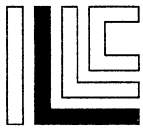
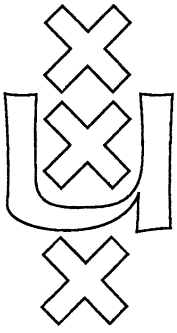


Institute for Logic, Language and Computation

AN ALGEBRAIC VIEW ON ROSETTA

Theo M.V. Janssen

ILLC Prepublication Series
for Computational Linguistics CL-93-02



University of Amsterdam

The ILLC Prepublication Series

1990 *Logic, Semantics and Philosophy of Language*

LP-90-01 Jaap van der Does — A Generalized Quantifier Logic for Naked Infinitives
LP-90-02 Jeroen Groenendijk, Martin Stokhof — Dynamic Montague Grammar
LP-90-03 Renate Bartsch — Concept Formation and Concept Composition
LP-90-04 Aarne Ranta — Intuitionistic Categorial Grammar
LP-90-05 Patrick Blackburn — Nominal Tense Logic
LP-90-06 Gennaro Chierchia — The Variablity of Impersonal Subjects
LP-90-07 Gennaro Chierchia — Anaphora and Dynamic Logic
LP-90-08 Herman Hendriks — Flexible Montague Grammar
LP-90-09 Paul Dekker — The Scope of Negation in Discourse, towards a Flexible Dynamic Montague grammar
LP-90-10 Theo M.V. Janssen — Models for Discourse Markers
LP-90-11 Johan van Benthem — General Dynamics
LP-90-12 Serge Lapierre — A Functional Partial Semantics for Intensional Logic
LP-90-13 Zhisheng Huang — Logics for Belief Dependence
LP-90-14 Jeroen Groenendijk, Martin Stokhof — Two Theories of Dynamic Semantics
LP-90-15 Maarten de Rijke — The Modal Logic of Inequality
LP-90-16 Zhisheng Huang, Karen Kwast — Awareness, Negation and Logical Omniscience
LP-90-17 Paul Dekker — Existential Disclosure, Implicit Arguments in Dynamic Semantics

Mathematical Logic and Foundations

ML-90-01 Harold Schellinx — Isomorphisms and Non-Isomorphisms of Graph Models
ML-90-02 Jaap van Oosten — A Semantical Proof of De Jongh's Theorem
ML-90-03 Yde Venema — Relational Games
ML-90-04 Maarten de Rijke — Unary Interpretability Logic
ML-90-05 Domenico Zambella — Sequences with Simple Initial Segments
ML-90-06 Jaap van Oosten — Extension of Lifschitz' Realizability to Higher Order Arithmetic, and a Solution to a Problem of F. Richman
ML-90-07 Maarten de Rijke — A Note on the Interpretability Logic of Finitely Axiomatized Theories
ML-90-08 Harold Schellinx — Some Syntactical Observations on Linear Logic
ML-90-09 Dick de Jongh, Duccio Pianigiani — Solution of a Problem of David Guaspari
ML-90-10 Michiel van Lambalgen — Randomness in Set Theory
ML-90-11 Paul C. Gilmore — The Consistency of an Extended NaDSet

Computation and Complexity Theory

CT-90-01 John Tromp, Peter van Emde Boas — Associative Storage Modification Machines
CT-90-02 Sieger van Denneheuvel, Gerard R. Renardel de Lavalette — A Normal Form for PCSJ Expressions
CT-90-03 Ricard Gavaldà, Leen Torenvliet, Osamu Watanabe, José L. Balcázar — Generalized Kolmogorov Complexity in Relativized Separations
CT-90-04 Harry Buhrman, Edith Spaan, Leen Torenvliet — Bounded Reductions
CT-90-05 Sieger van Denneheuvel, Karen Kwast — Efficient Normalization of Database and Constraint Expressions
CT-90-06 Michiel Smid, Peter van Emde Boas — Dynamic Data Structures on Multiple Storage Media, a Tutorial
CT-90-07 Kees Doets — Greatest Fixed Points of Logic Programs
CT-90-08 Fred de Geus, Ernest Rotterdam, Sieger van Denneheuvel, Peter van Emde Boas — Physiological Modelling using RL
CT-90-09 Roel de Vrijer — Unique Normal Forms for Combinatory Logic with Parallel

Other Prepublications

X-90-01 A.S. Troelstra — Conditional, a case study in conditional rewriting
X-90-02 Maarten de Rijke — Remarks on Intuitionism and the Philosophy of Mathematics, Revised Version
X-90-03 L.D. Beklemishev — Some Chapters on Interpretability Logic
X-90-04 — On the Complexity of Arithmetical Interpretations of Modal Formulae
X-90-05 Valentin Shehtman — Annual Report 1989
X-90-06 Valentin Goranko, Solomon Passy — Derived Sets in Euclidean Spaces and Modal Logic
X-90-07 V.Yu. Shavrukov — Using the Universal Modality: Gains and Questions
X-90-08 L.D. Beklemishev — The Lindenbaum Fixed Point Algebra is Undecidable
X-90-09 V.Yu. Shavrukov — Provability Logics for Natural Turing Progressions of Arithmetical Theories
X-90-10 Sieger van Denneheuvel, Peter van Emde Boas — On Rosser's Provability Predicate
X-90-11 Alessandra Carbone — An Overview of the Rule Language RL/1
X-90-12 Maarten de Rijke — Provable Fixed points in $IA_0+\Omega_1$, revised version
X-90-13 K.N. Ignatiev — Bi-Unary Interpretability Logic
X-90-14 L.A. Chagrova — Dzhaparidze's Polymodal Logic: Arithmetical Completeness, Fixed Point Property, Craig's Property
X-90-15 A.S. Troelstra — Undecidable Problems in Correspondence Theory
Lectures on Linear Logic

1991 *Logic, Semantics and Philosophy of Language*

LP-91-01 Wiebe van der Hoek, Maarten de Rijke — Generalized Quantifiers and Modal Logic
LP-91-02 Frank Veltman — Defaults in Update Semantics
LP-91-03 Willem Groeneveld — Dynamic Semantics and Circular Propositions
LP-91-04 Makoto Kanazawa — The Lambek Calculus enriched with Additional Connectives
LP-91-05 Zhisheng Huang, Peter van Emde Boas — The Schoenmakers Paradox: Its Solution in a Belief Dependence Framework
LP-91-06 Zhisheng Huang, Peter van Emde Boas — Belief Dependence, Revision and Persistence
LP-91-07 Henk Verkuyl, Jaap van der Does — The Semantics of Plural Noun Phrases
LP-91-08 Víctor Sánchez Valencia — Categorial Grammar and Natural Reasoning
LP-91-09 Arthur Nieuwendijk — Semantics and Comparative Logic
LP-91-10 Johan van Benthem — Logic and the Flow of Information

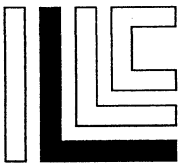
Mathematical Logic and Foundations

ML-91-01 Yde Venema — Cylindric Modal Logic
ML-91-02 Alessandro Berarducci, Rineke Verbrugge — On the Metamathematics of Weak Theories
ML-91-03 Domenico Zambella — On the Proofs of Arithmetical Completeness for Interpretability Logic
ML-91-04 Raymond Hoofman, Harold Schellinx — Collapsing Graph Models by Preorders
ML-91-05 A.S. Troelstra — History of Constructivism in the Twentieth Century
ML-91-06 Inge Bethke — Finite Type Structures within Combinatory Algebras
ML-91-07 Yde Venema — Modal Derivation Rules
ML-91-08 Inge Bethke — Going Stable in Graph Models
ML-91-09 V.Yu. Shavrukov — A Note on the Diagonalizable Algebras of PA and ZF
ML-91-10 Maarten de Rijke, Yde Venema — Sahlqvist's Theorem for Boolean Algebras with Operators
ML-91-11 Rineke Verbrugge — Feasible Interpretability
ML-91-12 Johan van Benthem — Modal Frame Classes, revisited

Computation and Complexity Theory

CT-91-01 Ming Li, Paul M.B. Vitányi — Kolmogorov Complexity Arguments in Combinatorics
CT-91-02 Ming Li, John Tromp, Paul M.B. Vitányi — How to Share Concurrent Wait-Free Variables
CT-91-03 Ming Li, Paul M.B. Vitányi — Average Case Complexity under the Universal Distribution Equals Worst Case Complexity
CT-91-04 Sieger van Denneheuvel, Karen Kwast — Weak Equivalence
CT-91-05 Sieger van Denneheuvel, Karen Kwast — Weak Equivalence for Constraint Sets
CT-91-06 Edith Spaan — Census Techniques on Relativized Space Classes
CT-91-07 Karen L. Kwast — The Incomplete Database
CT-91-08 Kees Doets — Levationis Laus
CT-91-09 Ming Li, Paul M.B. Vitányi — Combinatorial Properties of Finite Sequences with high Kolmogorov Complexity
CT-91-10 John Tromp, Paul Vitányi — A Randomized Algorithm for Two-Process Wait-Free Test-and-Set
CT-91-11 Lane A. Hemachandra, Edith Spaan — Quasi-Injective Reductions
CT-91-12 Krzysztof R. Apt, Dino Pedreschi — Reasoning about Termination of Prolog Programs

CL-91-01 J.C. Scholtes — *Computational Linguistics* — Kohonen Feature Maps in Natural Language Processing
CL-91-02 J.C. Scholtes — Neural Nets and their Relevance for Information Retrieval
CL-91-03 Hub Prüst, Remko Scha, Martin van den Berg — A Formal Discourse Grammar tackling Verb Phrase Anaphora



Institute for Logic, Language and Computation
Plantage Muidergracht 24
1018TV Amsterdam
Telephone 020-525.6051, Fax: 020-525.5101

AN ALGEBRAIC VIEW ON ROSETTA

Theo M.V. Janssen

Department of Mathematics and Computer Science
University of Amsterdam

ILLC Prepublications
for Computational Linguistics
ISSN 0928-3293

Coordinating editor: Dick de Jongh

To appear as Chapter 19 in
M.T. Rosetta, *Compositional Translation*
Kluwer, Dordrecht

received September 1993

AN ALGEBRAIC VIEW ON ROSETTA

19.1. INTRODUCTION

The principle of compositionality of translation, introduced in chapter ??, is the main principle of the Rosetta method and constitutes an important theme of this book. In the present chapter a mathematical model of compositional translation will be developed. The character of this chapter is somewhat different from the previous two chapters. Chapters ?? and ?? provide a formal treatment of the Rosetta system from two different perspectives, whereas the present chapter is more at distance, in the sense that it does not primarily deal with the system, but with its method. The model that will be presented will be abstract, using several notions from universal algebra, but the main example of this model will be of course the Rosetta system.

