Since we have no notion of obligatory versus optional arguments, in our set-up, we can leave out the subject as well. E.g.

$$\langle \textit{cuts}\ \langle \textit{with a knife}\rangle\rangle.$$

will mean: someone cuts something with a knife.

6.4.6. *Adjectives.* Adjectives are easy. We treat them as follows.

▷ *angry* := angry.

Here are two sample paraphrases:

$$\langle\langle \textit{he}\ \mathsf{sub}\rangle\ \mathsf{is}\ \langle \textit{angry}\ \mathsf{ob}\rangle\rangle \quad \text{and} \quad \langle\langle \textit{angry a-man}\ \mathsf{sub}\rangle\ \mathsf{comes\text{-}in}\rangle$$

We have to put *angry* before or after *a-man*. We will be ale to use the English word order *a angry man* as soon as we have developed the categorematic treatment of the articles. See Section 7.

6.4.7. *The Definite Article.* One way to treat the definite article syncategorematically is as follows.

▷ $\mathsf{ind}(\mathsf{father}^i) := \lceil \mathtt{arg}, \mathtt{father}^{i+1}, \mathtt{he} \rfloor$.
▷ $\xi := \mathsf{lab}(\mathsf{father}^i) := (\mathtt{val}^0\ \mathtt{father}^i\ \mathtt{he}^0)(\mathtt{of}^0)$,
▷ Let $\sigma := \mathsf{sort}_{\mathsf{father}^i}$.
$\sigma_0 := \{\mathfrak{organism}\}$,
$\sigma_1 := \{\mathfrak{organism}\}$.
▷ $I(\mathsf{father}^i) := F$, where F is the set of functions f such that $f(0)$ is an organism that is father of the organism $f(1)$.
▷ *the*^{i}*-father* := $\lceil_{\mathtt{he}}\ \mathsf{father}^i$

We might wish to make the present use of father salient to be picked up by subsequent uses of father. This can be done as follows.

▷ $\mathsf{ind}(\mathsf{father}^i_*) := \lceil \mathtt{arg}, \mathtt{father}^{i+1}, \mathtt{he} \rfloor$.
▷ $\xi := \mathsf{lab}(\mathsf{father}^i_*) := (\mathtt{val}^0\ \mathtt{father}^0\ \mathtt{father}^i\ \mathtt{he}^0)(\mathtt{of}^0)$,
▷ Let $\sigma := \mathsf{sort}_{\mathsf{father}^i_*}$.
$\sigma_0 := \{\mathfrak{organism}\}$,
$\sigma_1 := \{\mathfrak{organism}\}$.
▷ $I(\mathsf{father}^i_*) := F$, where F is the set of functions f such that $f(0)$ is an organism that is father of the organism $f(1)$.
▷ *the*^{i}*-father* := $\lceil_{\mathtt{father}}\ \lceil_{\mathtt{he}}\ \mathsf{father}^{i+1}_*$.
▷ *a-father* := $\lceil_{\mathtt{father}}\ \lceil_{\mathtt{he}}\ \mathsf{father}^0_*$.

For a categorematic treatment of the definite article, see Section 7.

6.4.8. *One Sentence on Top of Another.* In CML_0 we can drop one sentence on top of another. Here is an example: *a father —he is angry— shouts at his dog.* We may paraphrase this discourse as:

$$\langle\langle \textit{a-father}\ \mathsf{sub}\ \langle\langle \textit{he}\ \mathsf{sub}\rangle\ \textit{is}\ \langle \textit{angry}\ \mathsf{ob}\rangle\rangle\rangle\ \textit{shouts}\ \langle \mathsf{at}\ \textit{his a-dog}\rangle\rangle.$$

6.4.9. *Lists.* A child visits a children's farm. Walking around, it utters *donkey duck rabbit peacock.* Such a list can be paraphrased as follows.

$$\langle \textit{a-donkey}\rangle\ \langle \textit{a-duck}\rangle\ \langle \textit{a-rabbit}\rangle\ \langle \textit{a-peacock}\rangle.$$

Note that the paraphrase of the list is perfectly meaningful *but not sentential.* The child merely notes the presence of these animals. A disadvantage of our paraphrase is that the aspect of pointing (*deixis*) is not well modeled, we only get the existential claim.

The following would report the existence of a monster (or, alternatively, be simply false).

$$\langle \textit{a-donkey a-duck a-rabbit a-peacock} \rangle.$$

7. Categorematic Treatment of Articles

We want to design a categorematic treatment of the articles. This treatment should follow the overall style of our semantics. The obvious idea is that in saying *a* or *the*, we create a 'local discourse referent'. This local referent carries the information that it was created by an utterance of *a* or *the*. As soon as the noun is made public the local referent is transformed into the appropriate global referent corresponding to the noun. The present treatment embodies a mechanism for 'raising the salience' of the noun.

