

Referent systems

Udo Klein and Marcus Kracht

Tübingen, 03.05.2010

Contents

1	Motivating referent systems	1
1.1	Two perspectives on the construction of DRSs	1
1.1.1	Procedural perspective	1
1.1.2	Algebraic perspective	2
1.1.3	Accidental identification of referents	2
2	Defining referent systems	3
2.1	Referents	3
2.2	Names	3
2.3	Diacritics	5
2.4	Argument identification statements	6
2.5	Referent systems	8
2.5.1	Merge of rDRSs	9
2.5.2	Fusion of rDRSs	11
3	Applying referent systems	12
3.1	Raising	12
3.2	Control	14
3.2.1	Subject control	14
3.2.2	Object control	15
3.3	Crossing dependencies	17

1 Motivating referent systems

1.1 Two perspectives on the construction of DRSs

1.1.1 Procedural perspective

- Construction algorithm maps a DRS into a modified DRS' if the syntactic structure inside the DRS meets certain triggering conditions.
- The initial DRS contains the complete syntactic structure. So, DRSs are **not constructed in parallel** with syntactic structure.
- The variable introduced by a NP is **dependent** on the variables already introduced, because the semantic contribution of a NP is a procedure involving the introduction of a variable which has to be different from all the variables already introduced

1.1.2 Algebraic perspective

- there is a fixed set of basic DRSs
- there is a set of operations merging DRSs
- DRSs are **constructed in parallel** with syntactic structure
- the variable introduced by a NP is **independent** of the variables introduced by the other DRSs

1.1.3 Accidental identification of referents

If the merge ‘•’ of DRSs is

$$\langle U_1, C_1 \rangle \bullet \langle U_2, C_2 \rangle = \langle U_1 \cup U_2, C_1 \cup C_2 \rangle$$

and the referents of the basic DRSs are fixed, then we risk mistakenly identifying two referents (unless any two DRSs have different variables).

$$\begin{array}{|c|} \hline x \\ \hline \text{Mary}'(x) \\ \hline \end{array} \bullet \left(\begin{array}{|c|} \hline \\ \hline \text{see}'(x, y) \\ \hline \end{array} \bullet \begin{array}{|c|} \hline y \\ \hline \text{car}'(y) \\ \hline \end{array} \right) =$$

$$\begin{array}{|c|} \hline y \\ \hline \text{car}'(y) \\ \hline \end{array} \bullet \left(\begin{array}{|c|} \hline \\ \hline \text{see}'(x, y) \\ \hline \end{array} \bullet \begin{array}{|c|} \hline x \\ \hline \text{Mary}'(x) \\ \hline \end{array} \right) = \begin{array}{|c|} \hline x, y \\ \hline \text{Mary}'(x) \\ \hline \text{car}'(y) \\ \hline \text{see}'(x, y) \\ \hline \end{array}$$

Accidental identification of referents can be avoided by replacing (i) every left-DRS variable x which doesn't identify with a right-DRS variable with x^1 , and (ii) every right-DRS variable x which doesn't identify with a left-DRS variable with x^2 .

$$\begin{array}{|c|} \hline \\ \hline \text{see}'(x, y) \\ \hline \end{array} \bullet \begin{array}{|c|} \hline x \\ \hline \text{Mary}'(x) \\ \hline \end{array} = \begin{array}{|c|} \hline z \\ \hline \text{see}'(x^1, z) \\ \hline \text{Mary}'(z) \\ \hline \end{array}$$

What remains to be done is to specify when and how referents are identified. This is, in a nutshell, the task of referent systems.

There are three types of information relevant for the identification of referents:

- linear information, encoded by **horizontal diacritics**
- hierarchical information, encoded by **vertical diacritics**
- categorial information, encoded by a set of attribute-value pairs called **names**

Argument identification statements are triples consisting of a referent, a pair of vertical and horizontal diacritics and a set of attribute-value pairs.

Referent systems are lists of argument identification statements.

2 Defining referent systems

2.1 Referents

Definition 2.1 A *referent* $x\sigma$ consists of the variable symbol x followed by a sequence $\sigma \in \{1, 2\}^*$.

Example 2.1

$x, x1, x2, x11, x12, x21, x22, x111, x112, x121, x122, \dots$

Convention 2.1 To ease readability we use also the symbols $e, f, g, h, x, y, z, u, v, w, \dots$ standing for referents

2.2 Names

Definition 2.2 A *name space* \mathbf{N} is a triple $\langle A, V, f \rangle$, where A is a finite non-empty set of attributes, V is a finite non-empty set of values disjoint from A , and $f : A \rightarrow \wp(V)$ is a valuation function assigning every attribute in A a subset of V .

Example 2.2

$A = \{\text{GEND, PER, NUM, CASE, CAT, VOICE}\}$

$V = \{m, f, n, 1, 2, 3, \text{sg, pl, nom, gen, dat, acc, n, v, adj, adv, act, pass, } \star\}$

$f :$

GEND	\mapsto	$\{m, f, n, \star\}$
PER	\mapsto	$\{1, 2, 3, \star\}$
NUM	\mapsto	$\{\text{sg, pl, } \star\}$
CASE	\mapsto	$\{\text{nom, gen, dat, acc, } \star\}$
CAT	\mapsto	$\{n, v, \text{adj, adv, } \star\}$
VOICE	\mapsto	$\{\text{act, pass, } \star\}$

\star is a technically value, but is used to express the idea that an attribute is undefined as opposed to being underspecified.

Definition 2.3 A *simple name* N (over a name space $\mathbf{N} = \langle A, V, f \rangle$) is a set N of attribute-value pairs (over \mathbf{N}) such that (i) for every attribute $a \in A$ there is exactly one $v \subseteq f(a)$ with $\langle a : v \rangle \in N$ and $v \neq \emptyset$, and (ii) nothing else is in N .

Example 2.3

$N_1 = \{\langle \text{CAT}, \{n\} \rangle, \langle \text{VOICE}, \{\text{act, pass}\} \rangle, \langle \text{CASE}, \{\text{nom, acc}\} \rangle, \langle \text{NUM}, \{\text{sg}\} \rangle, \langle \text{GEND}, \{m\} \rangle, \langle \text{PER}, \{3\} \rangle\}$ is a simple name (every attribute has a nonempty value).