Developing a mathematical model of compositionality has several advantages. To mention some:

1. It gives a new perspective on the Rosetta system, in which other facets become interesting. The algebraic point of view links the Rosetta framework with Montague grammar and with model theoretic semantics.
2. It relates the Rosetta method to well-investigated mathematical theory. Thus it is possible to use tools from mathematics and to prove results concerning the method. It makes it possible to investigate whether additions made to the ideas of chapter ?? (e.g. control, introduced in chapter ??) constitute a relaxation to the method or not (see section 19.4).
3. It describes what are the essential ingredients for a compositional translation system, and it abstracts from aspects that are specific to the Rosetta system. Thus it becomes possible to recognise the compositional method in other situations; for instance, when other syntactic theories are used, or, when the involved languages are not natural languages (see section 19.5).
4. It offers a framework for the investigation of the power of the method and to answer questions such as: does compositionality form a restriction on the languages that can be dealt with? Or: what is the role of reversibility and of the measurement condition? (see section 19.6).

This chapter is organised as follows. In section 19.2 basic universal algebra is presented, which is used in section 19.3 to develop an algebraic model for compositional translation. In section 19.4 the Rosetta system is considered in the perspective of this model, and other instances of the model are mentioned in section 19.5. Finally, section 19.6 discusses aspects related to the formal power of the method.

19.2. BASIC UNIVERSAL ALGEBRA

This section will introduce several notions from universal algebra, the main ones being *algebra, generator, term, homomorphism,* and *isomorphism.* These notions will be used in the next section to define a mathematical model of compositional translation. The definitions in this section are standard (see e.g. Graetzer(1986)), except for the definition of homomorphism that deviates slightly from the usual one.

19.2.1. *Algebras*

An algebra is, intuitively speaking, a set with some operations defined on it. The relevant terminology is given in the following definitions.

Definitions:

An *algebra* A consists of a set A, called the *carrier* of the algebra, and a set of operators F defined on that set. So an algebra A is a pair $\langle A, F \rangle$, where A is a set and for each $f \in F$ holds $f \subset A^n \times A$. The *elements* of the algebra are by definition the elements of its carrier. An *n-ary operator* is an operator that takes n arguments, and a *partial operator* is an operator that is not defined on the whole carrier.

The notion 'set' is a very general notion, and so is the notion 'algebra' which depends on it. In order to get used to this abstract notion, some examples of a different nature will be given below.

1. The algebra with as carrier the finite strings formed from $\{a, b, ..., z\}$ with concatenation of two strings

as operation.
2. The algebra with as carrier the natural numbers $\{0, 1, 2, ...\}$ and with addition and division as operators. Here is division a partial operator since 3 cannot be divided by two if only natural numbers are available.
3. The algebra of all points in the $X - Y$ plane with as operation assigning to three points the point that has equal distance to all three points. Its operation is a 3-ary, and it is a partial operator since it is only defined for three points that do not lie on one straight line.

In order to avoid the misconception that everything is an algebra, an example of a non-algebra might be useful. Consider the first example: an algebra with finite strings as carrier, and with concatenation as operation. Add an operator that determines the length of a string. Then the result is not an algebra any more, since the lengths (natural numbers) are not elements of the carrier.

19.2.2. *Generators*

Often, the elements of an algebra can be obtained from a small subset using operations of the algebra. For instance, in the algebra of strings with concatenation as operation every string can be obtained starting from $\{a, b, .., z\}$ by application of the concatenation operator. These symbols are called the *generators* of the algebra, and the set $\{a, b, ..., z\}$ is called a *generating set.* In the algebra $\langle N^+, + \rangle$ of positive natural numbers with addition as operation is the set $\{1\}$ is a generating set since $1 + 1 = 2$, $(1 + 1) + 1 = 3$, etc..

Starting with the set $\{a, b\}$ and only using the concatenation operator, the set of all strings of as and bs is obtained. This is an algebra too, because the concatenation of two strings consisting of as and bs is again a string consisting of as and bs. Because its carrier forms a subset of all strings over $\{a, b, ..., z\}$, it is called a subalgebra of the algebra of all strings. An algebra which has a generating set B and a collection of operators F is denoted as $\langle [B], F \rangle$. So $\langle [\{a, b\}], concatenation \rangle$ is the algebra of all strings over $\{a, b\}$. And $\langle [\{2\}], + \rangle$ is the algebra of even numbers.

Definitions:

Let $A = \langle A, F \rangle$ be an algebra, and H be a subset of A. Then $\langle [H], F \rangle$ denotes the smallest algebra containing H, and is called the *subalgebra generated* by H. If $\langle [H], F \rangle = \langle A, F \rangle$, then H is called a *generating set* for A, its elements are called *generators of* A.

19.2.3. *Terms*

It often is important to know in which way an element is formed. In some kinds of algebra it seems difficult to describe this process, in particular when the elements are abstract (e.g. points in the plane). The general method to represent a generation process is as follows. Names for the generators are introduced (say, $a_1, a_2, ...$) and names for the rules (say, $f_1, f_2, ...$). Suppose f_1 is the name for a binary operator. Then the expression $f_1\langle a_1, a_2 \rangle$ denotes a derivation where f_1 is applied to a_1 and a_2. Such an expression is called a term. This term can occur as a subterm in a larger expression, for example in $f_2\langle f_1\langle a_1, a_2 \rangle, a_3 \rangle$. This term can also be represented as a tree as in figure 1. The terms (derivations, derivational histories) form

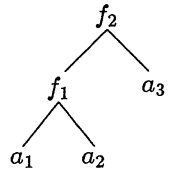


Figure 1.

an algebra themselves. The carrier consists of all the terms. The operators of this algebra combine small terms to larger ones. To continue the example given above, there is an operator F_{f_1}, which takes as inputs the terms a_1 and a_2 and yields the term $f_1\langle a_1, a_2 \rangle$. So for each operator in the original algebra there is precisely one operator in the algebra of terms. Hence, given an algebra A there is a corresponding term algebra (denoted as T_A). A term corresponds with an element in the original algebra in the obvious way (apply the corresponding operators to the obtained arguments). In a term algebra the operations are defined on the whole carrier, whereas in the corresponding algebra operators may be partial.

Definition:

Let $A = \langle [B], F \rangle$ be an algebra. Introduce for each element in B a distinct symbol b, and for each operator in F a distinct symbol f. Then $\mathrm{T}_{B,F}$, the set of terms over $\langle [B], F \rangle$, is defined as follows:

1. for each element in B the corresponding symbol $b \in \mathrm{T}_{B,F}$
2. if f corresponds with an n-ary operator, and if $t_1, t_2, ..., t_n \in \mathrm{T}_{B,F}$, then $f\langle t_1, t_2, ..., t_n \rangle \in \mathrm{T}_{B,F}$.

In case we do not want to be explicit about the set of generators we may use the algebra itself as subscript (as in T_A).

19.2.4. *Homomorphisms and Isomorphisms*

A homomorphism h from an algebra A to an algebra B is, intuitively speaking, a mapping which respects the structure of A in the following sense. If in A an element a is obtained by means of applying an operator f to an element a' then the image of a can be obtained in B by application of an operator corresponding with f to the element corresponding with a'. The structural difference that may arise between A and B is that two distinct elements of A may be mapped onto the same element of B, and that two distinct operators of A may correspond with the same operator in B. An isomorphism is a homomorphism for which such a structural difference does not arise, so an isomorphism has an inverse that also is a homomorphism.

Definitions:

Let $A = \langle A, F \rangle$ and $B = \langle B, G \rangle$ be algebras.

A mapping $h : A \cup F \rightarrow B \cup G$, where $h(A) \subseteq B$ and $h(F) \subseteq G$ is called a *homomorphism* if for all $f \in F$ and all $a, ..., a_n \in A$ holds: $h(f(a_1, ..., a_n)) = h(f)(h(a_1), ..., h(a_n))$.

An *isomorphism* $h : A \cup F \rightarrow B \cup G$ is a homomorphism such that there is a homomorphism $g : B \cup G \rightarrow A \cup F$ with the property that for all $a \in A : g(h(a)) = a$.

The above definitions of homomorphism and isomorphism differ slightly from the definitions in the literature. Here, the correspondence of the rules is incorporated in the notion of homomorphism, it is standard to assume this correspondence as given. The two types of definition differ in case two different homomorphisms are defined on the same algebra, a situation that will not arise here. An advantage of the given definition is that it makes sense to speak about the image of an operator (and in later sections about the translation of a rule).

Homomorphisms (and, consequently, isomorphisms) have a property that is very important in applications, and will be appealed on several times. It is the property that a homomorphism is uniquely determined by its value for the operators and generator. This means that a homomorphism on an infinite domain can be defined by finite means. This is expressed in theorem 1.

Theorem 1 *Let $A = \langle A, F \rangle$ and $B = \langle B, G \rangle$ be algebras, where D is the generating set of A.*

Let $h : A \cup F \rightarrow B \cup G$ be such that $h(D) \subset B, h(F) \subset G$, and suppose that for each f its image $h(f)$ has the same number of arguments as f. Then there is a unique extension of h from D to A.

Sketch of Proof Suppose h_1 and h_2 are extensions of h. Then it can be shown by induction that for all $a \in A : h_1(a) = h_2(a)$. **End of Proof**

19.2.5. *Polynomials*

It is useful to have a method available to introduce new operations in an algebra using already available operations. A simple example is composition: if f and g are operators which take one argument, then $f \circ g$ is defined by first applying f to the argument, and next applying g to the result. So for all a $f \circ g(a) = g(f(a))$.