We choose a designated variable $\mathtt{local}$. The desired 'transformation into a global referent' is implemented as follows. For any $\mathtt{v} \in \mathsf{VAR}$, we define a relabeling $\mathsf{H}_\mathtt{v}$. Consider $\langle \mathsf{a}, \mathsf{b}, \mathsf{c} \rangle \in \Lambda$. We take:

$$\triangleright\ \mathsf{H}_\mathtt{v}(\langle \mathsf{a}, \mathsf{b}, \mathsf{c} \rangle) := \begin{cases} \langle \mathtt{v}, *, i \rangle & \text{if } \langle \mathsf{a}, \mathsf{b}, \mathsf{c} \rangle = \langle \mathtt{local}, *, i \rangle \\ \langle \mathsf{a}, \mathsf{b}, \mathsf{c} \rangle & \text{otherwise} \end{cases}.$$

The function H is the only example of non-injective relabeling in this paper. We now define: $\mathsf{RELAB} := \{\lceil_{\mathtt{a},\mathtt{a}}\rceil \mid \mathtt{a} \in \mathsf{VAR}^+\} \cup \{\downarrow_\mathtt{v} \mid \mathtt{v} \in \mathsf{VAR}\}$. Let ν be an index. We write $\nu[\mathtt{local} := 0]$ for the result of resetting the $\mathtt{local}$-value of ν to zero. We define J on the brackets as before and we take:

$$\triangleright\ J_{\downarrow_\mathtt{v}}(\nu) = \begin{cases} \nu[\mathtt{local} : 0] & \text{if } \nu(\mathtt{local}) \leq \nu(v) \\ \text{undefined} & \text{otherwise} \end{cases},$$

$$\triangleright\ J_{\downarrow_\mathtt{v},\nu} = \mathsf{H}_v : \Lambda^\nu \to \Lambda^{\nu[\mathtt{local}:=0]}.$$

We define a new predicate ana^i. We take:

- $\triangleright\ \mathsf{ind}(\mathsf{ana}^i) := \lceil\!\lceil \mathtt{arg}, \mathtt{loc}^{(i+1)} \rceil\!\rceil$.
- $\triangleright\ \xi := \mathsf{lab}(\mathsf{ana}^i) := (\mathtt{val}^0\, \mathtt{local}^0\, \mathtt{local}^i)$,
- $\triangleright$ Let $\sigma := \mathsf{sort}_\mathsf{ana}$.
  $\sigma_0 := \emptyset$.
- $\triangleright\ I(\mathsf{ana}^i) := F$, where F is the set of all functions f from $\{0\}$ to objects of some domain.

We now take:

- $\triangleright\ \mathit{ana}^i := \lceil^{i+1}_\mathtt{local}\, \mathsf{ana}^i$.

Here $\lceil^{i+1}_\mathtt{local}$ is the $i+1$-fold iteration of $\lceil_\mathtt{local}$. Let: $a := \mathit{ana}^0$, $\mathit{the} := \mathit{ana}^1$, $\mathit{the}^i := \mathit{ana}^{i+1}$. We interpret e.g. *man* as follows:

- $\triangleright\ \lceil_\mathtt{man} \lceil_\mathtt{he} \mathsf{man} \downarrow_\mathtt{man}$.

The reader may amuse herself by inventing variants of the present approach. Evidently the way we are doing it is not always optimally efficient.

In appendix **??**, we give a brief sketch of a philosophy of naming that employs the same ideas as our implementation of the categorematic articles.

8. Heavy Brackets

Usually syntax is construed, *syncategorematically*, i.e. as something that helps us determine meaning but that has no independent meaning. In our framework, syntax is construed as meaningful. This move creates room to build non-standard

syntax. In this section we explore one possible direction of 'loosening up' the rigid constraints of standard syntax.

To introduce the idea, let's stare at an example for a moment. *Mary's mother's brother eats a doughnut.* In CML_0 we would paraphrase this as:

$\langle\langle\langle\langle \mathit{Mary}\ \mathsf{of}\rangle\ \mathit{a\text{-}mother}\ \mathsf{of}\rangle\ \mathit{a\text{-}brother}\ \mathsf{sub}\rangle\ \mathsf{eats}\ \langle \mathit{a\text{-}doughnut}\ \mathsf{ob}\rangle\rangle$

Or:

$\langle\langle\langle\langle\lceil_{\mathtt{she}}\mathsf{mary}\ \mathsf{of}\rangle\ \lceil_{\mathtt{mother}}\lceil_{\mathtt{she}}\mathsf{mother}\ \mathsf{of}\rangle\ \lceil_{\mathtt{brother}}\lceil_{\mathtt{he}}\mathsf{brother}\ \mathsf{sub}\rangle$
$\mathsf{eats}\ \langle\lceil_{\mathtt{doughnut}}\lceil_{\mathtt{it}}\mathsf{doughnut}\ \mathsf{ob}\rangle\rangle$

Eventually we would like to have an incremental parser to go with our incremental semantics. It would seem, however, that we would have to backtrack to get the four opening brackets in place. The idea we pursue here is to have one 'bracket' that means *let there be sufficiently many term levels to store subsequent term information.* Our paraphrase in the new set-up will look as follows.