$N_2 = \{\langle \text{CAT}, \{n\} \rangle, \langle \text{CASE}, \{\text{nom, acc}\} \rangle, \langle \text{NUM}, \{\text{sg}\} \rangle, \langle \text{GEND}, \{m\} \rangle, \langle \text{PER}, \{3\} \rangle\}$ is not a simple name, because it lacks an attribute value pair for the attribute VOICE.

$N_3 = \{\langle \text{CAT}, \{n\} \rangle, \langle \text{VOICE}, \emptyset \rangle, \langle \text{CASE}, \{\text{nom, acc}\} \rangle, \langle \text{NUM}, \{\text{sg}\} \rangle, \langle \text{GEND}, \{m\} \rangle, \langle \text{PER}, \{3\} \rangle\}$ is not a simple name, because the value of VOICE is the empty set.

Convention 2.2 The name

$\{\langle \text{CAT}, \{n\} \rangle, \langle \text{VOICE}, \{\text{act, pass}\} \rangle, \langle \text{CASE}, \{\text{nom, acc}\} \rangle, \langle \text{NUM}, \{\text{sg}\} \rangle, \langle \text{GEND}, \{m\} \rangle, \langle \text{PER}, \{3\} \rangle\}$

will be represented by the matrix

$$\begin{bmatrix} \text{CAT} & : & \text{n} \\ \text{CASE} & : & \text{nom} \sqcup \text{acc} \\ \text{NUM} & : & \text{sg} \\ \text{GEND} & : & \text{m} \\ \text{PER} & : & \mathbf{3} \end{bmatrix}$$

An attribute a may be omitted from the matrix if and only if its value is $f(a)$. This is why $\text{VOICE} : \text{act} \sqcup \text{pass}$ can be omitted from the matrix.

Definition 2.4 A *transformer name* \mathfrak{N} (over $\mathbf{N} = \langle A, V, f \rangle$) is a pair $\langle N, N' \rangle$ of simple names (over \mathbf{N}) such that $\langle a, f(a) \rangle \in N$ iff $\langle a, f(a) \rangle \in N'$.

Example 2.4

$$\mathfrak{N} = \left\langle \begin{bmatrix} \text{CAT} & : & \text{v} \\ \text{VOICE} & : & \star \end{bmatrix}, \begin{bmatrix} \text{CAT} & : & \text{v} \\ \text{VOICE} & : & \text{pass} \end{bmatrix} \right\rangle$$

is a transformer name.

Convention 2.3 We shall use the more compact

$$\begin{bmatrix} \text{CAT} & : & \text{v} \\ \text{VOICE} & : & \star \mapsto \text{pass} \end{bmatrix}$$

for

$$\left\langle \begin{bmatrix} \text{CAT} & : & \text{v} \\ \text{VOICE} & : & \star \end{bmatrix}, \begin{bmatrix} \text{CAT} & : & \text{v} \\ \text{VOICE} & : & \text{pass} \end{bmatrix} \right\rangle$$

Definition 2.5 Let M, N be two simple names over \mathbf{N} . Then $M \sqcap N$, the *intersection* of M and N , is defined as follows:

$$M \sqcap N = \{ \langle a : v_1 \cap v_2 \rangle : a \in A, \langle a : v_1 \rangle \in M \text{ and } \langle a : v_2 \rangle \in N \}$$

Definition 2.6 A name m *matches* a name n iff:

- (i) m, n and $m \sqcap n$ are simple names
- (ii) $m = \langle A, B \rangle$, $n = C$ is a simple name, and $A \sqcap C$ is a simple name
- (iii) $m = A$ is a simple name, $n = \langle C, D \rangle$ is a transformer name, and $A \sqcap D$ is a simple name
- (iv) $m = \langle A, B \rangle$, $n = \langle C, D \rangle$ and $A \sqcap D$ is a simple name

Example 2.5

$$\begin{bmatrix} \text{CAT} & : & \text{n} \\ \text{CASE} & : & \text{nom} \sqcup \text{acc} \\ \text{NUM} & : & \text{sg} \\ \text{GEND} & : & \text{m} \\ \text{PER} & : & \mathbf{3} \end{bmatrix} \text{ matches } \begin{bmatrix} \text{CAT} & : & \text{n} \\ \text{CASE} & : & \text{nom} \\ \text{NUM} & : & \text{sg} \\ \text{GEND} & : & \text{m} \\ \text{PER} & : & \mathbf{3} \end{bmatrix}$$

because

$$\begin{bmatrix} \text{CAT} & : & \text{n} \\ \text{CASE} & : & \text{nom} \sqcup \text{acc} \\ \text{NUM} & : & \text{sg} \\ \text{GEND} & : & \text{m} \\ \text{PER} & : & \text{3} \end{bmatrix} \sqcap \begin{bmatrix} \text{CAT} & : & \text{n} \\ \text{CASE} & : & \text{nom} \\ \text{NUM} & : & \text{sg} \\ \text{GEND} & : & \text{m} \\ \text{PER} & : & \text{3} \end{bmatrix} = \begin{bmatrix} \text{CAT} & : & \text{n} \\ \text{CASE} & : & \text{nom} \\ \text{NUM} & : & \text{sg} \\ \text{GEND} & : & \text{m} \\ \text{PER} & : & \text{3} \end{bmatrix}$$

Example 2.6

$$\begin{bmatrix} \text{CAT} & : & \text{n} \\ \text{CASE} & : & \text{acc} \\ \text{NUM} & : & \text{sg} \\ \text{GEND} & : & \text{m} \\ \text{PER} & : & \text{3} \end{bmatrix} \text{ does not match } \begin{bmatrix} \text{CAT} & : & \text{n} \\ \text{CASE} & : & \text{nom} \\ \text{NUM} & : & \text{sg} \\ \text{GEND} & : & \text{m} \\ \text{PER} & : & \text{3} \end{bmatrix}$$

because

$$\begin{bmatrix} \text{CAT} & : & \text{n} \\ \text{CASE} & : & \text{acc} \\ \text{NUM} & : & \text{sg} \\ \text{GEND} & : & \text{m} \\ \text{PER} & : & \text{3} \end{bmatrix} \sqcap \begin{bmatrix} \text{CAT} & : & \text{n} \\ \text{CASE} & : & \text{nom} \\ \text{NUM} & : & \text{sg} \\ \text{GEND} & : & \text{m} \\ \text{PER} & : & \text{3} \end{bmatrix} = \begin{bmatrix} \text{CAT} & : & \text{n} \\ \text{CASE} & : & \emptyset \\ \text{NUM} & : & \text{sg} \\ \text{GEND} & : & \text{m} \\ \text{PER} & : & \text{3} \end{bmatrix}$$

is not a simple name.

