Appendix

A coordination algorithm

(1) **INPUT**

(a) An array of Types $S_{\#n} = [S_{\#1}, ..., S_{\#n}]$ such that $n \geq 3$
(b) An array of Types $S_{\text{co}+fr} = [S_{\text{co}+fr_1}, ..., S_{\text{co}+fr_k}]$ such that $k \leq n$
(c) A set of triples of Types REDSET such that $|\text{REDSET}| = n - k$

{\{ The input is supposed to be the output of a Reduction Grammar as defined at the onset of chapter 3. Constituents are identified by their types and their positions in the input; the span is suppressed. \}}

(2) **OUTPUT**

A set COLEVELSET of pairs of pairs of integers and arrays of Types

{\{ The set stores combinations of left and right hand coordination levels, each provided with an index marking the scope of coordination at that side. If this set ends up empty, no grammatical coordination has been established. \}}

(3) **TYPE**

A Type is a symbol in the set of linear types

{\{ A linear type is defined in chapter 3, section 2 \}}

(4) **FUNCTIONS**

(a) Decompose

The function Decompose from Types to pairs of Types is the function $h$ such that $h(c) = (a, b)$ iff $(a, b, c)$ is in REDSET and $h(c) = ()$ (the empty pair) otherwise.

{\{ The function instantiates definition 207 \}}

(b) Insert Replace

The function Insert Replace from pairs of Types $(c_a, c_b)$ indices $i$ and $n$-place arrays $A$ to $(n+1)$-place arrays $A'$ is the function $f$ such
TOWARDS AN ALGORITHM FOR COORDINATION

that \( f(<C_1,C_2>,i,A) = A' \) iff for all \( j < i \), \( A_j = A'_j \) and \( A'_{i+1} = C_1 \) and for all \( j > i+1, A'_{j} = A_{j-1} \)

{{ The function is used for the construction of a string out of another by replacing one particular constituent by its decomposition. The function embodies the Decomposition Protocol (257) }}

(5) CONSTANTS

(a) Axis

\( AXIS \) is the lowest integer \( i \) such that \( S-co+fr_i = COORD \)

{{ \( AXIS \) identifies the relative position of the first coordinator in the input array \( S-co+fr \). The algorithm, however, only provides a solution for strings with at one coordinator }}

(b) L-co+fr

\( L-co+fr \) is the subarray \([S-co+fr_1,...,S-co+fr_{AXIS-1}]\) of \( S-co+fr \)

(c) R-co+fr

\( R-co+fr \) is the subarray \([S-co+fr_{AXIS+1},...,S-co+fr_n]\) of \( S-co+fr \)

(6) VARIABLES

\( S+CO_L \) and \( S+CO_R \) are sets of pairs of integers and arrays of Types

{{ The sets store for partial results in the search for coordination levels }}

(7) PROCEDURE: Find_Determinant

output: \( L_DET, R_DET, L_DET_POS, R_DET_POS \)

{{ This procedure fixes the Determinant, i.e., the central pair of constituents in the resolution of a coordination, and the positions of its members at \( S-co+fr \) }}

(a) FUNCTIONS

(a.1) Rleftc

\( rleftc \) is the function applied to a pair of Types, yields an integer such that

\[
\begin{align*}
    rleftc(y)(x/y) &= rleftc(y)(x\backslash y) = 0 \\
    rleftc(y)(x\backslash y) &= rleftc(y)(x) + 1
\end{align*}
\]

(a.2) Rrightc
**APPENDIX**

**rightc** is the function that applied to a pair of Types, yields an integer such that

\[ \text{rightc}(y)(x/y) = \text{rightc}(y)(x/z) = 0 \]
\[ \text{rightc}(y)(x) + 1 \]

\{ These two functions instantiate the notion of Relative Complexity (245) \}

(a.3) **Satisfy**

**Satisfy** is the function that applied to a pair of Types \( c_i \) and \( c_j \), yields the value **true** if \( c_i \) satisfies \( c_j \), and the value **false** otherwise

\{ The function reflects the definition of potential satisfaction in clause (159) \}

BEGIN

SET **Satpair** to the set of pairs of Types such that for each \( i \) and \( j \)

\( <L-co+fr_i, R-co+fr_j> \) is in that set IFF \( \text{Satisfy}(L-co+fr_i, R-co+fr_j) = \text{true} \)