$[\{\mathit{Mary}\ \mathsf{of}\rangle\ \mathit{a\text{-}mother}\ \mathsf{of}\rangle\ \mathit{a\text{-}brother}\ \mathsf{sub}\}\ \mathsf{eats}\ \{\mathit{a\text{-}doughnut}\ \mathsf{ob}\}]$

Or:

$[\{\lceil_{\mathtt{she}}\mathsf{mary}\ \mathsf{of}\rangle\ \lceil_{\mathtt{mother}}\lceil_{\mathtt{she}}\mathsf{mother}\ \mathsf{of}\rangle\ \lceil_{\mathtt{brother}}\lceil_{\mathtt{he}}\mathsf{brother}\ \mathsf{sub}\}$
$\mathsf{eats}\ \{\lceil_{\mathtt{doughnut}}\lceil_{\mathtt{it}}\mathsf{doughnut}\ \mathsf{ob}\}]$

Here '[' opens the sentence level. Inside the sentence level we are allowed to push infinitely many term levels in one full sweep, using the heavy bracket '{'. The subsequent pops of the form '⟩' are balanced by the original '{'. The closing bracket '}' pops infinitely many term levels and brings us back to the sentence level.

Here is the definition of CML_1, the logic in which heavy brackets are implemented.[10] The sets VAR, VAR^+ are as before. We employ two sets of arguments: `s-ARG`, the sentential arguments, and `t-ARG` the term arguments. E.g. `sub` and `ob` are in `s-ARG`. `val` is in `t-ARG`. The preposition `with` could be both in `s-ARG` and in `t-ARG`.

In what follows 'ω^2' will stand for *the ordinal* ω^2 and not the Cartesian product $\omega \times \omega$. We define auxiliary functions $\mathsf{Arg_s} : \omega^2 \to \{\mathsf{s\text{-}ARG}, \mathsf{t\text{-}ARG}\}$ and $\mathsf{Arg_t} : \omega^2 \to \{\mathsf{s\text{-}ARG}, \mathsf{t\text{-}ARG}\}$ as follows:

$$\triangleright\ \mathsf{Arg}_{\mathsf{s},\alpha} = \begin{cases} \mathsf{s\text{-}ARG} & \text{if } \alpha = \omega \cdot n \text{ for some } n \in \omega \\ \mathsf{t\text{-}ARG} & \text{otherwise} \end{cases},$$

$$\triangleright\ \mathsf{Arg}_{\mathsf{t},\alpha} = \begin{cases} \mathsf{s\text{-}ARG} & \text{if } \alpha = \omega \cdot n \text{ for some } n \in \omega \setminus \{0\} \\ \mathsf{t\text{-}ARG} & \text{otherwise} \end{cases}.$$

The set $\mathsf{Arg}_{\mathsf{s},\alpha}$ gives us the arguments that are appropriate in case the top level 0 is a sentence level. In this case, the levels 0 and ω^n, for $n > 0$, are sentence levels. All other levels are term levels. The set $\mathsf{Arg}_{\mathsf{t},\alpha}$ gives us the arguments that are appropriate in case the top level 0 is a term level. Note that $\mathsf{Arg}_{\mathsf{s},\alpha}$ and $\mathsf{Arg}_{\mathsf{t},\alpha}$ only differ at level 0.

We employ functions ν that send the elements of VAR to natural numbers and that send `arg` to an ordinal below ω^2. We call such functions, ad hoc, *a*-multisets. We will persist in using multiset notation for *a*-multisets. Our indices will be pairs $\langle \nu, \mathsf{x}\rangle$, where ν is an a-multiset and $\mathsf{x} \in \{\mathsf{s}, \mathsf{t}\}$. Our ordering on the indices becomes:

[10] The basic idea for this logic was originally developed by Kees Vermeulen and myself in response to a suggestion of Henk Zeevat to treat the point at the end of a sentence as 'full stop'. But, if you have a full stop, then there also must be a 'full start' ...

- $\langle \nu, \mathsf{x}\rangle \leq \langle \mu, \mathsf{y}\rangle :\Leftrightarrow \nu \leq \mu$ and $\mathsf{x} = \mathsf{y}$.

Here are the sets of labels that we assign to our indices.

- $\mathsf{LAB}_{\langle \nu,\mathsf{x}\rangle} := \Theta^{\nu,\mathsf{x}} := \{\langle \mathtt{v}, *, i\rangle \mid \mathtt{v} \in \mathsf{VAR} \text{ and } i < \nu(\mathtt{v})\} \cup \{\langle \mathtt{arg}, \mathsf{a}, \alpha\rangle \mid \mathsf{a} \in \mathsf{Arg}_{\mathsf{x},\alpha} \text{ and } \alpha < \nu(\mathtt{arg})\}$.