Definition 2.7 Let m, n be two names such that m matches n . Then the **resulting name** $m \cdot n$ is:

$$m \cdot n = \begin{cases} m \sqcap n, & \text{if } m, n \text{ are simple names} \\ B, & \text{if } m = \langle A, B \rangle, n = C \\ C, & \text{if } m = A, n = \langle C, D \rangle \\ \langle C, B \rangle, & \text{if } m = \langle A, B \rangle, n = \langle C, D \rangle \end{cases}$$

Example 2.7

$$\begin{bmatrix} \text{CAT} & : & \text{v} \\ \text{VOICE} & : & \star \mapsto \text{pass} \end{bmatrix} \cdot \begin{bmatrix} \text{CAT} & : & \text{v} \\ \text{VOICE} & : & \star \end{bmatrix} = \begin{bmatrix} \text{CAT} & : & \text{v} \\ \text{VOICE} & : & \text{pass} \end{bmatrix}$$

2.3 Diacritics

Definition 2.8 A **vertical diacritic** vd is either a subset of $\{\Delta, \nabla\}$ or a subset of $\{\blacktriangle, \blacktriangledown\}$. A **horizontal diacritic** hd is a subset of $\{\ominus, \otimes\}$.

Convention 2.4 For ease of readability, we use the following conventions for representing

vertical and horizontal diacritics:

	definition	convention
vertical diacritics	\emptyset	—
	$\{\Delta\}$	Δ
	$\{\nabla\}$	∇
	$\{\Delta, \nabla\}$	\diamond
	$\{\blacktriangle\}$	\blacktriangle
	$\{\blacktriangledown\}$	\blacktriangledown
	$\{\blacktriangle, \blacktriangledown\}$	\blacklozenge
horizontal diacritics	\emptyset	\circ
	$\{\ominus\}$	\otimes
	$\{\odot\}$	\odot
	$\{\oslash\}$	\oplus

Definition 2.9 A *diacritic* d is a pair $\langle vd, hd \rangle$ consisting of a vertical diacritic vd and a horizontal diacritic hd .

Definition 2.10 A diacritic $\langle vd, hd \rangle$ is a *legal diacritic* if:

- ∇ or $\blacktriangledown \in vd \leftrightarrow hd \neq \emptyset$

The diacritic $\langle \emptyset, \emptyset \rangle$ is called *trivial*.

Example 2.8

The following are examples of legal and illegal diacritics - in the right column we use the convention for representing vertical and horizontal diacritics:

legal diacritics	$\langle \{\nabla\}, \{\odot\} \rangle$	$\nabla \otimes$
	$\langle \{\Delta\}, \emptyset \rangle$	$\Delta \circ$
illegal diacritics	$\langle \{\nabla\}, \emptyset \rangle$	$\nabla \circ$
	$\langle \{\Delta\}, \{\odot\} \rangle$	$\Delta \otimes$

2.4 Argument identification statements

Definition 2.11 An *argument identification statement (AIS)* μ is a triple :

- $\langle x, \langle vd, hd \rangle, N \rangle$, where x is a referent, $\langle vd, hd \rangle$ a legal diacritic with $|vd| < 2$, and N a simple name (over a name space N), or
- $\langle x, \langle \{\Delta, \nabla\}, hd \rangle, \mathfrak{R} \rangle$, where x is a referent, $\langle \{\Delta, \nabla\}, hd \rangle$ a legal diacritic, and \mathfrak{R} a transformer name.

Example 2.9

$$\langle x12, \diamond \otimes, \left[\begin{array}{l} \text{CAT} : v \\ \text{VOICE} : \star \mapsto \text{pass} \end{array} \right] \rangle$$

stands for the triple

$$\langle x12, \langle \{\Delta, \nabla\}, \{\odot\} \rangle, \langle N_1, N_2 \rangle \rangle$$

where $N_1 = \{ \langle \text{CAT}, \{n\} \rangle, \langle \text{VOICE}, \star \rangle, \langle \text{CASE}, \{\text{nom}, \text{gen}, \text{dat}, \text{acc}\} \rangle, \langle \text{NUM}, \{\text{sg}, \text{pl}\} \rangle, \langle \text{GEND}, \{\text{m}, \text{f}, \text{n}\} \rangle, \langle \text{PER}, \{1, 2, 3\} \rangle \}$ and $N_2 = \{ \langle \text{CAT}, \{n\} \rangle, \langle \text{VOICE}, \{\text{pass}\} \rangle, \langle \text{CASE}, \{\text{nom}, \text{gen}, \text{dat}, \text{acc}\} \rangle, \langle \text{NUM}, \{\text{sg}, \text{pl}\} \rangle, \langle \text{GEND}, \{\text{m}, \text{f}, \text{n}\} \rangle, \langle \text{PER}, \{1, 2, 3\} \rangle \}$

Definition 2.12 Let $\alpha = \langle x, \langle \text{vd}, \text{hd} \rangle, n \rangle$ be an argument identification statement. Then $\text{ref}(\alpha) = x$, $\text{vd}(\alpha) = \text{vd}$, $\text{hd}(\alpha) = \text{hd}$, $\text{n}(\alpha) = n$.

Definition 2.13 Let $\text{vd}(\mu)$ and $\text{vd}(\nu)$ be the vertical diacritics of two AISs μ and ν . Then $\text{vd}(\mu) \cap \text{vd}(\nu)$ is defined as follows:

$\text{vd}(\mu)$	$\text{vd}(\nu)$	$\text{vd}(\mu) \cap \text{vd}(\nu)$
$\{\nabla\}$	$\{\Delta\}$	\emptyset
$\{\nabla\}$	$\{\nabla, \Delta\}$	$\{\nabla\}$
$\{\nabla, \Delta\}$	$\{\Delta\}$	$\{\Delta\}$
$\{\nabla, \Delta\}$	$\{\nabla, \Delta\}$	$\{\nabla, \Delta\}$
$\{\blacktriangledown\}$	$\{\Delta\}$	\emptyset
$\{\blacktriangledown\}$	$\{\nabla, \Delta\}$	$\{\blacktriangledown\}$
$\{\blacktriangledown, \blacktriangle\}$	$\{\Delta\}$	$\{\Delta\}$
$\{\blacktriangledown, \blacktriangle\}$	$\{\nabla, \Delta\}$	$\{\blacktriangledown, \blacktriangle\}$

Definition 2.14 The *rightward merge* of two AISs $\mu \triangleright \nu$ is defined iff:

- (i) $\nabla \in \text{vd}(\mu)$ or $\blacktriangledown \in \text{vd}(\mu)$
- (ii) $\Delta \in \text{vd}(\nu)$
- (iii) $\emptyset \in \text{hd}(\mu)$
- (iv) $\text{n}(\mu)$ matches $\text{n}(\nu)$

In this case $\mu \triangleright \nu = \langle \text{ref}(\mu)^{-1}, \langle \text{vd}(\mu) \cap \text{vd}(\nu), \text{hd}(\nu) \rangle, \text{n}(\mu) \cdot \text{n}(\nu) \rangle$.