Another example concerns the algebra of natural numbers with $+$ and $\times$ as operators. The new operator takes two arguments and is represented by the expression $x_1 \times x_1 + x_2 \times x_2$ (or equivalently, ${x_1}^2 + {x_2}^2$). The operator assigns to the arguments 1 and 2 (given in this order) the value $1 \times 1 + 2 \times 2$, i.e. 5, and it assigns to the arguments 2 and 3 the value $2 \times 2 + 3 \times 3$, i.e. 13. An expression like ${x_1}^2 + {x_2}^2$ is called a polynomial. Given two arguments, the value yielded by the polynomial ${x_1}^2 + {x_2}^2$ is obtained by substituting the first argument for x_1, the second argument for x_2, and performing the calculations which are indicated in the expression.

This method of defining new operations by means of polynomials can be used in every algebra, the relevant formal definitions are given below.

Definitions:

A *polynomial* is a term in which indexed variables for elements of the algebra may occur (terms are special polynomials: polynomials without variables). A polynomial symbol p defines a *polynomial operator*: its value for given arguments is found by evaluating the term that is obtained by replacing x_1 by the first argument, x_2 by the second, etc.

Given an algebra $A. = \langle [B], F \rangle$, and a set P of polynomial symbols over $A.$, we obtain a new algebra $\langle [B], P \rangle$ by replacing the original set of operators by the set of polynomial operators defined by P. An algebra obtained in this way is called a *polynomially derived algebra.*

19.3. AN ALGEBRAIC MODEL FOR COMPOSITIONAL TRANSLATION

19.3.1. *Introduction*

This section will give a mathematical model which characterises what AREthe essential aspects of a compositional translation system. Starting point of the discussion is the principle of compositionality of translation from chapter ??. For ease of discussion it is repeated here:

Principle of compositionality of translation

> Two expressions are each other's translation if they are built up from parts which are each other's translation, by means of rules with the same meaning.

The mathematical model will be introduced step by step, and will be illustrated by string grammars resembling those given in chapter ??, but now in an algebraic fashion.

19.3.2. *Syntax*

The compositionality principle speaks about *parts*, and the mathematical model should have a formal definition of this notion. Since the rules of the syntax determine how expressions are formed, we let the syntax determine what are the parts of an expression. For this purpose, the rules should take inputs and yield an output, and we define the *parts* of an expression E as those expressions from which E is formed by means of some rule. This means that the rules of syntax can be regarded as operators in an algebra.

The parts of an expression can again be expressions, and the principle is intended to hold for these parts as well. So there can be a chain of 'parts of parts of parts..'. Since the principle is intended to give a constructive approach to translating, this chain should have an end. These final parts can be taken as the generators of the algebra.

A summarising the above discussion: in a compositional translation system the syntax of source and target language are generated algebras. The parts of an expression E are the expressions that can be used as input for an operator yielding output E.

As an example the grammar $G_{English}2$ from chapter ?? will be presented as an algebra. For convenience's sake, the grammar is repeated here:

$G_{English}2$:

1. Basic expressions:
   N(*girl*), N(*boy*), ADJ(*intelligent*), ADJ(*brave*),
   IV(*cry*), IV(*laugh*)
2. Rules:
   R_E1: N(α) + ADJ(β) $\Rightarrow$ NP(β αs)
   R_E2: IV(α) $\Rightarrow$ VP(α)
   R_E3: IV(α) $\Rightarrow$ VP(*do not* α)
   R_E4: VP(α) + NP(β) $\Rightarrow$ S(β α)

The carrier of the algebra $A_{English}2$ consists of strings (e.g. N(*boy*) or S(*intelligent boys cry*)), its generators are the basic expressions, and its operators are the rules. For instance, rule R_E1: N(α) + ADJ(β) $\Rightarrow$ NP(β αs) can be considered as a partial operator that takes two strings of a certain form as arguments, and yields one string as output. Using the algebraic notation introduced before, we may represent the algebra as:

$A_{English}2 = \langle$[N(*girl*), N(*boy*), ADJ(*brave*),IV(*cry*), IV(*laugh*)],
$\{R_E1, R_E2, R_E3, R_E4\}\rangle$,

where the effects of R_E1, etc., are as defined above.

19.3.3. *Translations*

The principle of compositionality expresses that the way in which expressions are formed from basic parts, give the complete information needed for determining its translation. In algebraic terminology it means that the terms over an algebra form the domain for the translation relation. The principle also states that the translations of an expression are formed in an analogous way from the translations of parts. So also the range of the translation relation consists of terms. An example of a term over the algebra $A_{English}5$ (see section 19.3.2), is $R_E4\langle R_E3\langle IV(laugh)\rangle, R_E1\langle N(girl),ADJ(brave)\rangle\rangle$, which represents the derivation of S(*brave girls do not laugh*). The D-tree for this sentence is given in chapter ?? and gives precisely the same information (viz. which basic expressions are used, and which rules).

Next, we consider the *nature* of the translation relation between two languages A and B. For clarity's sake, we assume for the moment the simplification that in A and B each basic expression and each rule has precisely one translation. So there are no synonyms or ambiguities on the level of terms, and the translation relation between T_A and T_B is a function. Structural or derivational ambiguities of expressions of A and B are still possible.

Let us consider a simple example: algebra A, with a two-place operator f, and an algebra B in which the operator g corresponds with f. Let Tr_{AB} denote the translation function from T_A and T_B. Then the principle of compositionality of translation tells that the translation of the term $f\langle a1, a2\rangle$ is obtained from $Tr_{AB}(a_1)$ and $Tr_{AB}(a_2)$ by means of the operation g. So

$Tr_{AB}(f\langle a_1, a_2\rangle) = g\langle Tr_{AB}(a1), Tr_{AB}(a2)\rangle =$

$Tr_{AB}(f)\langle Tr_{AB}(a_1), Tr_{AB}(a_2)\rangle$

This means that Tr_{AB} is a homomorphism.

For the reverse translation the same argumentation holds. So Tr_{BA}, the reverse translation function, is an homomorphism as well:

$Tr_{BA}(g\langle b_1, b_2\rangle) = Tr_{BA}(g)\langle Tr_{BA}(b_1), TrBA(b_2)\rangle$.

Note that a translation followed by a reverse translation yields the original term:

$Tr_{BA}(Tr_{AB}(f\langle a_1, a_2\rangle)) = Tr_{BA}(Tr_{AB}(f)\langle Tr_{AB}(a_1), Tr_{AB}(a_2)\rangle) =$

$Tr_{BA}(Tr_{AB}(f))\langle Tr_{BA}(Tr_{AB}(a_1)), Tr_{BA}(Tr_{AB}(a_2))\rangle = f\langle a_1, a_2\rangle$

A homomorphism of which the inverse is a homomorphism as well, by definition is an isomorphism. Hence Tr_{AB} is an isomorphism.

We may summarise the formalisation obtained so far as follows. For a compositional translation system without synonyms and ambiguities at the level of terms holds: the translation relation is an isomorphism between the term algebra over the source language and the term algebra over the target language.

As an example, we will consider a translation into Dutch of the expressions that are generated by $A_{English}2$. In order to obey the assumptions made in this section (no ambiguities on the level of terms) the grammar $G_{Dutch}2$ from chapter ?? is modified in one respect: there is only one translation for *cry* viz. *huilen*. This grammar ($G_{Dutch}5$) is as follows:

$G_{Dutch}5$:

1. Basic expressions:
   N(*meisje*), N(*jongen*), ADJ(*dapper*), IV(*huilen*), IV(*lachen*)
2. Rules:
   R_D1: N(α) + ADJ(β) $\Rightarrow$ NP(βe αs)
   R_D2: IV(α) $\Rightarrow$ VP(α)
   R_D3: IV(α) $\Rightarrow$ VP(α niet)
   R_D4: VP(α) + NP(β) $\Rightarrow$ S(β α)

An algebraic presentation of this grammar that is isomorphic with the algebra $A_{English}2$ given in section 19.3.2 is $A_{Dutch}5 = \langle$[N(*meisje*), N(*jongen*), ADJ(*dapper*), IV(*huilen*), IV(*lachen*)], {R_D1, R_D2, R_D3, R_D4}$\rangle$, where the effects of R_D1, etc. are as defined above. The first generator of $A_{Dutch}5$ corresponds with the first of $A_{English}2$, the second generator with the second, etc., and the same goes for the operators. Hence the translation isomorphism maps the term

$R_E4\langle R_E3\langle IV(laugh)\rangle, R_E1\langle N(girl), ADJ(brave)\rangle\rangle$

on the term

$R_D4\langle R_D3\ \langle IV(lachen)\rangle, R_D1\langle N(meisje), ADJ(dapper)\rangle\rangle$.

In figure 2 the isomorphism between term algebras for $A_{English}2$ and $A_{Dutch}5$ is illustrated by indicating the values for some of the terms in these algebras.

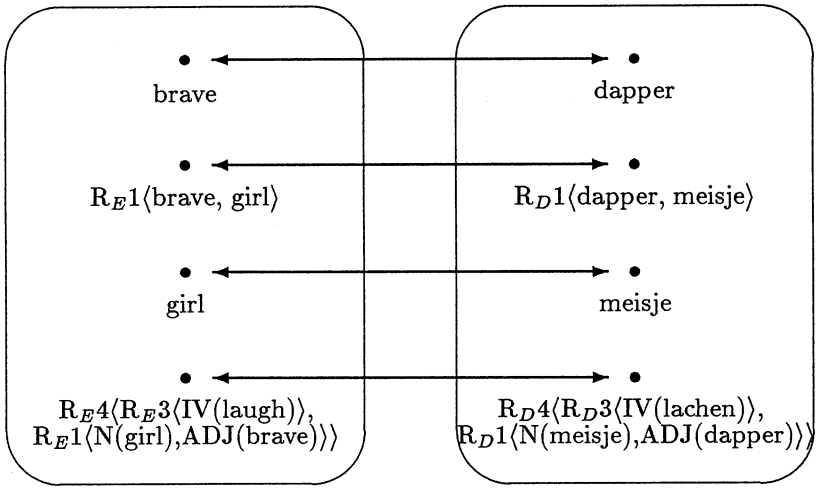


Figure 2. A translation isomorphism between some terms in $T_{English}2$ and in $T_{Dutch}5$.