For $\mathtt{v} \in \mathsf{VAR}$, we define:

- $\mathsf{S}_{\mathtt{v}}(\langle \mathsf{x}, \mathsf{y}, \beta\rangle) := \begin{cases} \langle \mathsf{x}, \mathsf{y}, 1+\beta\rangle & \text{if } \mathsf{x} = \mathtt{v} \\ \langle \mathsf{x}, \mathsf{y}, \beta\rangle & \text{otherwise} \end{cases}$.

We define further, for $\alpha \in \{1, \omega\}$:

- $\mathsf{S}^{(\alpha)}_{\mathtt{arg}}(\langle \mathsf{x}, \mathsf{y}, \beta\rangle) := \begin{cases} \langle \mathsf{x}, \mathsf{y}, \alpha+\beta\rangle & \text{if } \mathsf{x} = \mathtt{arg} \\ \langle \mathsf{x}, \mathsf{y}, \beta\rangle & \text{otherwise} \end{cases}$.

Note that the operation $+$ on ω^2 is not commutative, so the order of addition is important. The function $\mathsf{S}^{(\alpha)}_{\mathtt{arg}}$ is injective. The reason that we use $\alpha + (\cdot)$ rather than $(\cdot) + \alpha$ is that we are pushing α things on a stack, *where we take 0 to be the top of the stack*.

The set RELAB contains $\langle \mathtt{v}|$ and $|\mathtt{v}\rangle$, for $\mathtt{v} \in \mathsf{VAR}$, and $\langle \alpha, \mathsf{xy}|$, $|\alpha, \mathsf{xy}\rangle$, for $\langle \alpha, \mathsf{xy}\rangle \in \{\langle 1, \mathtt{tt}\rangle, \langle 1, \mathtt{ts}\rangle, \langle \omega, \mathtt{st}\rangle\}$. We take:

- $\lceil_{\mathtt{v}} := \langle \mathtt{v}|$ and $\rceil_{\mathtt{v}} := |\mathtt{v}\rangle$,
- $\langle\!\langle := \langle 1, \mathtt{tt}|$ and $\rangle\!\rangle := |1, \mathtt{tt}\rangle$,
- $[:= \langle 1, \mathtt{ts}|$ and $] := |1, \mathtt{ts}\rangle$,
- $\{ := \langle \omega, \mathtt{st}|$ and $\} := |\omega, \mathtt{st}\rangle$.

We define:

- $J_{\langle \mathtt{v}|}(\langle \nu, \mathsf{x}\rangle) := \langle\!\langle \lceil \mathtt{v} \rceil + \nu, \mathsf{x}\rangle$,
- $J_{\langle \mathtt{v}|, \langle \nu, \mathsf{x}\rangle} := \mathsf{S}_{\mathtt{v}} : \Theta^{\nu,\mathsf{x}} \to \Theta^{\lceil \mathtt{v} \rceil + \nu, \mathsf{x}}$,
- $J_{|\mathtt{v}\rangle}(\langle \mu, \mathsf{x}\rangle) := \begin{cases} \langle \nu, \mathsf{x}\rangle & \text{if } \mu = \lceil \mathtt{v} \rceil + \nu \\ \text{undefined} & \text{otherwise} \end{cases}$,
- $J_{|\mathtt{v}\rangle, \langle\!\langle \lceil \mathtt{v} \rceil + \nu, \mathsf{x}\rangle} := \mathsf{S}^{-1}_{\mathtt{v}} : \Theta^{\lceil \mathtt{v} \rceil + \nu, \mathsf{x}} \to \Theta^{\nu,\mathsf{x}}$,
- $J_{|\alpha,\mathsf{xy}\rangle}(\langle \nu, \mathsf{z}\rangle) := \begin{cases} \langle\!\langle \lceil \mathtt{arg}^{(\alpha)} \rceil + \nu, \mathsf{y}\rangle & \text{if } \mathsf{z} = \mathsf{x} \\ \text{undefined} & \text{otherwise} \end{cases}$,
- $J_{\langle \alpha,\mathsf{xy}|, \langle \nu,\mathsf{x}\rangle} := \mathsf{S}^{(\alpha)}_{\mathtt{arg}} : \Theta^{\nu,\mathsf{x}} \to \Theta^{\lceil \mathtt{arg}^{(\alpha)} \rceil + \nu, \mathsf{y}}$,
- $J_{|\alpha,\mathsf{xy}\rangle}(\langle \mu, \mathsf{z}\rangle) := \begin{cases} \langle \nu, \mathsf{x}\rangle & \text{if } \mu = \lceil \mathtt{arg}^{(\alpha)} \rceil + \nu \text{ and } \mathsf{z} = \mathsf{y} \\ \text{undefined} & \text{otherwise} \end{cases}$,
- $J_{|\alpha,\mathsf{xy}\rangle, \langle \lceil \mathtt{arg}^{(\alpha)} \rceil + \nu, \mathsf{y}\rangle} := \mathsf{S}^{(\alpha),-1}_{\mathtt{arg}} : \Theta^{\lceil \mathtt{arg} \rceil^{(\alpha)} + \nu, \mathsf{y}} \to \Theta^{\nu,\mathsf{x}}$.