Definition 2.15 The *leftward merge* of two AISs $\nu \triangleleft \mu$ is defined iff:

- (i) $\nabla \in \text{vd}(\mu)$ or $\blacktriangledown \in \text{vd}(\mu)$
- (ii) $\Delta \in \text{vd}(\nu)$
- (iii) $\emptyset \in \text{hd}(\mu)$
- (iv) $\text{n}(\mu)$ matches $\text{n}(\nu)$

In this case $\nu \triangleleft \mu = \langle \text{ref}(\mu)^{-2}, \langle \text{vd}(\nu) \cap \text{vd}(\mu), \text{hd}(\nu) \rangle, \text{n}(\mu) \cdot \text{n}(\nu) \rangle$.

Example 2.10

$$\langle x, \Delta \circ, \left[\begin{array}{l} \text{CAT} : \nabla \\ \text{VOICE} : \star \end{array} \right] \rangle \triangleleft \langle x1, \diamond \emptyset, \left[\begin{array}{l} \text{CAT} : \nabla \\ \text{VOICE} : \star \mapsto \text{pass} \end{array} \right] \rangle = \langle x11, \Delta \circ, \left[\begin{array}{l} \text{CAT} : \nabla \\ \text{VOICE} : \text{pass} \end{array} \right] \rangle$$

2.5 Referent systems

Definition 2.16 A list of argument identification statements $[\mu_1, \dots, \mu_m]$, $m \geq 1$, is called a *referent system*.

Convention 2.5 The list

$$[\langle x : \Delta \circ : \left[\begin{array}{l} \text{CAT} : v \end{array} \right] \rangle \\ \langle x1 : \nabla \ominus : \left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right] \rangle \\ \langle x2 : \nabla \ominus : \left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{acc} \end{array} \right] \rangle]$$

will be represented by

$x : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : v \end{array} \right]$
$x1 : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right]$
$x2 : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{acc} \end{array} \right]$

When two referent systems merge, one is required to be saturated. The fusion of two referent systems does not require one of them to be saturated.

Definition 2.17 A referent system $\alpha = [\mu_1, \dots, \mu_m]$ is *saturated* iff $\forall i, 1 \leq i \leq m, \{\Delta\} \subseteq \mathbf{vd}(\mu_i)$.

Definition 2.18 Let $\alpha = [\mu_1, \dots, \mu_m]$ and $\beta = [v_1, \dots, v_n]$ be two referent systems. Then v_j ($1 \leq j \leq n$) *E-accesses* μ_i ($1 \leq i \leq m$) iff

- (i) $j = 1$, and
- (ii) $i = m$, and
- (iii) either $\mu_m \triangleright v_1$ or $v_1 \triangleleft \mu_m$ is defined

Definition 2.19 Let $\alpha = [\mu_1, \dots, \mu_m]$ and $\beta = [v_1, \dots, v_n]$ be two referent systems. Then v_j ($1 \leq j \leq n$) *G-accesses* μ_i ($1 \leq i \leq m$) iff

- (i) $j = 1$, and
- (ii) either $\mu_m \triangleright v_1$ or $v_1 \triangleleft \mu_m$ is defined
- (iii) there is no μ_k with $i < k \leq m$ such that $\mu_k \triangleright v_1$ or $v_1 \triangleleft \mu_k$ is defined

Example 2.11

$x : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \sqcup \text{acc} \end{array} \right]$
----------------------	--

$x1 : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : v \end{array} \right]$
$x2 : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right]$
$x11 : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{acc} \end{array} \right]$
$x12 : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{dat} \end{array} \right]$

ν_1 does not E -access μ_4 , because their names do not match.

However, ν_1 G -accesses μ_3 , because (i) both the diacritics and the names match, and (ii) there is no μ_k with $k > 3$ where ν_1 G -accesses μ_k .

ν_1 does not G -access μ_2 , although both the diacritics and the names match, because there is a $k > 2$ where ν_1 G -accesses μ_k , namely $k = 3$.

Definition 2.20 Let $\alpha = [h \mid t]$ be a list. Then $r(\alpha)$, the **reduction of α** , is defined as follows:

$$r(\alpha) = \begin{cases} [h \mid r(t)], & \text{if } \alpha = [h \mid t], h \text{ is an AIS, and } \mathbf{vd}(h) \neq \emptyset \\ r(t), & \text{if } \alpha = [h \mid t], h \text{ is an AIS, and } \mathbf{vd}(h) = \emptyset \\ [] & \text{if } \alpha = [] \end{cases}$$

When the reduction function applies to a referent systems it throws out all AISs μ_i where $\mathbf{vd}(\mu_i) = \emptyset$, while keeping the relative order of the other AISs intact.

Example 2.12

$$r\left(\begin{array}{l} x11 : \Delta \circ : \left[\begin{array}{l} \text{CAT} : v \\ \end{array} \right] \\ x12 : \nabla \ominus : \left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right] \\ x21 : -\circ : \left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{acc} \end{array} \right] \\ x22 : \nabla \ominus : \left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{dat} \end{array} \right] \end{array} \right) = \begin{array}{l} x11 : \Delta \circ : \left[\begin{array}{l} \text{CAT} : v \\ \end{array} \right] \\ x12 : \nabla \ominus : \left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right] \\ x22 : \nabla \ominus : \left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{dat} \end{array} \right] \end{array}$$

2.5.1 Merge of rDRSs

Definition 2.21 An **rDRS** \mathfrak{D} is a pair $\langle \alpha, \Delta \rangle$ consisting of a referent system α and a discourse representation structure Δ such that every unbound referent of Δ occurs in α .