19.3.4. *Sets of synonymous translations*

Next, we allow that there are synonyms of basic expressions and of rules. Then the translation relation is not a function from T_A to T_B because a term may have several translations. But we can see the relation as a function yielding a set of synonymous terms. In the source language algebra the same can be done, and then the translation relation relates sets of synonymous terms in T_A with such sets in T_B. These collections of sets have the structure of an algebra. The result of the application of a set of operators to a set of terms, is defined as the union of application of all operators to all terms. Figure 3 shows which situation arises if the grammar from figure 2 is extended with the synonyms *courageous* for *brave* and *moedig* for *dapper*. One may notice that in figure 3 for the sets of synonymous terms the same situation arises as in figure 2 for the terms themselves. This suggests that *the translation function is an isomorphism between the algebras of synonymous terms in T_A and T_B*. The suggestion can be proven to be correct using a theorem on isomorphisms that is not presented in the previous section: since synonymy is a congruence relation, it induces an isomorphism on the corresponding quotient algebra (see Graetzer(1986)).

19.3.5. *Ambiguities in translations*

Finally, the consequences of ambiguous basic expressions and rules are considered. The introduction of ambiguities on the term level is an essential difference with the traditional situation in Montague grammar (e.g Montague(1970, 1973) where the terms uniquely determine the meaning of an expression and form an disambiguated language. However, a more recent development (called 'flexible Montague grammar') allows for ambiguous terms as well. See, for example, Partee and Rooth(1983), Hendriks(1987), and Groenendijk and Stokhof(1989).

Ambiguous basic expressions and rules may disturb the symmetry between source languageand target language because an ambiguity in the one language needs not to correspond to an ambiguity in the other language. An example in the grammar $G_{English}2$ from chapter ?? is *cry*, which can be translated into Dutch by *schreeuwen* (meaning *shout*) and *huilen* (*weep*). We consider three ways to conceive ambiguous terms in algebraic perspective:

1. The ambiguities of basic expressions (and of rules) form an exception to the situation that the grammars of source and target algebra are isomorphic algebras. This point of view might be useful if there are few exceptions.
2. Consider the translation relation as a function which yields a set of possible translations (not necessarily synonymous). So the translation of the basic expression cry is the set {*schreeuwen, huilen*}, and the translation of a term is a set of terms. The translation function is a homomorphism from T_A to sets of elements in T_B. This perspective seems attractive if one is interested in only one direction of a translation.

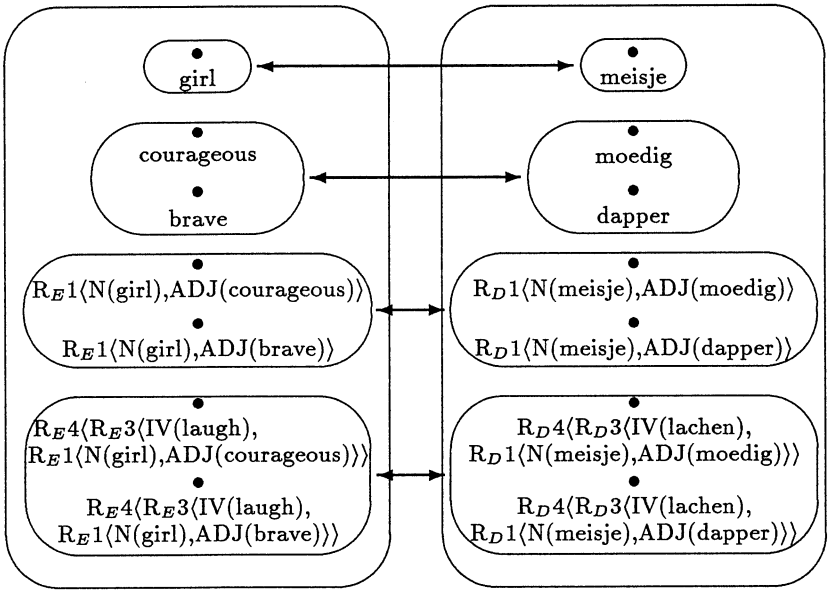


Figure 3. A translation isomorphism between some sets of synonymous terms in $T_{English}2$ and in $T_{Dutch}5$.

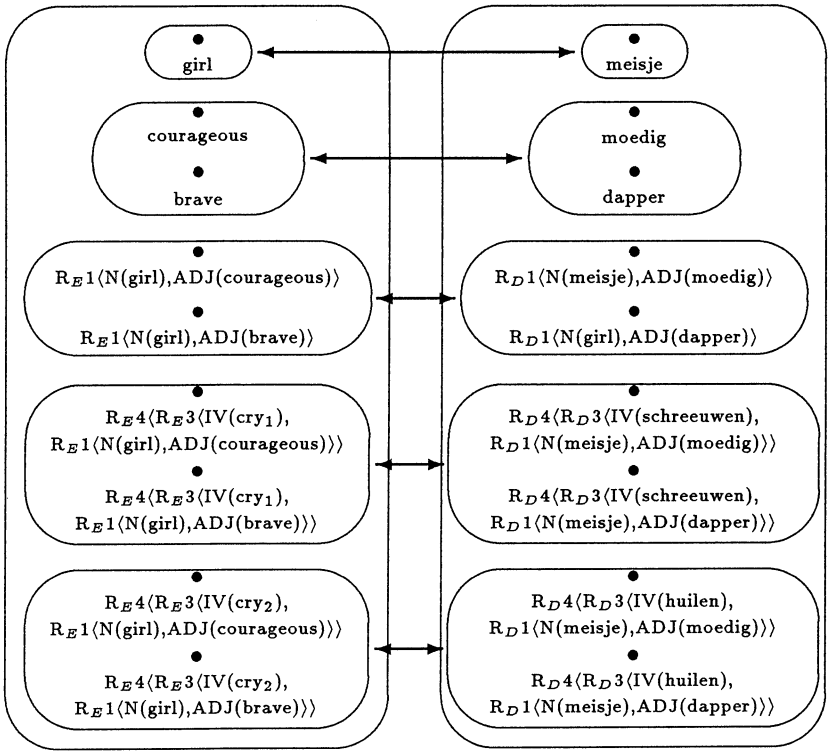


Figure 4. The translation isomorphism between some sets of synonymous terms after resolution of the ambiguity of cry.

3. The expression *cry* is not a basic expression of the term algebra, but it is a convenient notation for the set consisting of the basic expressions cry_1 (*shout*) and cry_2 (*weep*) , and every term containing *cry* is a notation for a set of terms with cry_1 and cry_2. The translation relation is then an isomorphism between sets of such unambiguous terms, see figure 4. This perspective is the most appropriate for the Rosetta system, as will be explained in the next section.

19.3.6. *Summary*

The algebraic model for compositional translation is as follows:
A compositional translation system between languages A and B consists of:

1. *Generated algebras as syntax for A and B. The parts of an expression E are the expressions that can be used as input for an operator yielding output E.*
2. *A translation relation that is defined between T_A and T_B and consists of an isomorphism between sets of synonymous terms in T_A and such sets in T_B.*

19.4. ROSETTA AND THE ALGEBRAIC MODEL

The present section will show that the grammars of Rosetta form an instance of the algebraic model. Furthermore, it will discuss some issues that are interesting from the algebraic point of view. The attention is restricted to the kernel of the grammar: the syntactic component and to the translation relation, whereas morphology and surface syntax are not considered.

19.4.1. *Free M-grammars as algebras*

The formalisation of the principle of compositionality of translation requires that the syntax of the languages in the system is organised as an algebra. This is the case for free M-grammars: they form an algebra, with all possible S-trees as carriers, and the M-rules as its operators. The incorporation of control will be considered in section 19.4.2. The generators of the Rosetta algebra are its basic expressions; in section 19.4.4 we will consider them in more detail.

The formalisation of compositional translation requires that the translation relation is defined on the level of terms over the syntactic algebra. The syntactic D-trees are a representation of the terms, and the translation relation is indeed defined on this level.

An aspect of the Rosetta system that does not necessarily follow from compositionality is its interlingua. This interlingua is also an algebra. Its generators are the semantic keys (each corresponding to a set of synonymous basic expressions in the source language as well as with such a set in the target language. For the operators, the situation is analogous. In the interlingua there are no ambiguous terms. This gives an argument to conceive ambiguous terms over the syntactic algebras as denoting a set of unambiguous terms (the third solutions mentioned in section 19.3.5): in the transfer to the interlingua this set is split. So using ambiguous terms is a method to postpone ambiguity resolution to the stage where the differences in meaning really matter. In this perspective, the transfer to the interlingua is an isomorphism between sets of terms over the source language algebra to terms over the interlingua. Also the transfer to the target language is an isomorphism from such terms in the interlingua to sets of synonymous terms over the target language. Since the composition of these two isomorphisms is an isomorphism again, the two isomorphisms could be replaced by a single one. This illustrates that the interlingua is not an essential aspect of the system.

19.4.2. *Control as conditions for application*

As chapters ?? and ?? have shown, the rules in the subgrammars of a controlled M-grammar cannot be applied freely; they have to be used in some controlled order. Algebras, however, do not have such a control mechanism: an operator can be applied as soon as its arguments satisfy the conditions on its input. One might think that the addition of control is something that gives additional power to the algebra, or, alternatively, that it is a restriction since some strict order has to be specified. This section will show that the introduction of control neither constitutes a restriction on the algebra, nor it gives additional power. For this, two facts have to be shown: 1) that an algebra without control can be seen as a special algebra with control, and 2) that an algebra with control can be formulated as one without control.

First, consider an algebra without control, so with free order of application of the operators. Suppose the algebra has operators R_1,R_2, ...,R_n. Then free application is described by the control expression $\{R_1|R_2|, ..., |R_n\}$. Thus, 'without control' is a special case of 'with control'.

Consider next an algebra with control. With this, the same effects can be obtained as without control. The idea is to extend the elements with information concerning their position in the course of the controlled generation process. Then the application conditions of the operators take care of the rule ordering. This idea is worked out in the following example.