\{ **Satpair** is the set of Satisfaction Pairs in S-co+fr as defined in chapter 3 \}

SET **Iospset** to that subset of **Satpair** such that \( <L-co+fr_i, R-co+fr_j> \) is in **Iospset** IFF for no \( i \)'s < \( i \) and no \( j \)'s > \( j \) OR for no \( i \)'s > \( i \) and \( j \)'s < \( j \)

\( <L-co+fr_i, R-co+fr_j> \) is in **Satpair**

\{ This clause determines the Innermost Outermost Satisfaction Pairs according to definition (228). Evidently, **Iospset** contains at least one and at most two members if **Satpair** is not empty \}

SET **Func_Iospset** to the set of those Types that are the first or the second member of a pair in **Iospset** and that \( f \)-satisfy the other member of the pair.

\{ Selects the \( f \)-satisfiers in **Iospset** \}

SET **Inner_Iospset** to the set of those Types \( L-co+fr_i \) and \( R-co+fr_j \)

such that for no \( i \)'s < \( i \) \( L-co+fr_i \) is the first member of a pair in **Iospset** and for no \( j \)'s > \( j \) \( R-co+fr_j \) is the second member of a pair in **Iospset**

\{ Selects the innermost **Types** in **Iospset** \}

SET **Comarker** to that Type that is in the intersection of **Func_Iospset** and **Inner_Iospset**
TOWARDS AN ALGORITHM FOR COORDINATION

{{ Determines the unique coordination marker, according to definition (239) }}

SET compl to the integer such that
IF for some i Comarker = L-co+fr_i and Comarker = x/y
THEN compl = rright(y)(Comarker)
ELSE if Comarker = x/y, then compl = rleft(y)(Comarker)

{{ Determines the relative complexity of Comarker }}

SET Comarker_sp to that ordered subset of Satpair such that for all i and for all j
<L-co+fr_i R-co+fr_j> is in Comarker_sp IFF either Comarker = L-co+fr_i or Comarker = R-co+fr_j
AND
IF for some i Comarker = L-co+fr_i THEN the set is ordered by decreasing j of R-co+fr_i ELSE the set is ordered by increasing i of L-co+fr_i

{{ Selects the relevant satisfaction pairs in an ordered way, from outermost to innermost }}

SET Det to the compl-th pair of Types in Comarker_sp

{{ Selects the determinant according to (247) }}

SET L_DET and R_DET to the member of L-co+fr and the member of R-co+fr in Det, respectively

SET L_DET_POS to the integer i such that L_DET = S-co+fr_i

SET R_DET_POS to the integer j such that R_DET = S-co+fr_j

{{ This fixation of the index of the members of the determinant with respect to the input string S-co+fr is motivated by the locality of coordination, theorem (256). The ordering will be L_DET_POS < AXIS < R_DET_POS }}

END.

(8) PROCEDURE: L_Compare

input: a pair (two-place array) of Types: L_Supplier
an array of Types: Images
an integer i
an array of Types: Co_String
output: the set S+CO_L of pairs of integers and arrays of Type
The procedure is to compare decompositions at the left to potential coordination-images at the right. L_Searcher is a decomposition pair, occurring in a left-hand-side Co_String at a position marked by the integer. Images is the array of categories in R-co+frs, not yet imaged. The variable S-co_left is the storage for left coordinates and their peripheral strings. The integer marks the index of the left border. Storing the left coordinates separated from the right coordinates (see the procedure R_Compare below) is allowed by clause (259b). By the Locality of Coordination Lemma (256), different coordinations only differ "vertically", which means that further decomposition at one side does not affect established coordinates at the other.}

BEGIN

IF L_Searcher is the empty pair OR Images is the empty array
THEN true

THEN

ELSE IF L_Searcher \[2\] = last(Images)

THEN IF length(Images) = 1

THEN add \(<i, \text{Insert}_\text{Replace}(L_{\text{Searcher}}, i, \text{Co_String})>\) to S+CO_L

THEN add \(<i, \text{Insert}_\text{Replace}(L_{\text{Searcher}}, i, \text{Co_String})>\) to S+CO_L

THEN

AND

L_Compare(Decompose(L_Searcher \[2\]), Images, i+1, Insert_Replace(L_Searcher, i, Co_String))