A small verification of the correctness of the above definitions is necessary: we have to verify that term levels are always mapped to term levels and sentence levels to sentence levels. This would go wrong if we would also allow e.g. $\langle \omega, \mathtt{tt}|$ and $|\omega, \mathtt{tt}\rangle$.

The treatment of models is as before. There are now two tabula rasa states. The first is the empty sentential state: $\langle\langle \varnothing, \mathsf{s}\rangle, \Box, \emptyset, \{\emptyset\}\rangle$, the second is the empty term state: $\mho := \langle\langle \varnothing, \mathsf{t}\rangle, \Box, \emptyset, \{\emptyset\}\rangle$. We will only work with $\mho$, taking it as the basic underlying discourse state. We take e.g.

- $\mathsf{ind}(\mathsf{sees}) := \langle\!\langle \lceil \mathtt{arg} \rceil, \mathsf{s}\rangle$ and $\mathsf{lab}(\mathsf{sees}) := (\mathtt{sub}^0\, \mathtt{ob}^0)$,
- $\mathsf{ind}(\mathsf{man}) := \langle\!\langle \lceil \mathtt{arg}, \mathtt{man}, \mathtt{he} \rceil, \mathsf{t}\rangle$ and $\mathsf{lab}(\mathsf{man}) := (\mathtt{val}^0\, \mathtt{man}^0\, \mathtt{he}^0)$,
- $\mathsf{ind}(\mathsf{sub}) := \langle\!\langle \lceil \mathtt{arg}^{(\omega+1)} \rceil, \mathsf{t}\rangle$ and $\mathsf{lab}(\mathsf{sub}) := (\mathtt{val}^0\, \mathtt{sub}^{\omega})$.
- $\mathsf{ind}(\mathsf{who}) := \langle\!\langle \lceil \mathtt{arg}^{(\omega+2)} \rceil, \mathsf{t}\rangle$ and $\mathsf{lab}(\mathsf{who}) := (\mathtt{val}^0\, \mathtt{val}^{\omega+1})$.

So sub links the value at the top term level with the subject at the underlying sentence level and who links the value at the top term level with the value of the term level directly below the first underlying sentence level. Here is a paraphrase.

$$[\{\textit{the-man}\ \mathsf{sub}\ [\{\textit{a-umbrella}\ \langle \mathsf{of}\ \mathsf{who}\}\ \textit{is}\ \{\textit{nice}\ \mathsf{ob}\}]\}\ \textit{is}\ \{\textit{angry}\ \mathsf{ob}\}]$$

We take *self* to be a new word, reminiscent of the Dutch *zichzelf* and the German *sichselbst*, that always means the subject of the closest underlying sentence level. Thus, we interpret *self* as sub. Now we can paraphrase: *John looks at a photo of a painting of self*, as:

$$[\{\textit{John}\ \mathsf{sub}\}\ \textit{looks}\ \{\textit{at}\ \textit{a-photo}\ \langle \textit{of}\ \textit{a-painting}\ \langle \textit{of}\ \textit{self}\}]$$

Here we take $\textit{at} := \mathsf{at}$ and $\textit{of} := \mathsf{of}$.

9. Perspectives

The main ingredient that is lacking from this paper is the treatment of validity, negation and implication. (As in DPL, universal quantification will simply be a special case of implication.) There is one obvious way of extending our positive fragments. Simply take over the definitions of negation and implication from DPL, using the relational form of our semantics described in remark 6.1. Regrettably, this definition gets the phenomena wrong. E.g. if we consider a use of a name in the antecedent of an implication, then the effect of the 'salience quantifier' associated with the name will be canceled by the fact that implications are conditions. (This phenomenon is well known from the treatment of names in DRT.) What we need is a notion of negation and implication that is more dynamic than the one of DPL. For a better definition see the masters thesis [Pie01].

The machinery as we developed it, has no explicit device to trace the descendent, if any, in the output state of a transition of a given referent in the input state. There are various ways to add such a device. Enriching the semantics systematically with such a device, will make our transitions into natural transformations (w.r.t. suitably chosen categories and functors); for every transition we have a mapping from discourse referents to discourse referents in a sufficiently uniform way. Such an enrichment is useful for the definition of the semantics of negation, implication and validity. Also we need such an enrichment if we want to extend the semantics with Frank Veltman's treatment of *maybe*. In this case, we loose distributivity, which necessitates a somewhat different style of definition of validity and implication. (See [GSV96], for the classical integration of DPL and Veltman's Update Semantics.)

Along another line, we may wish to try to fit CML into a computational paradigm that is 'close in spirit'. See the masters thesis [Bg01], in which CML is developed in graph rewriting style.

It is rather obvious how to build a DRT version of CML. We use the actions associated to the predicate symbols to produce relational databases instead of sets of assignments. See also [Bg01].