- The direction of the merge operation is dictated by the position of the saturated referent system: if it is to the right, then merge is rightward, if it is to the left, then merge is leftward.
- The definition of merge (and fusion) is such that the result of merging two AISs is not part of the resulting referent system if the resulting AIS has an empty diacritic (they cannot participate in another merge or fusion operation, and are therefore not further needed).
- The definitions of rightward merge and fusion require that the AISs of the left referent system which \triangleright -merge form an uninterrupted (or strict) sequence.
- As a consequence of (i) not adding AISs with empty diacritics to the resulting referent system and (ii) lexical referent systems not containing any empty AISs, one of the two referent systems in a merge will contain only AISs where $\{\Delta\} \subseteq \mathbf{vd}$.
- The merge definition is parameterised – it is short for two (or more) definitions, one for each type of access. We introduced two types of access, namely E -access and G -access.

Definition 2.22 Let $\langle \alpha, \Delta \rangle$ and $\langle \beta, \Gamma \rangle$ be two rDRSs, with $\alpha = [\mu_1, \dots, \mu_m], \beta = [v_1, \dots, v_n]$. The *rightward merge of two rDRSs* $\langle \alpha, \Delta \rangle \circ_r \langle \beta, \Gamma \rangle$ is defined iff:

- β is saturated
- there is an $i, n \leq i \leq m$ such that
 - v_1 L-accesses μ_i
 - for every k with $1 \leq k \leq n$, $\mu_{i-k+1} \triangleright v_k$ is defined

In this case $\langle \alpha, \Delta \rangle \circ_r \langle \beta, \Gamma \rangle = \langle \mathfrak{r}([\epsilon_p : 1 \leq p \leq m]), \mathbf{1}(\Delta) \cup \mathbf{p}_1(\Gamma) \rangle$ where:

- $\epsilon_p = \begin{cases} \mu_{i-p+1} \triangleright v_p & \text{if } i - n + 1 \leq p \leq i \\ \langle \mathbf{ref}(\mu_p)^{\frown} 1, \langle \mathbf{vd}(\mu_p), \mathbf{hd}(\mu_p) \rangle, \mathbf{n}(\mu_p) \rangle & \text{else} \end{cases}$
- $\mathbf{1}(\Delta)$ is the result of replacing all occurrences of free variables r in Δ by $r^{\frown} 1$
- $\mathbf{p}_1(\Gamma)$ is the result of replacing all occurrences of free variables r in Γ by $\mathbf{ref}(\mathbf{p}(v_k))^{\frown} 1$, where $\mathbf{ref}(v_k) = r$ and $\mathbf{p}(v_k) = \mu_{i-k+1}$

Definition 2.23 Let $\langle \alpha, \Delta \rangle$ and $\langle \beta, \Gamma \rangle$ be two rDRSs, with $\alpha = [\mu_1, \dots, \mu_m], \beta = [v_1, \dots, v_n]$. The *leftward merge of two rDRSs* $\langle \beta, \Gamma \rangle \circ_l \langle \alpha, \Delta \rangle$ is defined iff:

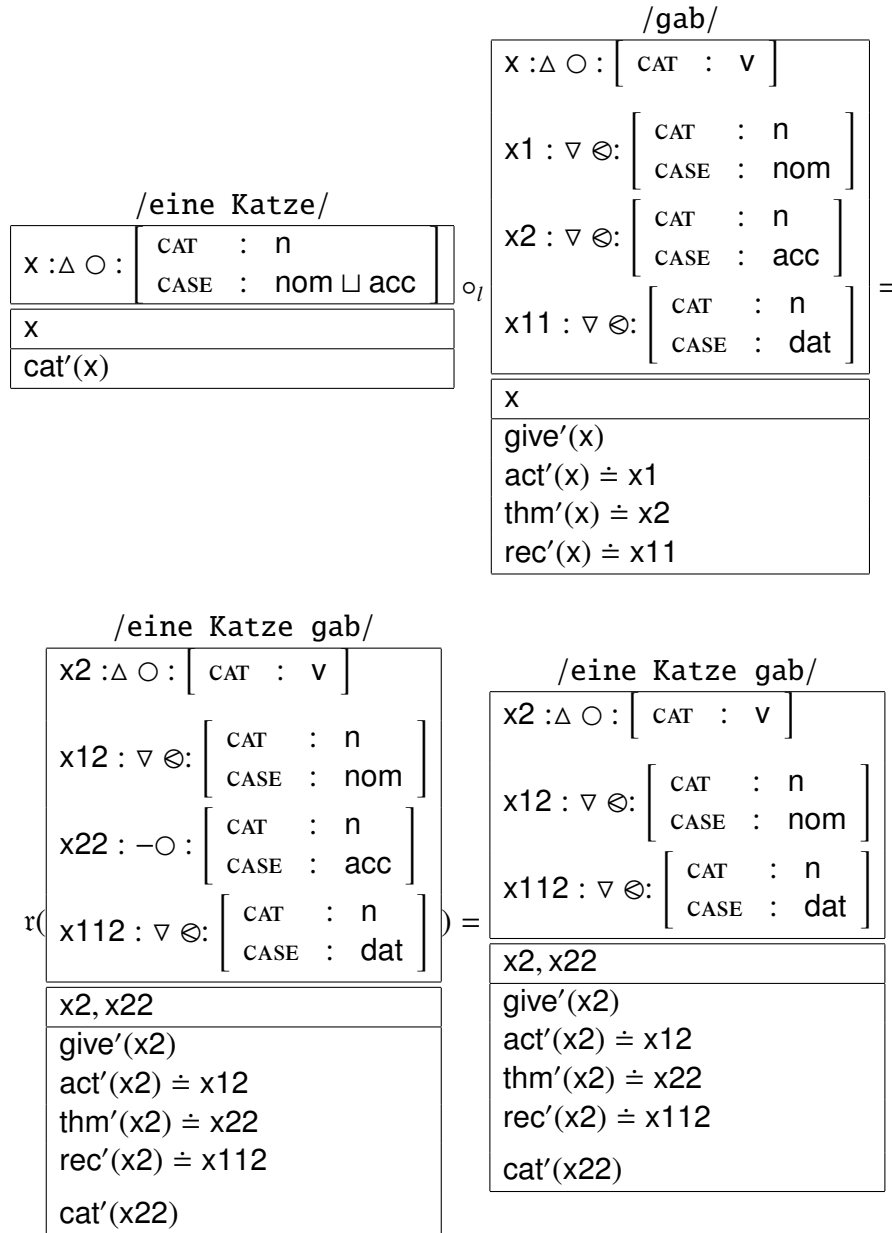
- β is saturated
- there is an $i, n \leq i \leq m$ such that
 - v_1 L-accesses μ_i
 - for every k with $1 \leq k \leq n$, $v_k \triangleleft \mu_{i-k+1}$ is defined