Suppose we have, as in M-grammars, an algebra in which the elements are trees. We will enrich the top node with an attribute for control. Suppose the following control expression is given:

(1) $(R_1).\{R_2\}.[R_3]$

The possible values of the attribute are all strings which can be obtained by inserting a $*$ somewhere in this control expression. The initial value of the attribute for control is

(2) $*(R_1).\{R_2\}.[R_3]$

The operator R_1 is only applicable to inputs that have this initial value for the control attribute. By application of R_1 the value of the attribute is changed into

(3) $(R_1) * \{R_2\}.[R_3]$

For R_2 two operators are introduced, R_{2a} and R_{2b}. Both require (3) as input value. Then R_{2a} applies R_2 and leaves the value of the attribute unchanged, whereas R_{2b} only changes it into

(4) $(R_1).\{R_2\} * [R_3]$

For R_3 we introduce two operators, viz. R_{3a} and R_{3b}. The effect of R_{3a} is the effect of applying R_3 and changing the control attribute into

(5) $(R_1).\{R_2\}.[R_3]*$

The effect of R_{3b} consists in changing the control attribute into (5), without application of R_3.

This example shows that an algebraic grammar with a control expression can be simulated by an uncontrolled one. In M-grammars the situation is somewhat more complex, because there is a control expression for each of the subgrammars. In order to incorporate the subgrammar structure, the attribute for control has to be enriched with information about the subgrammar this control expression originates from. Then the result can be recognised as such, and may be imported into other subgrammars. Furthermore, in a free grammar each rule has to satisfy the measure condition, whereas in a controlled M-grammar this only holds for the iterative rules. So a measure has to be designed together with the above construction. For this purpose we might take an unorthodox measure, which for the non-iterative rules looks at the position of the $*$ in the attribute for control (the more to the right, the more complex), and for iterative rules in addition uses the measure given with such a rule. This point completes the discussion of the example that illustrates that a controlled grammar can be simulated by a free grammar.

19.4.3. *Rules and transformations as polynomials*

In chapter ?? syntactic transformations were introduced. These have no semantic effects on expressions, and are not relevant for defining the translation. Therefore, they are ignored in the definition of isomorphism between source language derivations and target language derivations. This suggests that the introduction of transformations is a relaxation of compositionality of translation. Somers(1990), quoting Carroll(1989), even stated that the introduction of transformations makes 'a complete mockery' of the statement that grammars are isomorphic. However a change of perspective shows that it fits perfectly into the algebraic framework, and hence it is an instance of the compositional approach.

The simplest solution would be to assume that for each transformation in the source language algebra there is an operation in the target language algebra which gives its input unchanged as output: a syntactic identity rule. The same would be done in the source language for all transformations from the target language algebra (see chapter ??). Mentioning these rules explicitly in the translation relation does not yield any interesting information, therefore they can be omitted. This perspective is completely in accordance with the formalisation of compositionality of translation as given before, but it introduces another problem.

If a syntactic identity rule is applied, it might be applied indefinitely many times. So an expression can have infinitely many derivations, and obtaining all derivation trees is impossible. Hence parsing is impossible. Related to this objection, is that the identity rules do not obey the measure condition (because the complexity of the output of the rule equals the complexity of its input). This measure condition, together with reversibility, is a prerequisite for the parsing method of M-grammars. Since the measure condition is not satisfied, the solution with the identity rules is not in accordance with the restrictions imposed on the Rosetta grammars. But another perspective on transformations is possible in which the measure condition is not violated.

The second method is based upon polynomially derived algebras. This is illustrated by means of a simple example. Suppose that source algebra A has the following control expression:

$R_1.[T_1].R_2.\{T_2\}$

where R_1 and R_2 are rules and T_1 and T_2 are transformations. Then consider this algebra with control, as an algebra A' without control, derived from A, and where the operators are defined by the polynomial

symbols

R_1,R_1T_1,R_2,R_2T_2,$R_2T_2T_2$,$R_2T_2T_2T_2$,...

As a result, the transformations have been connected with the preceding rules to form a polynomial operator. This choice is arbitrary, we might as well connect them with subsequent rules. If the control expression is more complicated, e.g. with more transformation classes mentioned between R_1 and R_2, then these classes define a set of possible sequences of applications of transformations. Each of these sequences can be connected with R_1 to form a polynomially derived operator.

In the target algebra we will use the same method to define new operators. Suppose the control expression is

$T_3.R_3.[T_4].R_4$

Here R_3 is the rule that corresponds with R_1, and R_4 with R_2. This control expression defines the derived operators:

T_3R_3,$T_3R_3T_4$ and R_4.

The derived algebras have a correspondence of operators: all derived rules containing R_1 correspond with all rules containing R_3, and all derived rules containing R_2 correspond with all rules containing R_4.

In this perspective, the control expression is a finite method to define an infinite set of polynomial operators of the derived algebra A'. If the transformations and rules the polynomial operators are composed of, satisfy the reversibility and measure condition, the polynomials do so as well. This shows that the transformations are not a relaxation of compositionality: they are a method to define a derived algebra with polynomial operations.

It might not be clear that the derived algebra can be parsed because, in the above example, the transformation T_2 is responsible for an infinite number of derived operators. However, only a finite number of derived rules has to be applied. Each application in analysis of a transformation decreases the complexity by at least 1. So if an input expression has complexity measure m, only sequences $R_2T_2T_2...T_2$ with length $\leq m$ have to be tried, and this is a finite number of derived rules. Note that in M-grammars there is another situation in which an infinite number of rules is available for parsing. When a variable has to be introduced in analysis by a substitution rule, there are in principle infinitely many variable indices possible. Here a related solution is used: the rule is only applied for one new (conventionally determined) index.

19.4.4. *Lexicon as generating set*

The formalisation of compositionality requires that some suitable subset of expressions is selected as the set of generators. In mathematics it often is required that a generating set is *minimal*, or *finite*. These two properties will be considered below. Furthermore, the definition of lexicon for idioms will be looked at.

A generating set is *minimal* if no elements of the set can be missed. In some applications one aims at using minimal generating sets, for instance in geometrical applications because the dimension of the object in some cases equals the minimal number of generators. The generating set of the Rosetta algebras is not minimal. The grammars produce, for instance, an S-tree for *John kicked the bucket* from two different sets of generators. Either from the elements {*John, bucket, kick*} in S-LEX, or from *(kick the bucket)* in ID-DICT and *(John)* in S-LEX. If the idiom were not in the lexicon, then still the same S-tree could be formed. This is done because in the translation process idioms are considered as basic expressions.

In applications one often aims at a finitely generated algebra, because such an algebra can be specified by finite means. The generating set of the Rosetta algebra is infinite, because an unlimited number of variables is available. This infinity causes no problem for a finite specification of the translation relation because the translation of these infinitely many elements are completely regular (a variable is translated into a variable with the same index). There are infinitely many different analyses of a sentence which differ only with respect to the choice of a variable. As indicated in section 19.4.3, this causes no serious problems.

It is interesting to see that idioms and translation idioms are defined in the lexicons by polynomial symbols. For instance, the translation idiom *cocinar* is translated into *prepare a meal*, and this complex expression is defined by giving the keys of the basic words and of the rules which form it, hence by giving the term which describes the derivation of the complex expression. Idioms with an open position give rise to polynomials with variables (e.g. *to pull* x*'s leg*). Representing idioms by means of polynomials is done for a practical reason: to guarantee that the S-trees that are assigned to them are suitable input for later rules (see chapter ??).

19.4.5. *The translation relation as a homomorphism*

The translation relation in a system for compositional translation is a homomorphism (in fact a special one: an isomorphism). Due to the properties of homomorphisms, there are several advantages.

1. A homomorphism is fully determined by its values for the generators and operators. This justifies the way in which the Rosetta system translations are defined: by giving the translations of generators and operators. Thus the infinite translation relation is reduced to a finite relation between grammars.
2. The translation is defined for the same operators and generators as used in the syntax to define the language. Thus it is guaranteed that if an expression is generated in the syntax, it is automatically accepted as input for the translation. This situation is in contrast with a situation in which the generated language is defined by one mechanism and the translation by another. Then there is no guarantee that a generated expression is suitable as input for translation (cf. the intersection problem discussed in chapter ??).
3. The last, but important, advantage concerns the correctness of the translations. As was already explained in chapter ??, generators and rules of the Rosetta grammars are designed in such a way that they represent a (basic) meaning or an operation on meanings. So there is an (implicit) algebra of meanings, and the assignment of a meaning to terms is a homomorphism. Hence the meaning of a term is uniquely determined by the meanings of generators and operators. Furthermore, the generators and operators of the target language have the same meaning as the corresponding generators of the source language. So the algebra of meaning for the source language is identical to the algebra of meanings for target the language. Consequently a term of the target language has the same meaning as the corresponding meaning in the source language. So, due to the fact that translation is a homomorphism, the correctness of the translation follows from the correctness of the translations of the generators and operators.

19.4.6. *Conclusions*

We have considered several aspects of the Rosetta system from an algebraic perspective. It turned out that M-grammars satisfy the algebraic model also in those cases where this was not evident at first (e.g. control and transformations).

19.5. OTHER EXAMPLES OF COMPOSITIONAL TRANSLATION

19.5.1. *Related work*

The algebraic model as described above, can be found outside the context of the Rosetta system. In some cases this is not surprising since there is a common background, such as Montague Grammar and compositionality, or other ideas originating from Rosetta publications. Below we mention some examples where the same algebraic model is used, most of them have already been mentioned in chapter ?? in another context.

1. Dowty(1982) noticed the analogy of derivation trees for small fragments of English, Japanese, Breton and Latin, and proposed such derivation trees for translation. For Latin (a language with free word order) the grammar would not generate strings (or structures), but (unordered!) sets of words.
2. Tent(1990) uses Montague grammar to give a method for investigating a typology of languages. She does so by comparing grammatical rules in different languages for the same meaning operation. In fact she designs (very small) isomorphic grammars or English, Japanese, Indonesian and Finnish.
3. Rupp(1986) attempts to write isomorphic grammars for a machine translation system with generalised phrase structure rules.
4. The CAT-framework of Eurotra is a (not implemented) proposal for the design of the Eurotra system, the translation project of the EC (e.g. Arnold *et al.*(1985), Arnold *et al.*(1986), Des Tombe(1985) and Arnold and Des Tombe(1987)). This proposal is based upon a variant of the principle of compositionality of translation, viz. *The translation of a compound expression is a function of the translations of its parts.* This formulation of compositionality is also given by Nagao(1989). The algebraic model of this principle gives rise to a model in which translation is a homomorphism (see Janssen(1989)). It can be turned into an isomorphism, by introducing sets in the way indicated in section 19.3.5.