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THEN

THE
TOWARDS AN ALGORITHM FOR COORDINATION

$\text{Decompose}(L_{\text{Searcher}}), Images = Images, i = i+1, Co_{\text{String}} = Insert_{\text{Replace}}(L_{\text{Searcher}}, i, Co_{\text{String}})\}

\text{ELSE L}_{\text{Compare}}(\text{Decompose}(L_{\text{Searcher}}), Images - \text{last}(Images), i, Insert_{\text{Replace}}(L_{\text{Searcher}}, i, Co_{\text{String}}))

\{\text{If the right member of L}_{\text{Searcher}} \text{ has a C-image there were it should be in Images but other images are not yet paired, the left member is called for decomposition, according to statement (259d)}\}\}

\text{AND}

\text{L}_{\text{Compare}}(\text{Decompose}(L_{\text{Searcher}}), Images, i+1, Insert_{\text{Replace}}(L_{\text{Searcher}}, i, Co_{\text{String}}))

\{\text{Withdraws the assignment, backtracking as above}\}\}

\text{ELSE L}_{\text{Compare}}(\text{Decompose}(L_{\text{Searcher}}), Images, i+1, Insert_{\text{Replace}}(L_{\text{Searcher}}, i, Co_{\text{String}}))

\{\text{Backtracking is necessary if no proper image was found in Images}\}\}

\text{END.}

(9) \text{PROCEDURE: R}_{\text{Compare}}

\text{output: a pair (two-place array) of Types: R}_{\text{Searcher}}
   \text{an array of Types: Images}
   \text{an integer i}
   \text{an array of Type Co_{String}}
\text{output: the set S+CO}_{R} \text{ of pairs of an integer and an array of Types} 
APPENDIX

{{{ R_Compare operates as L_Compare, under symmetrical adoptions. The parameters of the two procedures are essentially disjoint in extension }}}

BEGIN

IF R_Searcher or Images is the empty pair cq. the empty array

THEN halt

ELSE IF R_Searcher, = first(Images)

THEN IF length(Images) = 1

THEN add <i, Insert_Replace(R_Searcher, i, Co_String)> to the set S+CO_R

AND

R_Compare(Decompose(R_Searcher), Images, i, Insert_Replace(R_Searcher, i, Co_String))

ELSE R_Compare(Decompose(R_Searcher), Images, i, Insert_Replace(R_Searcher, i, Co_String))

AND

R_Compare(Decompose(R_Searcher), Images - first(Images), i+1, Insert_Replace(R_Searcher, i, Co_String))

ELSE R_Compare(Decompose(R_Searcher), Images, i, Insert_Replace(R_Searcher, i, Co_String))

END.
TOWARDS AN ALGORITHM FOR COORDINATION

(10) PROCEDURE: Fix_Coord
output: the sets of pairs of integers and arrays of Types
S+CO_L and S+CO_R

{{ The procedure takes care of Coordination Resolution (196) in case one or both
members of the innermost satisfaction pair qualify for decomposition, as pointed at in the
Coordination Lemma (250). It activates and steers the initial run of the procedures
L_Compare and R_compare. The output stores the left and right coordinates found—see
also L_Compare}}

BEGIN

SET R_Images to the subarray [R-co+fr, ... R-co+fr R_DET_POS . AXIS .] of
R-co+fr
SET L_Images to the subarray [L-co+fr AXIS . L_DET_POS .] ... L-co+fr AXIS .] of
L-co+fr

{{ The two variables are motivated by the Resolution Statement (259). They contain the
categories in S-co+fr that are within the scope of coordination at S+co and have to be
C-imaged by decomposition. One of them may be empty, in case a member of the
determinant is adjacent to COORD }}

IF L_DET_POS = AXIS - 1

{{ Inspecting the position of the left-wing det member: is it adjacent to COORD? }}
THEN

L_Compare(
    Decompose(L_DET),
    R_Images,
    L_DET_POS,
    L-co+fr )
AND

add <R_DET_POS, R-co+fr> to S+CO_R

{{ Following clause (250b): no decomposition at the right wing; the right coordinate is
fixed }}

ELSE IF R_DET_POS = AXIS + 1

{{ Is the right member of det a neighbour to COORD? }}
THEN

R_Compare(
    Decompose(R_DET),
    L_Images,
    R_DET_POS,
    R-co+fr )