In this case $\langle \beta, \Gamma \rangle \circ_l \langle \alpha, \Delta \rangle = \langle \mathfrak{r}([\epsilon_p : 1 \leq p \leq m]), \mathbf{p}_2(\Gamma) \cup \mathbf{2}(\Delta) \rangle$ where:

- $\epsilon_p = \begin{cases} v_p \triangleleft \mu_{i-p+1} & \text{if } i - n + 1 \leq p \leq i \\ \langle \mathbf{ref}(\mu_p)^{\frown} 2, \langle \mathbf{vd}(\mu_p), \mathbf{hd}(\mu_p) \rangle, \mathbf{n}(\mu_p) \rangle & \text{else} \end{cases}$
- $\mathbf{2}(\Delta)$ is the result of replacing all occurrences of free variables r in Δ by $r^{\frown} 2$
- $\mathbf{p}_2(\Gamma)$ is the result of replacing all occurrences of free variables r in Γ by $\mathbf{ref}(\mathbf{p}(v_k))^{\frown} 2$, where $\mathbf{ref}(v_k) = r$ and $\mathbf{p}(v_k) = \mu_{i-k+1}$

Example 2.13

Monadic merge with G -access:



Example 2.14

Polyadic merge: see section 3.1

2.5.2 Fusion of rDRSs

Both leftward and rightward merge of two rDRSs require one of the referent systems to be saturated, i.e. one of the referent systems contains no AIS ν with $\mathbf{vd}(\nu) = \emptyset$ or $\mathbf{vd}(\nu) = \{\nabla\}$.

Fusion, on the other hand, is more liberal, in the sense that it allows the combination of two rDRS even if neither referent system is saturated.

Fusion only applies if the referent system of the 'argument' rDRS contains at least one AHS ν with $\mathbf{vd}(\nu) = \{\nabla\}$

We assume that the order of AIS in a lexical referent system is such that all AIS ν with $\mathbf{vd}(\nu) = \{\nabla\}$ follow all AIS with $\mathbf{vd}(\nu) \neq \{\nabla\}$

Definition 2.24 Let $\langle \alpha, \Delta \rangle$ and $\langle \beta, \Gamma \rangle$ be two rDRSs, with $\alpha = [\mu_1, \dots, \mu_m], \beta = [\nu_1, \dots, \nu_n]$. The *rightward fusion of two rDRSs* $\langle \alpha, \Delta \rangle \bullet_r \langle \beta, \Gamma \rangle$ is defined iff:

- $\beta = [\nu_{a_1}, \dots, \nu_{a_i}, \nu_{b_1}, \dots, \nu_{b_j}], a_i \geq 1, b_j \geq 1$ such that
 - for every $k, 1 \leq k \leq i, \{\Delta\} \subseteq \mathbf{vd}(\nu_{a_k})$
 - for every $l, 1 \leq l \leq j, \mathbf{vd}(\nu_{b_l}) = \{\nabla\}$
- there is an $i, 1 \leq i \leq m$ such that
 - ν_{a_1} L-accesses μ_i
 - $\nabla \in \mathbf{vd}(\mu_i)$
 - for every k with $a_1 \leq a_k \leq a_{i-1}, \mu_{i-(k-1)} \triangleright \nu_{1+a_k}$ is defined

In this case $\langle \alpha, \Delta \rangle \bullet_r \langle \beta, \Gamma \rangle = \langle r([\epsilon_p : 1 \leq p \leq m]), \mathbf{1}(\Delta) \cup \mathbf{p}_1(\Gamma) \rangle$ where:

$$\bullet \epsilon_p = \begin{cases} \mu_p \triangleright \nu_{1+(1-p)} & \text{if } i - (n - 1) \leq p \leq i \\ \langle \mathbf{ref}(\mu_p) \frown 1, \langle \mathbf{vd}(\mu_p), \mathbf{hd}(\mu_p) \rangle, \mathbf{n}(\mu_p) \rangle & \text{if } 1 \leq p < i - (n - 1) \text{ or } i < p \leq m \\ \langle \mathbf{ref}(\nu_p) \frown 2, \langle \mathbf{vd}(\nu_p), \mathbf{hd}(\nu_p) \rangle, \mathbf{n}(\nu_p) \rangle & \text{if } p = m + b_l \text{ for some } 1 \leq l \leq b_j \end{cases}$$

- $\mathbf{1}(\Delta)$ is the result of replacing all occurrences of free variables r in Δ by $r \frown 1$
- $\mathbf{p}_1(\Gamma)$ is the result of replacing all occurrences of free variables r by:

$$\begin{cases} \mathbf{ref}(\mathbf{p}(v_k)) \frown 1, & \text{if } \mathbf{ref}(v_k) = r \text{ and } \mathbf{p}(v_k) = \mu_{i-k+1} \\ r \frown 2, & \text{otherwise} \end{cases}$$