Well, these are only the most direct perspectives. On a longer term we might wish to integrate our system with:

(1) A Reichenbach style treatment of time;
(2) Plurals;
(3) Generalized Quantifiers;
(4) Contentual mechanisms of fusing;
(5) ...

References

[Bg01] S. Bruggink. *Discourse Representation by Hypergraphs*. Number 5 in CKI masters thesis series. CKI, Utrecht, `http://preprints.phil.uu.nl/scripties/`, November 2001.

[Fre75] G. Frege. Über Sinn und Bedeutung. In *Funktion, Begriff, Bedeutung*, pages 40–65. Vandenhoeck and Ruprecht, 1975. reprinted in e.g. [**?**, pp. 142–160].

[GSV96] J. Groenendijk, M. Stokhof, and F. Veltman. Coreference and modality. In S. Lappin, editor, *Handbook of Contemporary Semantic Theory*, pages 179–213. Blackwell, Oxford, 1996.

[Kap90] David Kaplan. Words. *Aristotelian Society*, Supp. 64:93–119, 1990.

[KvE97] H. Kamp and J. van Eijck. Representing discourse in context. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 179–237. Elsevier, Amsterdam & MIT Press, Cambridge, 1997.

[MBV97] R. Muskens, J. van Benthem, and A. Visser. Dynamics. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 587–648. Elsevier, Amsterdam & MIT Press, Cambridge, 1997.

[Pie01] D. Pietersen. *Dynamics and Negation, investigations on extending CML with a negative fragment*. Number 6 in CKI masters thesis series. CKI, Utrecht, `http://preprints.phil.uu.nl/scripties/`, December 2001.

[Ter03] Terese. *Term Rewriting Systems*. Cambridge University Press, Cambridge, 2003.

[Vis01] A. Visser. On the ambiguation of Polish notation. Artificial Intelligence Preprint Series 26, Department of Philosophy, Utrecht University, Heidelberglaan 8, 3584 CS Utrecht, `http://preprints.phil.uu.nl/aips/`, July 2001.

[VV96] A. Visser and C. Vermeulen. Dynamic bracketing and discourse representation. *Notre Dame Journal of Formal Logic*, 37:321–365, 1996.

Appendix A. Stack-numbers

In this appendix we compute the effect of repeated compositions of certain relabelings.

A.1. **Stack-numbers on Natural Numbers.** In this subsection, we compute the repeated compositions of the S_a and the $\mathsf{S}_\mathsf{b}^{-1}$ introduced in Subsection 5.2. We need an auxiliary definition. Cutoff-substraction between natural numbers is defined as follows.

$$\triangleright\ n \mathbin{\dot{-}} m := \max(0, n-m) = \begin{cases} n-m & \text{if } m \leq n \\ 0 & \text{otherwise} \end{cases}.$$

We have:

$\triangleright\ (a+b) \mathbin{\dot{-}} c = (a \mathbin{\dot{-}} (c \mathbin{\dot{-}} b)) + (b \mathbin{\dot{-}} c)$,
$\triangleright\ a \mathbin{\dot{-}} (b+c) = (a \mathbin{\dot{-}} b) \mathbin{\dot{-}} c$,
$\triangleright\ a + (b \mathbin{\dot{-}} a) = b + (a \mathbin{\dot{-}} b) = \max(a, b)$.

A *stack-number* is a pair $\langle n, m\rangle$, where $n, m \in \omega$. We will write $(n \succ m)$ for $\langle n, m\rangle$. The variables $\alpha, \beta, \ldots$ will range over stack-numbers.

The intuition behind a stack number $(n \succ m)$ is that n stands for n pops from an incoming stack and m stands for m subsequent pushes.

Addition or composition of stack-numbers is defined as follows:

$$\triangleright\ (q \succ p) + (n \succ m) = (q + (n \mathbin{\dot{-}} p)) \succ (m + (p \mathbin{\dot{-}} n))$$

We take $\mathbb{O} := (0 \succ 0)$. We have:

$\triangleright\ \mathbb{O} + \alpha = \alpha + \mathbb{O} = \alpha$,
$\triangleright\ +$ on stack-numbers is associative.

Stack-numbers can be represented by a rewrite system as follows. Here is the language.

$$\triangleright\ B ::= \varepsilon \mid \langle \mid \rangle \mid BB.$$

So a term is e.g. $\langle\rangle\rangle\varepsilon\rangle\varepsilon\langle\langle\rangle$. These are the rewrite rules.

1.	$\varepsilon B \to B$	3.	$\langle\ \rangle \to \varepsilon$
2.	$B\varepsilon \to B$		

Clearly, in this system we have normal forms of the form

$$\overbrace{\rangle\cdots\rangle}^{n\times}\ \overbrace{\langle\cdots\langle}^{m\times}$$

for $m+n>0$, or we have a normal form ε. These forms represent respectively the stack-numbers $n \succ m$ and $0 \succ 0$. The rewrite system shows that stack-numbers are the numbers relevant for the usual bracket balance test.