8
APPENDIX

\[
\text{ADD} \quad \text{add} <L_{\text{DET} \_ \text{POS}}, L_{\text{co} + \text{fr}}> \text{ to } S + CO _ {L} \\
\{ \{ \text{If so, the left wing is fixed for coordination and the right decomposing and comparison takes place} \} \}
\]

\text{ELSE} \quad L_{\text{Compare}}( \\
\quad \text{Decompose}(L_{\text{DET}}), \\
\quad R_{\text{Images}}, \\
\quad L_{\text{DET} \_ \text{POS}}, \\
\quad L_{\text{co} + \text{fr}}) \\
\text{AND} \quad R_{\text{Compare}}( \\
\quad \text{Decompose}(R_{\text{DET}}) \\
\quad L_{\text{Images}}, \\
\quad R_{\text{DET} \_ \text{POS}} \\
\quad R_{\text{co} + \text{fr}}) \\
\{ \{ \text{Otherwise, there is simultaneous decomposition at the left and the right, according to (250)} \} \}
\]

\text{END.}

(\text{11) PROGRAM}

\text{BEGIN} \\
\text{IF length}(L_{\text{co} + \text{fr}}) = \text{length}(R_{\text{co} + \text{fr}}) = 1 \\
\{ \{ \text{A special case} \} \} \\
\text{THEN} \quad \text{IF} \quad L_{\text{co} + \text{fr}} = s \quad \text{AND} \quad R_{\text{co} + \text{fr}} = s \\
\{ \{ \text{Most special; cf. statement (199)} \} \} \\
\text{THEN} \quad \text{add} <1, L_{\text{co} + \text{fr}}> \text{ to } S + CO _ {L} \text{ AND add} <1, R_{\text{co} + \text{fr}}> \text{ to } S + CO _ {R} \\
\{ \{ \text{Coordination found, of course} \} \} \\
\text{ELSE} \quad \text{IF} \quad L_{\text{co} + \text{fr}} \neq s \quad \text{AND} \quad R_{\text{co} + \text{fr}} \neq s \\
\text{THEN} \quad \text{SET} \text{ COLEVELSET } \text{ to the empty set} \\
\{ \{ \text{If the two strings have each just one member and none is of type } s, \text{ there cannot be any coordination according to statement (220)} \} \}
\]
TOWARDS AN ALGORITHM FOR COORDINATION

ELSE IF \( L_{co+fr_1} = s \)
THEN \( L_{\text{Compare}}(\) Decompose\( (L_{co+fr_1})\), \( R_{co+fr_1}\), \( l\), \( L_{co+fr}\) )
AND
add \( <1, R_{co+fr}> \) to \( S+CO_R \)
\{\} There cannot be a determinant. Yet, decomposition is called for; cf. statement 215.
The right side is fixed \}\}

ELSE \( R_{\text{Compare}}(\) Decompose\( (R_{co+fr_1})\), \( L_{co+fr_1}\), \( l\), \( R_{co+fr}\) )
AND
add \( <1, L_{co+fr}> \) to \( S+CO_L \)
\{\} The former case reversed \}\}

ELSE IF length\( (L_{co+fr}) = 1 \)
THEN \( L_{\text{Compare}}(\) Decompose\( (L_{co+fr_1})\), \( R_{co+fr_1}\), \( l\), \( L_{co+fr}\) )
AND
add \( <\text{length}(R_{co+fr}), R_{co+fr}> \) to \( S+CO_R \)
\{\} On behalf of statements \((200), (215), (216), (219)\) and \((220)\), theorem \((250)\) forces us to apply decomposition to a single category and to fix the other string if it contains more than one member \}\}

ELSE IF length\( (R_{co+fr}) = 1 \)
THEN \( R_{\text{Compare}}(\) Decompose\( (R_{co+fr_1})\), \( L_{co+fr_1}\), \( l\), \( R_{co+fr}\) )
AND
add \( <\text{length}(L_{co+fr}), L_{co+fr}> \) to \( S+CO_L \)
\{\} The same case, the other way around \}\}
APPENDIX

\textbf{ELSE Find Determinant}
\textbf{Fix Coord}

\{\textit{Normal business, finally}\}

\textbf{SET} \textbf{COLEVELSET} to the cartesian product of \textit{S+CO_L} and \textit{S+CO_R}

\{\textit{The mission is over}\}

END.