3 Applying referent systems

3.1 Raising

<p style="text-align: center;">/seems/</p> <table border="1" style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 5px;">$e : \Delta \circ :$</td> <td style="padding: 5px;"> <table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: v</td></tr> <tr><td style="padding: 2px;">TYP</td><td style="padding: 2px;">: fin</td></tr> </table> </td> </tr> <tr> <td style="padding: 5px;">$x : \nabla \otimes :$</td> <td style="padding: 5px;"> <table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: n</td></tr> <tr><td style="padding: 2px;">CASE</td><td style="padding: 2px;">: nom</td></tr> </table> </td> </tr> <tr> <td style="padding: 5px;">$f : \nabla \otimes :$</td> <td style="padding: 5px;"> <table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: v</td></tr> <tr><td style="padding: 2px;">TYP</td><td style="padding: 2px;">: inf</td></tr> </table> </td> </tr> </table>	$e : \Delta \circ :$	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: v</td></tr> <tr><td style="padding: 2px;">TYP</td><td style="padding: 2px;">: fin</td></tr> </table>	CAT	: v	TYP	: fin	$x : \nabla \otimes :$	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: n</td></tr> <tr><td style="padding: 2px;">CASE</td><td style="padding: 2px;">: nom</td></tr> </table>	CAT	: n	CASE	: nom	$f : \nabla \otimes :$	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: v</td></tr> <tr><td style="padding: 2px;">TYP</td><td style="padding: 2px;">: inf</td></tr> </table>	CAT	: v	TYP	: inf	\circ_r	<p style="text-align: center;">/to sleep/</p> <table border="1" style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 5px;">$g : \Delta \circ :$</td> <td style="padding: 5px;"> <table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: v</td></tr> <tr><td style="padding: 2px;">TYP</td><td style="padding: 2px;">: inf</td></tr> </table> </td> </tr> <tr> <td style="padding: 5px;">$y : ? :$</td> <td style="padding: 5px;"> <table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: n</td></tr> <tr><td style="padding: 2px;">CASE</td><td style="padding: 2px;">: ★</td></tr> </table> </td> </tr> </table>	$g : \Delta \circ :$	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: v</td></tr> <tr><td style="padding: 2px;">TYP</td><td style="padding: 2px;">: inf</td></tr> </table>	CAT	: v	TYP	: inf	$y : ? :$	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: n</td></tr> <tr><td style="padding: 2px;">CASE</td><td style="padding: 2px;">: ★</td></tr> </table>	CAT	: n	CASE	: ★
$e : \Delta \circ :$	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: v</td></tr> <tr><td style="padding: 2px;">TYP</td><td style="padding: 2px;">: fin</td></tr> </table>	CAT	: v	TYP	: fin																											
CAT	: v																															
TYP	: fin																															
$x : \nabla \otimes :$	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: n</td></tr> <tr><td style="padding: 2px;">CASE</td><td style="padding: 2px;">: nom</td></tr> </table>	CAT	: n	CASE	: nom																											
CAT	: n																															
CASE	: nom																															
$f : \nabla \otimes :$	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: v</td></tr> <tr><td style="padding: 2px;">TYP</td><td style="padding: 2px;">: inf</td></tr> </table>	CAT	: v	TYP	: inf																											
CAT	: v																															
TYP	: inf																															
$g : \Delta \circ :$	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: v</td></tr> <tr><td style="padding: 2px;">TYP</td><td style="padding: 2px;">: inf</td></tr> </table>	CAT	: v	TYP	: inf																											
CAT	: v																															
TYP	: inf																															
$y : ? :$	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">CAT</td><td style="padding: 2px;">: n</td></tr> <tr><td style="padding: 2px;">CASE</td><td style="padding: 2px;">: ★</td></tr> </table>	CAT	: n	CASE	: ★																											
CAT	: n																															
CASE	: ★																															
e	g																															
$\text{seem}'(e)$	$\text{sleep}'(g)$																															
$\text{thm}'(e) \doteq f$	$\text{act}'(g) \doteq y$																															

How can the referents x and y be identified? What is the vertical diacritic of y ? **Idea:**

/seems/				/to sleep/	
$e : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{fin} \end{array} \right]$	\circ_r	$g : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{inf} \end{array} \right]$	=
$x : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right]$		$y : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \star \end{array} \right]$	
$z : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \star \end{array} \right]$		g		
$f : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{inf} \end{array} \right]$		$\text{sleep}'(g)$		
e		$\text{act}'(g) \doteq y$			
$\text{seem}'(e)$		$\text{thm}'(e) \doteq f$			
$x \doteq z$					

/seems to sleep/				/seems to sleep/	
$e1 : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{fin} \end{array} \right]$	\circ_r	$e1 : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{fin} \end{array} \right]$	=
$x1 : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right]$		$x1 : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right]$	
$z1 : -\circ :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \star \end{array} \right]$		$e1, f1$		
$f1 : -\circ :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{inf} \end{array} \right]$		$\text{seem}'(e1)$		
$e1, f1$		$\text{thm}'(e1) \doteq f1$			
$\text{seem}'(e1)$		$x1 \doteq z1$			
$\text{thm}'(e1) \doteq f1$		$\text{sleep}'(f1)$			
$x1 \doteq z1$		$\text{act}'(f1) \doteq z1$			
$\text{sleep}'(f1)$					
$\text{act}'(f1) \doteq z1$					

f identifies with g , and at the same time z identifies with y .

/Rebecca/				/seems to sleep/	
$v : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \sqcup \text{acc} \sqcup \text{dat} \end{array} \right]$	\circ_r	$e1 : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{fin} \end{array} \right]$	=
v			$x1 : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right]$	
$v \doteq \text{rebecca}'$			$e1, f1$		
			$\text{seem}'(e1)$		
		$\text{thm}'(e1) \doteq f1$			
		$x1 \doteq z1$			
		$\text{sleep}'(f1)$			
		$\text{act}'(f1) \doteq z1$			

/Rebecca seems to sleep/

$e_{12} : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{fin} \end{array} \right]$
$x_{12} : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right]$
e12, f12, x12	
seem'(e12) thm'(e12) \doteq f12 x12 \doteq z12	
sleep'(f12) act'(f12) \doteq z12	
x12 \doteq rebecca'	

3.2 Control

3.2.1 Subject control

/promised/

$e : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{fin} \end{array} \right]$
$x : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right]$
$z : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \star \end{array} \right]$
$f : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{inf} \end{array} \right]$
e	
promise(e) act'(e) \doteq x thm'(e) \doteq f x \doteq z	

/to sleep/

$g : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{inf} \end{array} \right]$
$y : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \star \end{array} \right]$
g	
sleep'(g) act'(g) \doteq y	

or

=

/promised to sleep/	
$e1 : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{fin} \end{array} \right]$
$x1 : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right]$
$z1 : -\circ :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \star \end{array} \right]$
$r(\text{f1} : -\circ :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{inf} \end{array} \right]$
$e1, f1$	
<p>promise(e1) act'(e1) \doteq x1 thm'(e1) \doteq f1 x1 \doteq z1</p> <p>sleep'(f1) act'(f1) = z1</p>	

f identifies with g, and at the same time z identifies with y.