19.5.2. *Other proposals that are compositional*

For some proposals it might not be immediately obvious that the compositional approach is followed. Then an algebraic reformulation of a proposal might make the analogy evident. Two examples of this are given below.

A TAG grammar is a tree grammar: its basic expressions are trees, and the operations are operations on trees. Such an operation may for instance replace, under certain conditions, an internal node by a subtree. Abeill *et al.*(1990) propose to use synchronised TAG grammars, for translating. A synchronised TAG grammar consists of two TAG grammars. The basic trees of the two grammars are given in pairs: a tree in one grammar with the corresponding tree in the other grammar. The positions in which the expansions may take place are 'synchronised': for each node that may be expanded in one tree, a link with a node in the other tree is given. A derivation starts with two coupled basic expressions, and continues by expanding two linked nodes. This means that the grammars are isomorphic and derivations are made in parallel. So, synchronised TAG grammars are an instance of compositional translation.

Some theories of language do not have constituent structures as a notion in their theory. For instance, Dik's functional grammar (Dik(1978)) is inspired by logic: predicate frames constitute the skeleton of the syntax and rules operate on such frames (e.g. attaching a modifier or substitution of an argument). Van der Korst(1989) proposed a translation system based upon functional grammar, and, to a large extent, this can be seen as a compositional translation system. The predicate frames filled with arguments and with modifiers attached, can be considered terms in an algebra, where the frames and modifiers are operations.

That these rather divergent theories fit into the algebraic framework may give rise to the misconception that this holds for all syntactic theories. A counter-example is given by the approach which says that the strings of a language have a structure and that this can be any structure that meets certain well-formedness conditions. So there are no rules, only conditions. Such a system was proposed by McCawley(1986). Since there are no rules, there is no algebra with operations, and a compositional translation system in the sense of the principle cannot be designed.

A somewhat related situation arises in the theory of 'Principles and Parameters', because, there too, the conditions are the central part of the theory. Formally, the situation is slightly different because the crucial part of the grammar is a rule (called move-alfa) which can move any constituent to any position, and this movement is controlled by many conditions. An algebraic formulation of this theory is possible, with as most important operator the partial operator corresponding with move-alfa. In this case, the algebraic approach is not interesting because the syntactic algebra hardly has any structure, and it is unlikely that a compositional translation system can be based upon this algebra. Note that in M-grammars many generalisations are used which were proposed within the 'Principles and Parameters' theory, but in a much more structured way than with one movement rule.

19.5.3. *Other applications of compositional translation*

Languages that exist next to natural languages are programming languages and logical languages. Translations are made between such languages, as well. We will consider some examples of this, illustrating that the compositional approach, which is innovative for translations between natural languages, is accepted and more or less standard for other kinds of translations.

1. *From logic to logic.*
   In logic, translations are frequently used to investigate the relation between different kinds of logic, for instance their relative strength or their semantic relation. The standard method of translation is the compositional method. An example is the Gödel translation; this translates intuitionistic logic into modal logic. For instance the intuitionistic implication $\phi \rightarrow \psi$ is translated into $\Box(\phi' \rightarrow \psi')$, where ϕ' and ψ' are translations of ϕ and ψ, respectively. Therefore, the translation of the implication is a polynomial symbol over the syntactic algebra of modal logic.
2. *From natural language to logic.*
   The aim of Montague grammar is to associate meanings with natural language expressions. This is done by translating natural language into intensional logic. The methodological basis of Montague Grammar is the principle of compositionality of meaning, and, therefore, this translation has to be compositional. In many cases a syntactic operator is translated into a compound expression over the logic. For instance, verb phrase disjunction is translated as $\lambda x[\alpha(x) \vee \beta(x)]$, where α and β are the translations of the verb phrases.

3. *From programming language to programming language.*
   Computers have to execute programs written in some programming language, and for this purpose the programming language is translated into machine code. The compiler is the part of the computer software which performs this translation. Often, one instruction from the programming language has to be translated into a compound of machine code instructions. A standard method to perform the translation is the so called 'syntax directed translation'. This method can be seen as a form of compositional translation. The power of the compositional approach is used by Thatcher *et al.*(1979) to design a method which allows to prove the correctness of such a compiler. Following the compositional approach, the problem of proving correctness of an infinite set of possible programs is reduced to proving the correctness of translating the generators and the syntactic constructions.

In all these examples the source algebra and the target algebra were not designed in connection with each other, but with independent motivations. It is, therefore, not surprising that in all cases the source language and the target language have algebraic grammars that are not isomorphic with each other. Nevertheless, the translations are isomorphisms. This is possible because in all cases the translation covers only a subset of the target language. Furthermore in all cases only a subset of the target language arises as output of the translation. For this subset a new algebra is designed using polynomials over the original target language algebra. This derived algebra is then isomorphic to the source language algebra, and a compositional translation is obtained.

19.6. POWER OF COMPOSITIONAL TRANSLATION

19.6.1. *Generative Power*

In this section, the power of the framework with respect to the generated language is considered. First we will look at compositional grammars in which the rules are unrestricted. The theorem below shows that the unrestricted compositional grammars have the same power as Turing machines. The simulation method used in the proof is interesting because it stimulates the discussion in the next section, in which the role of the reversibility and the measurement condition are investigated.

Theorem 2 *Any recursively enumerable language can be generated by a compositional grammar.*

Proof In order to prove the theorem, we will simulate a Turing machine by means of a compositional grammar. For this purpose we take a non-deterministic Turing machine that starts on an empty tape. The machine halts when no instruction is applicable and the string of symbols (neglecting the blanks) is the generated string. The set of all strings it can generate is the language generated by the machine. Our aim is to design a compositional grammar for the same language.

We assume that the Turing machine is of the following type. It operates on a tape that has a beginning but no end, and the machine starts on an empty tape filled with blanks, with its read/write head placed on the initial one. The machine acts on the basis of its memory state q and of the symbol read by the head. It may move to theright (R), to the left (L) or print a symbol, together with a change of memory state. So, two examples of instructions are

1. $q_1 s q_2 R$ (= if the Turing machine is in state q_1 and reads an s, then the state changes to q_2 and the head moves to the right).
2. $q_1 s q_2 t$ (= if the Turing machine reads an s when in state q_1, then it goes into state q_2 and writes a t).

A compositional grammar is of another nature. It has neither memory, nor an infinite tape. But these features of a Turing machine can be encoded by a finite string in the following way. In any stage of the calculations, the head of the Turing machine has passed only a finite number of positions on the tape. That string determines the whole tape, since the remainder is filled with blanks. The current memory state is inserted as an extra symbol in the string on a position to the left of the symbol that is currently scanned by the head.

Each instruction of the Turing machine will be mimicked by an operation of the algebra. Below, this will be done for the two examples mentioned before. Besides this, some additional operations are needed: Operations that add additional blanks to the string if the head stands on the last symbol on the right and has to move to the right, and operations that remove at the end of the calculations the state symbol and the blanks from the string. We will not describe these additional operations in detail.

The first example of the simulation of a Turing machine instruction concerns the instruction $q_1 s q_2 R$ (moving to the right). The corresponding operator F is defined for strings of the form $w_1 q_1 s w_2$ where

w_1 and w_2 are strings consisting of symbols from the alphabet and blanks. The effect of F is defined by $F(w_1q_1sw_2) = w_1sq_2w_2$. The second example concerns writing: q_1sq_2t (replace s by t). The corresponding operator G is defined for strings of the form $G(w_1q_1sw_2) = w_1q_2tw_2$. Since the algebra imitates the machine, the generated language is the same. **End of Proof**

The recursively enumerable languages form the class of languages which can be generated by the most powerful kinds of grammars (unrestricted rewriting systems, transformational grammars, Turing machine languages, etc.), or, more generally, by any kind of algorithm. Theorem 2 shows that if a language can be generated by any algorithm, it can be generated by a compositional grammar. So, compositional grammars can generate any language that can be generated by the other grammars. This means that the method of compositionality of translation does not restrict the class of languages that can be dealt with. This conclusion, however, does not apply to M-grammars because they are not arbitrary compositional grammars, as explained in section 19.6.2.

19.6.2. *Reversibility and complexity*

In order to make parsing possible, the rules of an M-grammar have to obey two restrictions which originate from Landsbergen(1981). For free grammars these conditions are (see also chapter ??):

1. Reversibility
   For each rule R there is a reverse rule R' such that
   $y \in R(x_1, x_2, ..., x_n)$ *if and only if* $(x_1, x_2, ..., x_n) \in R'(y)$
2. Measure condition
   There is a computable function that assigns to an expression a natural number: its measure. The output expression of a generative rule has a measure that is greater than the measures of its input expressions.

The combination of the two conditions guarantees that the output expression of an analytical rule has smaller measure than its input expressions. For controlled M-grammars the measure condition is more complicated (see chapter ??).

The parsing algorithm for M-grammars is based upon the above two conditions. Condition 1 makes it possible to find, given the output of a generative rule, potential inputs for the rule. Condition 2 guarantees termination of the recursive application of this process. Since this parsing algorithm is available, it follows that the generated languages are decidable languages. A well-known result of formal language theory is that the decidable languages form a subset of the class of the recursively enumerable languages. Hence, in the light of theorem 2 it follows that the two restrictions are not only restrictions on the kind of grammars, but also decrease the generative power of the grammars.

Let us investigate the proof to see where the restrictions play a role. The kernel of the proof is the simulation of the Turing machine instructions. The rules which do this are reversible. For instance, if the rule simulating a move of the head to the right, then the reverse rule simulates a move to the left. And if the rule makes the head replace the symbol t by the symbol s, the reverse rule overwrites an s by a t. Therefore, the reversibility condition is satisfied. It is difficult, however, to imagine in which respect the string becomes more complex when the head moves to the right or when a symbol is overwritten. And indeed, if such a measure would exist, the generated language would be decidable (the parsing algorithm sketched above could then be used).