Let S be the successor function on the natural numbers. We define:

- $\mathsf{S}^{(n\succ m)}(k) = p :\Leftrightarrow \exists s{\in}\omega\ k = n+s$ and $p = m+s$.

We have:

- $\mathsf{S}^{\mathbb{O}} = \mathsf{id}_\omega$,
- $\mathsf{S}^{(0\succ 1)} = \mathsf{S}$,
- $\mathsf{S}^{(1\succ 0)} = \mathsf{S}^{-1}$,
- $\mathsf{S}^{\alpha};\mathsf{S}^{\beta} = \mathsf{S}^{\alpha+\beta}$,
- $\mathsf{S}^{(n\succ m)} = S^{-n};S^{m}$.

Let μ be a function from VAR^+ to stack-numbers. We assume that μ is almost everywhere $\mathbb{O}$. We define S^μ as the iterated composition of all $\mathsf{S}_\mathsf{a}^\alpha$ for $\mu(\mathsf{a}) = \alpha$. It is easy to see that the value this composition does not depend on the order of composing. We find that any iterated product of S_a's and $\mathsf{S}_\mathsf{b}^{-1}$'s is of the form S^μ.

A.2. **Stack-numbers on Ordinals.** We can build stack-numbers on ordinals in full analogy of the stack-numbers on natural numbers. We first have to generalize the notion of cutoff-subtraction $\dot{-}$. The relevant observation is that cutoff-subtraction is fully determined on the natural numbers by the equation:

$$k \leq m+n \Leftrightarrow k \mathbin{\dot{-}} m \leq n.$$

This equation tells us that $k \mathbin{\dot{-}} m$ is the minimum of $\{n \mid k \leq m+n\}$. The existence of this minimum is guaranteed: any non-empty set of natural numbers has a minimum.[11]

We can easily generalize this idea to ordinals. Consider any weakly monotonic function R, either on a limit ordinal λ or on the class of all ordinals ON. Suppose that R is cofinal, i.e. for every $\alpha \in \lambda(\mathsf{ON})$, there is a $\beta \in \lambda(\mathsf{ON})$, such that $\alpha \leq R(\beta)$. Now define $L(\alpha) := \min\{\beta{\in}\lambda(\mathsf{ON}) \mid \alpha \leq R(\beta)\}$. Then we have:

$$\alpha \leq R(\beta) \Leftrightarrow L(\alpha) \leq \beta.$$

Applying this insight to post- and to pre-additon with γ, we find the existence of two cut-off substractions or residuals:

- $\alpha \leq \beta+\gamma \Leftrightarrow (\alpha \mathbin{/} \gamma) \leq \beta$,
- $\alpha \leq \gamma+\beta \Leftrightarrow (\gamma \mathbin{\backslash} \alpha) \leq \beta$.

[11] For those who know some category theory: the equation states that $L_m := \lambda k{\in}\omega \cdot k \mathbin{\dot{-}} m$ is the left adjoint of $R_m := \lambda n{\in}\omega \cdot m+n$. L_m and R_m are here considered as endofunctors of $\langle\omega,\leq\rangle$ considered as a preorder category.

We have two different residuals, since addition on the ordinals in not commutative. E.g. $(\omega + 1) \,/\, \omega = \omega$ and $\omega \setminus (\omega + 1) = 1$. Ordinals have the following important property: for any $\gamma \leq \alpha$, there is a *unique* β_0, such that $\gamma + \beta_0 = \alpha$. It is easy to see that:

$$\gamma \setminus \alpha = \begin{cases} \text{the unique } \beta_0 \text{ such that } \gamma + \beta_0 = \alpha & \text{if } \gamma \leq \alpha \\ 0 & \text{otherwise} \end{cases} .$$

So we have, for $\gamma \leq \alpha$, that $\gamma + (\gamma \setminus \alpha) = \alpha$. Note, in contrast, that

$$((\omega + 1) \,/\, \omega) + \omega = \omega + \omega = \omega \cdot 2 > \omega + 1.$$

So $\setminus$ is, in a sense, the better form of cut-off subtraction. We will concentrate on $\setminus$.