3.2.2 Object control

/persuaded/	
$e : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{fin} \end{array} \right]$
$x : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right]$
$z : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \star \end{array} \right]$
$f : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{inf} \end{array} \right]$
$y : \nabla \ominus :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{acc} \end{array} \right]$
e	
<p>persuade(e) act'(e) \doteq x pat'(e) \doteq y thm'(e) \doteq f y \doteq z</p>	

/Rebecca/	
$v : \Delta \circ :$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \sqcup \text{acc} \sqcup \text{dat} \end{array} \right]$
v	
$v \doteq \text{rebecca}'$	

\circ_r =

/persuaded Rebecca/

$e1:\Delta \circ:$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{fin} \end{array} \right]$
$x1:\nabla \otimes:$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right]$
$z1:\nabla \otimes:$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \star \end{array} \right]$
$f1:\nabla \otimes:$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{inf} \end{array} \right]$
$e1, y1$	
<p>persuade(e1) act'(e1) \doteq x1 pat'(e1) \doteq y1 thm'(e1) \doteq f1 y1 \doteq z1</p> <p>y1 \doteq rebecca'</p>	

/persuaded Rebecca/

$e1:\Delta \circ:$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{fin} \end{array} \right]$
$x1:\nabla \otimes:$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \text{nom} \end{array} \right]$
$z1:\nabla \otimes:$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \star \end{array} \right]$
$f1:\nabla \otimes:$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{inf} \end{array} \right]$
$e1, y1$	
<p>persuade(e1) act'(e1) \doteq x1 pat'(e1) \doteq y1 thm'(e1) \doteq f1 y1 \doteq z1</p> <p>y1 \doteq rebecca'</p>	

/to sleep/

$g:\Delta \circ:$	$\left[\begin{array}{l} \text{CAT} : v \\ \text{TYP} : \text{inf} \end{array} \right]$
$y:\Delta \circ:$	$\left[\begin{array}{l} \text{CAT} : n \\ \text{CASE} : \star \end{array} \right]$
g	
<p>sleep'(g) act'(g) \doteq y</p>	

\circ_r

=

/persuaded Rebecca to sleep/

e11:Δ ○:	CAT : v TYP : fin
x11:∇ ⊗:	CAT : n CASE : nom
e11, y11, f11	
<p>persuade(e11) act'(e11) ≐ x11 pat'(e11) ≐ y11 thm'(e11) ≐ f11 y11 ≐ z11</p> <p>y11 ≐ rebecca'</p> <p>sleep'(f11) act'(f11) ≐ z11</p>	

3.3 Crossing dependencies

- (1) dat Karl Maria Peter zag laten zwemmen

/zag/		/laten/	
e:Δ ○:	CAT:v TYP:fin	g:Δ ○:	CAT:v TYP:inf
x:∇ ⊗:	CAT:n CASE:nom	u:Δ ○:	CAT:n CASE:★
y:∇ ⊗:	CAT:n CASE:acc	v:∇ ⊗:	CAT:n CASE:acc
z:∇ ⊗:	CAT:n CASE:★	w:∇ ⊗:	CAT:n CASE:★
f:∇ ⊗:	CAT:v TYP:inf	h:∇ ⊗:	CAT:v TYP:inf
• _r		=	
e		g	
<p>see'(e) act'(e) ≐ x pat'(e) ≐ y thm'(e) ≐ f y ≐ z</p>		<p>let'(g) act'(g) ≐ u pat'(g) ≐ v thm'(g) ≐ h v ≐ w</p>	

/zag laten/

e1 : Δ \emptyset :	CAT:V TYP:fin	}
x1 : ∇ \emptyset :	CAT :n CASE:nom	
y1 : ∇ \emptyset :	CAT :n CASE:acc	}
v2 : ∇ \emptyset :	CAT :n CASE:acc	
w2 : ∇ \emptyset :	CAT :n CASE:★	}
h2 : \blacktriangledown \emptyset :	CAT:V TYP:inf	
e1, f1		
see'(e1)		
act'(e1) \doteq x1		
pat'(e1) \doteq y1		
thm'(e1) \doteq f1		
y1 \doteq z1		
let'(f1)		
act'(f1) \doteq z1		
pat'(f1) \doteq v2		
thm'(f1) \doteq h2		
v2 \doteq w2		

/zag laten/

e1:Δ ⊙:	CAT:V TYP:fin
x1:∇ ⊙:	CAT :n CASE:nom
y1:∇ ⊙:	CAT :n CASE:acc
v2:∇ ⊙:	CAT :n CASE:acc
w2:∇ ⊙:	CAT :n CASE:★
h2:∇ ⊙:	CAT:V TYP:inf

e1, f1

see'(e1)
act'(e1) ≐ x1
pat'(e1) ≐ y1
thm'(e1) ≐ f1
y1 ≐ z1

let'(f1)
act'(f1) ≐ z1
pat'(f1) ≐ v2
thm'(f1) ≐ h2
v2 ≐ w2

/zweimmen/

g:Δ ⊙:	CAT : v TYP : inf
y:Δ ⊙:	CAT : n CASE : ★

•_r

=

g

swim'(g)
act'(g) ≐ y

/zag laten zweimmen/

e11:Δ ⊙:	CAT:V TYP:fin
x11:∇ ⊙:	CAT :n CASE:nom
y11:∇ ⊙:	CAT :n CASE:acc
v21:∇ ⊙:	CAT :n CASE:acc

e11, f11, h21

see'(e11)
act'(e11) ≐ x11
pat'(e11) ≐ y11
thm'(e11) ≐ f11
y11 ≐ z11

let'(f11)
act'(f11) ≐ z11
pat'(f11) ≐ v21
thm'(f11) ≐ h21
v21 ≐ w21

swim'(h21)
act'(h21) ≐ w21

/dat Karl Maria Peter zag laten zwemmen/

e11:Δ ○:	CAT:V TYP:fin	
x11:-○:	CAT :n CASE:nom	}
y11:-○:	CAT :n CASE:acc	
v21:-○:	CAT :n CASE:acc	
e11, f11, h21, v21, y11, x11		
<p>see'(e11) act'(e11) ≐ x11 pat'(e11) ≐ y11 thm'(e11) ≐ f11 y11 ≐ z11</p> <p>let'(f11) act'(f11) ≐ z11 pat'(f11) ≐ v21 thm'(f11) ≐ h21 v21 ≐ w21</p> <p>swim'(h21) act'(h21) ≐ w21</p> <p>v21 ≐ peter' y11 ≐ maria' x11 ≐ karl'</p>		