As we have just seen, the grammar used in the proof obeys the reversibility condition but not the measure condition. So, either the combination of the two conditions is responsible for the decrease of generative power, or only the measure condition. The following theorem shows that it is the measure condition.

Theorem 3 *Let G be a free algebraic grammar with a finite number of generators and rules. Suppose G satisfies the measure condition. G then generates a recursive language.*

Sketch of Proof An algorithm deciding whether an expression belongs to the language is as follows. Determine the complexity of the given expression. Then try all generations that are possible with the given grammar, but stop as soon as the generated (intermediate) results have a complexity greater than the complexity of the given expression. Since the grammar satisfies the measure condition, this process terminates. **End of Proof**

This theorem can, under certain conditions, be generalised to infinite sets of generators and for rule schemes. One might, for instance, require that only a finite number of generators of a given complexity is available.

In a controlled M-grammar not all the rules need to satisfy the measure condition. However, a variant of the theorem expressing that an expression can be parsed by generation still holds. There is a measure condition on the relation between import and export conditions of subgrammars that restricts the generation on the level of subgrammars. Within the subgrammars the generation is restricted by the control and the measure for iterative rules.

The results presented in this section show that the measure condition is responsible for the decrease in generative power of the compositional grammars. The reversibility condition is only a restriction on the kind of rules used in the grammar, but is not a restriction on the power. It is a restriction that makes an attractive parsing algorithm possible. It would even be incorrect to claim that the reversibility condition guarantees efficient parsing: both with the algorithm sketched in the proof, and with the algorithm based upon reversibility, the running time can be exponential in the length of the input.

19.6.3. *Translation power*

Next, we investigate the power of the framework with respect to the relation between source language and target language. The first result shows that by means of unrestricted compositional grammars 'any' language can be translated into 'any' language.

Theorem 4 *Let L be a recursively enumerable language, and $Tr : L \to M$ a computable translation function of the expressions of L into M. We will show that there are isomorphic compositional grammars for L and M such that Tr is an isomorphism.*

Sketch of Proof In the proof of theorem 2 the existence of an algebra as syntax for the source language L is proved. For the target language we take a copy of this algebra and extend it with rules that perform the translation. This is possible for the following reason. Since the function Tr is computable there exists a Turing machine computing Tr, and this Turing machine can be simulated by an algebraic grammar. These rules which perform the translation, are considered as transformations of the target language algebra. So, when an expression is generated in the source language algebra, its translation is generated isomorphically in the target language algebra. **End of Proof**

This result shows that compositionality does not restrict the possibilities of translation. It is, of course, good to know this, but the above theorem does not help to find such a translation, because Tr is assumed to be given. The present book argues that a good method is to design isomorphic grammars. The above theorem does not apply to M-grammars because of the reversibility conditions and measure conditions, but the method might be used to obtain a related result for Turing machines satisfying these two conditions.

The last question concerns the problem of strict isomorphy (see chapter ?? for a definition). Formulated in the algebraic terminology the problem is: given two algebras with partial rules, and a isomorphism between the term algebras, is it decidable whether the term for the source language yields an expression if and only if this is the case for the target language? The answer is negative, as is shown below.

Theorem 5 *It is undecidable, given two algebras $\langle A, F\rangle$ and $\langle B, G\rangle$ and a isomorphism $h : A \cup F \to B \cup G$, whether for all $a \in A, f \in F : f(a)$ is defined if and only if $h(f)(h(a))$ is defined.*

Proof Let TM_1 and TM_2 be arbitrary Turing machines. Consider the algebras $A = \langle \{0,1\}^*, \{f\}\rangle$ and $B = \langle \{0,1\}^*, \{g\}\rangle$, where

$$f(w) = \begin{cases} 1 & \text{if } TM_1(w) \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

and

$$g(w) = \begin{cases} 1 & \text{if } TM_2(w) \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Let the isomorphism $h : A \to B$ be defined by $h(f) = g$, $h(0) = 0$ and $h(1) = 1$. Suppose that it is decidable whether $f(w)$ is defined if and only if $g(w)$ is defined. Then it would be decidable whether TM_1 and TM_2 accept the same language. This is known to be undecidable (the problem can be reduced to the halting problem by taking for one of the machines a Turing machine that accepts all strings). Therefore there cannot be a general method to decide whether for two corresponding algebras the one yields an expression

if and only if the other does. **End of Proof**

Of course M-grammars have no rules of which the applicability conditions depend on a Turing machine that generates a recursively enumerable language. The conditions in the rules of the M-grammars are intended to be decidable, but there is no formal restriction guaranteeing this yet. This means that assuming decidable conditions would not be a good model. Furthermore, also for Turing machines that accept decidable languages, equivalence is not decidable.

Since the notion of strict isomorphy is linked so closely to the equivalence of Turing machines, it is unlikely that strict isomorphy can be proved for two grammars if the conditions are independent in the two languages. Only if applicability conditions for corresponding rules (or rule classes) of different languages are in a well defined connection, such a proof might be possible. For some examples where strict isomorphy is proved, see Landsbergen(1987). All the examples can be seen as many sorted algebras with total rules, which are good candidates for the class of algebras for which total isomorphy is decidable.

19.7. CONCLUDING REMARKS

This chapter has presented a mathematical model of compositional translation systems. This model has given us a new point of view on Rosetta, more in particular, an algebraic one. The model was helpful in evaluating certain aspects of the Rosetta system the compositionality of which might be doubted. Some of these turn out to be direct instances of the model (e.g. compound basic expressions), for other aspects (e.g. control and transformations) results from universal algebra were needed to show their compositional nature. Ambiguous basic expressions (and rules) could be viewed in such a way that the translation remains an isomorphism. The model was also useful to give a characterisation of what are the essential aspects of the compositional method, and to recognise the compositional method of translation in other contexts. It has been shown that several fields of science that deal with translations accept compositionality as standard method. Finally, the power of compositional translation and the effects of the restrictions on M-grammars have been investigated.

REFERENCES

A. Abeill, Y. Schabes, and A.K. Joshi. Using lexicalized TAG's for machine translation. In H. Karlgren, editor, *Proceedings COLING 1990*, volume III, pages 1–6, Helsinki Finland, 1990.

D. Arnold and L. des Tombe. Basic theory and methodology in EUROTRA. In S. Nirenburg, editor, *Machine Translation. Theoretical and methodological issues*, pages 114–134. Cambridge University Press, 1987.

D.J. Arnold, L. Jaspaert, R.L. Johnson, S. Krauwer, M. Rosner, L. des Tombe, G.B. Varile, and S. Warwick. A MU1 view of the CAT framework in EUROTRA. In *Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, pages 1–14, Hamilton, NY, 1985. Colgate University.

D.J. Arnold, S. Krauwer, M. Rosner, L. des Tombe, and G.B. Varile. The CAT framework in EUROTRA: a theoretically committed notation for MT. In *Proceedings of Coling 1986*, pages 297–303, 1986.

J.J. Carroll. *Graph Grammars: an approach to transfer-based MT exemplified by a Turkish-English system.* PhD thesis, Centre of Computational Linguistics, UMIST, Manchester, 1989.

S.C. Dik. *Functional grammar.* Number 27 in North Holland Linguistics series. North Holland, Amsterdam, 1978. (third printing, 1981, Foris, Dordrecht).

D. Dowty. Grammatical relations and Montague grammar. In P. Jacobson and G.K. Pullum, editors, *The Nature of Syntactic Representation*, pages 79–130. Reidel, Dordrecht, 1982.

G. Graetzer. *Universal Algebra.* Van Nostrand, Princeton, 1986. (Second edition published by: Springer, New York ,1979).

G. Groenendijk and M. Stokhof. Type-shifting rules and the semantics of interrogatives. In G. Chierchia, B. Partee, and R. Turner, editors, *Semantics issues*, volume II of *Properties, Types and Meaning*, pages 21–68. Kluwer, Dordrecht, 1989.

J.A.G. Groenendijk, T.M.V. Janssen, and M.B.J. Stokhof, editors. *Formal methods in the study of language. Proceedings of the third Amsterdam colloquium.* MC-Tracts 135 and 136. Mathematical Centre, Amsterdam, 1981.

H. Hendriks. Type change in semantics: the scope of quantification and coordination. In E. Klein and J. van Benthem, editors, *Categories, polymorphism and unification.* Institute for Language, Logic and Information, University of Amsterdam, 1987.

T.M.V. Janssen. A mathematical model for the CAT framework of EUROTRA. In *Computerlinguistik und ihre theoretische Grundlagen. Proceedings Symposium Saarbrücken, 1988, Informatik Fachberichte 195*, Berlin, 1989. Springer.

B. van der Korst. Functional grammar and machine translation. In J.H. Conolly and S.C. Dik, editors, *Functional Grammar and the Computer*, number 10 in Functional Grammar Series, pages 290–316. Foris, Dordrecht, 1989.

J. Landsbergen. *Adaptation of Montague grammar to the requirements of parsing*, pages 399–420. Number 136 in MC Tract. Mathematical Centre, Amsterdam, 1981. Philips Research Reprint 7573.

J. Landsbergen. *Isomorphic grammars and their use in the Rosetta translation system.* Edinburgh University Press, 1987. Philips Research M.S. 12.950.

J.D. McCawley. Concerning the base component in a transformational grammar. *Foundations of Language*, 4:55–81, 1986.

R. Montague. Universal grammar. *Theoria*, 36:373–398, 1970. Reprinted in Thomason(1974),pp. 222-246.

R. Montague. The proper treatment of quantification in ordinary english. In K.J.J. Hintikka, J.M.E. Moravcsik, and P. Suppes,

editors, *Approaches to natural language, Synthese Library 49*, pages 221–242. Reidel, Dordrecht, 1973. (Reprinted in Thomason(1974), pp. 247-270).

M. Nagao. *Machine Translation, how far can it go?* Oxford University Press, Oxford, 1989.

B. Partee and M. Rooth. Generalized conjunction and type ambiguity. In R. Baüerle, Ch. Schwarze, and A. van Stechow, editors, *Interpretation of Language*, pages 361–384. De Gruyter, Berlin, 1983.