We can extend $(\cdot) \setminus (\cdot)$ to ordinal valued multisets in the obvious way. Using this extension we can e.g. reformulate the clause for $[\alpha, \mathsf{xy}\rangle$ as:

$$\triangleright \; J_{[\alpha,\mathsf{xy}\rangle}(\langle \mu, \mathsf{z}\rangle) := \begin{cases} \langle \, \lceil \mathtt{arg}^{(\alpha)} \rceil \setminus \mu, \mathsf{x} \rangle & \text{if } \lceil \mathtt{arg}^{(\alpha)} \rceil \leq \mu \text{ and } \mathsf{z} = \mathsf{y} \\ \text{undefined} & \text{otherwise} \end{cases} ,$$

We define stack-numbers on λ (ON) as pairs $\alpha' \succ \alpha$. To make sense of the definition of c-addition of stack-numbers below, we must assume that λ is closed under ordinal addition. The operation c-addition or $\breve{+}$ on stack-numbers is defined as follows:

$$\triangleright \; (\alpha' \succ \alpha) \mathbin{\breve{+}} (\beta' \succ \beta) := (\alpha' + (\alpha \setminus \beta')) \succ (\beta + (\beta' \setminus \alpha)).$$

Our pop in the result of c-addition is intended to mean the result of first popping α' and then popping $\alpha \setminus \beta'$. So the resulting pop has α' *on top of* $\alpha \setminus \beta'$. Remember that we have 0 as our top element. Thus, we are 'counting' the levels of our stacks top-down. So α' on top of $\alpha \setminus \beta'$ becomes $\alpha' + (\alpha \setminus \beta')$. Similarly, our push in the result of addition is intended to be the result of first pushing $(\beta' \setminus \alpha)$ and then β. Thus, we will have β on top of $(\beta' \setminus \alpha)$. So, we get $\beta + (\beta' \setminus \alpha)$.

It's a nice exercise to show that $\breve{+}$ on stack-ordinals is associative. We define addition on stack-numbers by $\xi + \eta := \eta \mathbin{\breve{+}} \xi$. Note that addition is, from our perspective, somewhat less natural than c-addition.

Remark A.1. We can embed the ordinals into the stack-ordinals as follows. We send the ordinal α to the stack-ordinal $0 \succ \alpha$, say via emb. It is easily seen that emb commutes with $+$. We can define the operation $-$ on stack-ordinals by $-(\alpha' \succ \alpha) := (\alpha \succ \alpha')$. Thus, notationally supressing emb, for any ordinal α, we get its corresponding 'negative ordinal' $-\alpha$.

We have, for ordinal α, $-\alpha + \alpha = 0$, but $\alpha + -\alpha = \alpha \succ \alpha$. Clearly, $\alpha \succ \alpha$ is only 0 if $\alpha = 0$. Note that using $-\alpha + \alpha = 0$, we can prove the left cancelation law for ordinals: if $\alpha + \beta = \alpha + \beta'$, then $\beta = -\alpha + \alpha + \beta = -\alpha + \alpha + \beta' = \beta'$.

It is impossible to extend the ordinals to full integer analogues. There can be no homomorphic embedding of the ordinals into a (non-Abelian) group, since, if there was such an embedding, the ordinals would satisfy the right cancelation law: $\alpha + \beta = \alpha' + \beta \Rightarrow \alpha = \alpha'$. Quod non. ❑

To a stack-number $\alpha' \succ \alpha$ we may associate a relation $S_{\alpha' \succ \alpha}$ as follows:

$$\triangleright \; \mathsf{S}^{(\alpha' \succ \alpha)}(\beta) = \gamma :\Leftrightarrow \exists \delta \in \lambda\, (\mathsf{ON})\; \beta = \alpha' + \delta \text{ and } \gamma = \alpha + \delta$$

The functionality of $\mathsf{S}^{(\alpha' \succ \alpha)}$ follows from the fact that δ is uniquely determined by α' and β. We can alternatively define $\mathsf{S}^{(\alpha' \succ \alpha)}$ as follows.

$$\triangleright \; \mathsf{S}^{(\alpha' \succ \alpha)}(\beta) = \begin{cases} \alpha + \alpha' \setminus \beta & \text{if } \alpha' \leq \beta \\ \text{undefined} & \text{otherwise} \end{cases} .$$

One can show that $\mathsf{S}^{\xi};\mathsf{S}^{\eta} = \mathsf{S}^{\xi \dot{+} \eta}$. We end this remark by presenting a rewrite system for the stack-numbers on ω^2. Here is the language.

▷ $B ::= \varepsilon \mid \langle \mid \rangle \mid \{ \mid \} \mid BB.$

These are the rewrite rules.

1.	$\varepsilon B \to B$	5.	$\{\langle \to \{$
2.	$B\varepsilon \to B$	6.	$\{\rangle \to \{$
3.	$\langle\rangle \to \varepsilon$	7.	$\rangle\} \to \}$
4.	$\{\} \to \varepsilon$	8.	$\langle\} \to \}$

Clearly, in this system we have normal forms of the form

$$\overbrace{\}\cdots\}}^{n'\times}\overbrace{\rangle\cdots\rangle}^{m'\times}\overbrace{\langle\cdots\langle}^{m\times}\overbrace{\{\cdots\{}^{n\times}$$

for $m + m' + n + n' > 0$, or we have a normal form ε. These forms represent respectively the stack-numbers $\omega \cdot n' + m' \succ \omega \cdot n + m$ and $0 \succ 0$.

Department of Philosophy, Utrecht University, Janskerhof 13A, 3584 CS Utrecht, The Netherlands

E-mail address: albert.visser@phil.uu.nl