C.J. Rupp. Machine translation between German and English using logically isom orphic grammars. Master's thesis, University of Sussex, August 1986.

H.L. Somers. Current research in machine translation. In *Proceedings TMI*, Austin, Texas, 1990.

K. Tent. The application of Montague translations in universal research and typology. *Linguistics and Philosophy*, 13:661–686, 1990.

J.W. Thatcher, E.G. Wagner, and J.B. Wright. More on advice on structuring compilers and proving them correct. In H.A. Maurer, editor, *Automata, languages and programming*, number 71 in Lecture notes in computer science. Springer, Berlin, 1979.

B. Thomason. *Formal Philosophy. Selected Papers of Richard Montague.* Yale University Press, New Haven, 1974.

L. des Tombe, D.J. Arnold, L. Jaspaert, R.L. Johnson, S. Krauwer, M. Rosner, G.B. Varile, and S. Warwick. A preliminary linguistic framework for EUROTRA. In *Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, pages 1–4, Hamilton, NY, 1985. Colgate University.

The ILLC Prepublication Series

Other Prepublications

X-91-01 Alexander Chagrov, Michael Zakharyaschev The Disjunction Property of Intermediate Propositional Logics
X-91-02 Alexander Chagrov, Michael Zakharyaschev On the Undecidability of the Disjunction Property of Intermediate Propositional Logics
X-91-03 V. Yu. Shavrukov Subalgebras of Diagonalizable Algebras of Theories containing Arithmetic
X-91-04 K.N. Ignatiev Partial Conservativity and Modal Logics
X-91-05 Johan van Benthem Temporal Logic
X-91-06 Annual Report 1990
X-91-07 A.S. Troelstra Lectures on Linear Logic, Errata and Supplement
X-91-08 Giorgie Dzhaparidze Logic of Tolerance
X-91-09 L.D. Beklemishev On Bimodal Provability Logics for Π_1-axiomatized Extensions of Arithmetical Theories
X-91-10 Michiel van Lambalgen Independence, Randomness and the Axiom of Choice
X-91-11 Michael Zakharyaschev Canonical Formulas for K4. Part I: Basic Results
X-91-12 Herman Hendriks Flexibele Categoriale Syntaxis en Semantiek: de proefschriften van Frans Zwarts en Michael Moortgat
X-91-13 Max I. Kanovich The Multiplicative Fragment of Linear Logic is NP-Complete
X-91-14 Max I. Kanovich The Horn Fragment of Linear Logic is NP-Complete
X-91-15 V. Yu. Shavrukov Subalgebras of Diagonalizable Algebras of Theories containing Arithmetic, revised version
X-91-16 V.G. Kanovei Undecidable Hypotheses in Edward Nelson's Internal Set Theory
X-91-17 Michiel van Lambalgen Independence, Randomness and the Axiom of Choice, Revised Version
X-91-18 Giovanna Cepparello New Semantics for Predicate Modal Logic: an Analysis from a standard point of view
X-91-19 Papers presented at the Provability Interpretability Arithmetic Conference, 24-31 Aug. 1991, Dept. of Phil., Utrecht University

1992 *Logic, Semantics and Philosophy of Langauge* Annual Report 1991

LP-92-01 Víctor Sánchez Valencia Lambek Grammar: an Information-based Categorial Grammar
LP-92-02 Patrick Blackburn Modal Logic and Attribute Value Structures
LP-92-03 Szabolcs Mikulás The Completeness of the Lambek Calculus with respect to Relational Semantics
LP-92-04 Paul Dekker An Update Semantics for Dynamic Predicate Logic
LP-92-05 David I. Beaver The Kinematics of Presupposition
LP-92-06 Patrick Blackburn, Edith Spaan A Modal Perspective on the Computational Complexity of Attribute Value Grammar
LP-92-07 Jeroen Groenendijk, Martin Stokhof A Note on Interrogatives and Adverbs of Quantification
LP-92-08 Maarten de Rijke A System of Dynamic Modal Logic
LP-92-09 Johan van Benthem Quantifiers in the world of Types
LP-92-10 Maarten de Rijke Meeting Some Neighbours (a dynamic modal logic meets theories of change and knowledge representation)
LP-92-11 Johan van Benthem A note on Dynamic Arrow Logic
LP-92-12 Heinrich Wansing Sequent Caluli for Normal Modal Propositional Logics
LP-92-13 Dag Westerståhl Iterated Quantifiers
LP-92-14 Jeroen Groenendijk, Martin Stokhof Interrogatives and Adverbs of Quantification

Mathematical Logic and Foundations

ML-92-01 A.S. Troelstra Comparing the theory of Representations and Constructive Mathematics
ML-92-02 Dmitrij P. Skvortsov, Valentin B. Shehtman Maximal Kripke-type Semantics for Modal and Superintuitionistic Predicate Logics
ML-92-03 Zoran Marković On the Structure of Kripke Models of Heyting Arithmetic
ML-92-04 Dimiter Vakarelov A Modal Theory of Arrows, Arrow Logics I
ML-92-05 Domenico Zambella Shavrukov's Theorem on the Subalgebras of Diagonalizable Algebras for Theories containing $I\Delta_0 + EXP$
ML-92-06 D.M. Gabbay, Valentin B. Shehtman Undecidability of Modal and Intermediate First-Order Logics with Two Individual Variables
ML-92-07 Harold Schellinx How to Broaden your Horizon
ML-92-08 Raymond Hoofman Information Systems as Coalgebras
ML-92-09 A.S. Troelstra Realizability
ML-92-10 V.Yu. Shavrukov A Smart Child of Peano's

Compution and Complexity Theory

CT-92-01 Erik de Haas, Peter van Emde Boas Object Oriented Application Flow Graphs and their Semantics
CT-92-02 Karen L. Kwast, Sieger van Denneheuvel Weak Equivalence: Theory and Applications
CT-92-03 Krzysztof R. Apt, Kees Doets A new Definition of SLDNF-resolution

Other Prepublications

X-92-01 Heinrich Wansing The Logic of Information Structures
X-92-02 Konstantin N. Ignatiev The Closed Fragment of Dzhaparidze's Polymodal Logic and the Logic of Σ_1 conservativity
X-92-03 Willem Groeneveld Dynamic Semantics and Circular Propositions, revised version
X-92-04 Johan van Benthem Modeling the Kinematics of Meaning
X-92-05 Erik de Haas, Peter van Emde Boas Object Oriented Application Flow Graphs and their Semantics, revised version

1993 *Logic, Semantics and Philosophy of Langauge*

LP-93-01 Martijn Spaan Parallel Quantification
LP-93-02 Makoto Kanazawa Dynamic Generalized Quantifiers and Monotonicity
LP-93-03 Nikolai Pankrat'ev Completeness of the Lambek Calculus with respect to Relativized Relational Semantics
LP-93-04 Jacques van Leeuwen Identity, Quarrelling with an Unproblematic Notion
LP-93-05 Jaap van der Does Sums and Quantifiers
LP-93-06 Paul Dekker Updates in Dynamic Semantics
LP-93-07 Wojciech Buszkowski On the Equivalence of Lambek Categorial Grammars and Basic Categorial Grammars
LP-93-08 Zisheng Huang, Peter van Emde Boas Information Acquisition from Multi-Agent resources; abstract

Mathematical Logic and Foundations

ML-93-01 Maciej Kandulski Commutative Lambek Categorial Grammars
ML-93-02 Johan van Benthem, Natasha Alechina Modal Quantification over Structured Domains
ML-93-03 Mati Pentus The Conjoinablity Relation in Lambek Calculus and Linear Logic
ML-93-04 Andreja Prijatelj Bounded Contraction and Many-Valued Semantics
ML-93-05 Raymond Hoofman, Harold Schellinx Models of the Untyped λ-calculus in Semi Cartesian Closed Categories
ML-93-06 J. Zashev Categorial Generalization of Algebraic Recursion Theory
ML-93-07 A.V. Chagrov, L.A. Chagrova Algorithmic Problems Concerning First-Order Definability of Modal Formulas on the Class of All Finite Frames
ML-93-08 Raymond Hoofman, Ieke Moerdijk Remarks on the Theory of Semi-Functors
ML-93-09 A.S. Troelstra Natural Deduction for Intuitionistic Linear Logic
ML-93-10 Vincent Danos, Jean-Baptiste Joinet, Harold Schellinx The Structure of Exponentials: Uncovering the Dynamics of Linear Logic Proofs
ML-93-11 Lex Hendriks Inventory of Fragments and Exact Models in Intuitionistic Propositional Logic
ML-93-12 V.Yu. Shavrukov Remarks on Uniformly Finitely Precomplete Positive Equivalences

Compution and Complexity Theory

CT-93-01 Marianne Kalsbeek The Vanilla Meta-Interpreter for Definite Logic Programs and Ambivalent Syntax
CT-93-02 Sophie Fischer A Note on the Complexity of Local Search Problems
CT-93-03 Johan van Benthem, Jan Bergstra Logic of Transition Systems
CT-93-04 Karen L. Kwast, Sieger van Denneheuvel The Meaning of Duplicates in the Relational Database Model
CT-93-05 Erik Aarts Proving Theorems of the Lambek Calculus of Order 2 in Polynomial Time
CT-93-06 Krzysztof R. Apt Declarative programming in Prolog

Computaional Linguistics

CL-93-01 Noor van Leusen, László Kálmán The Interpretation of Free Focus
CL-93-02 Theo M.V. Janssen An Algebraic View On Rosetta

Other Prepublications

X-93-01 Paul Dekker Existential Disclosure, revised version
X-93-02 Maarten de Rijke What is Modal Logic?
X-93-03 Michiel Leezenberg Gorani Influence on Central Kurdish: Substratum or Prestige Borrowing
X-93-04 A.S. Troelstra (editor) Metamathematical Investigation of Intuitionistic Arithmetic and Analysis, Corrections to the First Edition
X-93-05 A.S. Troelstra (editor) Metamathematical Investigation of Intuitionistic Arithmetic and Analysis, Second, corrected Edition
X-93-06 Michael Zakharyashev Canonical Formulas for K4. Part II: Cofinal Subframe Logics