Separating Syntax and Combinatorics in Categorial Grammar

Reinhard Muskens

Tilburg University

Lambda Calculus and Formal Grammar
NII, Tokyo, 18 February 2005

Tectogrammar and Phenogrammar

- Curry (1961) distinguishes two levels of grammar, which he calls tectogrammatics and phenogrammatics.
- He considers a type hierarchy of strings, functions from strings to strings, functions from functions from strings to strings to strings, etc.
- The tectogrammatics of an expression is the way it was built with the help of such functions ("grammatical structure in itself"), the phenogrammatics is the result of evaluating these functions.

Purpose of this Talk

- Curry's work was a form of categorial grammar and in this talk I will argue that today's categorial grammar can still benefit from a separation between tectogrammar ('combinatorics') and phenogrammar ('syntax').
- I will argue that multimodal type logical grammar had better be split into a combinatorial and a multimodal part.
- The logic of the combinatorial part will be that of the $\multimap$ fragment of linear logic; that of the syntactic part will be a pure multimodal logic (not an amalgam of a multimodal logic and a resource concious one).
- The advantages are both linguistic (a simpler treatment of medial gaps) and formal (a modularized architecture).

Lambda Grammars and Abstract Categorial Grammars

- The treatment will essentially be based on the Abstract Categorial Grammars (ACGs) of de Groote (2001, 2002) and the Lambda Grammars of Muskens (2001, 2003).
- These two formalisms were developed simultaneously but independently and are closely akin: modulo a relatively minor liberalization of the definition of ACGs, a Lambda Grammar can be viewed as a certain collection of the latter.
- Marcus Kracht's recent book contains a treatment of what he calls de Saussure Grammars, which are also somewhat close.

Abstract Terms

- Start with a small collection of basic abstract types such as S, NP, N, $\mathrm{INF}, \ldots$
- From these build abstract types such as $\mathrm{NP}(\mathrm{NP}\ \mathrm{S})$ (officially $\mathrm{NP} \multimap (\mathrm{NP} \multimap \mathrm{S})$).
- We will have a vocabulary of abstract constants, typed with abstract types and infinitely many abstract variables in each abstract type.
- Using λ-abstraction and application build linear terms from these: each abstractor λX must bind exactly one variable X. The resulting terms are abstract terms.
- That's all. This will be our tectogrammar.

Examples

- `every` : $\mathrm{N}((\mathrm{NP}\ \ \mathrm{S})\mathrm{S})$
  `a` : $\mathrm{N}((\mathrm{NP}\ \ \mathrm{S})\mathrm{S})$
  `man` : N
  `woman` : N
  `loves` : $\mathrm{NP}(\mathrm{NP}\ \ \mathrm{S})$
  ξ, ξ' : NP.
- (`a woman`) $\lambda\xi$.(`every man`)(`loves` ξ) : S
- (`every man`) $\lambda\xi'$.(`a woman`) $\lambda\xi$.`loves` $\xi\xi'$: S

Dimensions of the Grammar

- The grammar will be multidimensional; among the dimensions are at least syntax (phenogrammar) and semantics.
- In each dimension there will be concrete types and concrete terms, built up in the usual way. The underlying logic in each case is classical type theory.
- Abstract terms and abstract types will have concretizations in each dimension.
- In each dimension a type homomorphism sends abstract types to concrete types and a term homomorphism sends abstract terms to concrete terms.

Type Homomorphisms

- For each dimension d, a type homomorphism c^d sends abstract types to concrete types. $c^d(AB) = c^d(A)c^d(B)$
- For basic abstract types, the values of the c^d can be chosen on a per grammar basis.
-

abstract type	syntax $(d = 1)$	semantics $(d = 2)$
S	νt	st
NP	νt	e
N	νt	$e(st)$
INF	νt	$e(st)$

- $c^1(\mathrm{N}((\mathrm{NP}\ \mathrm{S})\mathrm{S}) = (\nu t)(((\nu t)(\nu t))(\nu t))$
  $c^2(\mathrm{N}((\mathrm{NP}\ \mathrm{S})\mathrm{S}) = (e(st))((e(st))(st))$

Term Homomorphisms

- For each dimension d, a term homomorphism C^d sends abstract terms of type A to concrete terms of type $c^d(A)$. A term homomorphism is what you think it is.
- Choosing values for C^d for elements of the abstract vocabulary can again be done on a per grammar basis.
- $C^1(\mathtt{loves}) = \lambda t_1 \lambda t_2.(t_2 \bullet (\text{loves} \bullet t_1))$
  $C^2(\mathtt{loves}) = \lambda x \lambda y \lambda i.\mathit{love}(y, x, i)$
- A choice of $C^d(A)$ for each element A of the abstract vocabulary determines $C^d(A)$ for all terms A.
- The interpretation will be that, for each term A, $C^1(A)$ expresses $C^2(A)$.
- For each A, $\langle C^1(A), C^2(A) \rangle$ will be called a generated sign.

A Toy Lexicon

- `every`
  $\lambda t \lambda T.T(\text{every} \bullet t)$
  $\lambda P' P \lambda i \forall x[P'(x)(i) \rightarrow P(x)(i)]$
- `a`
  $\lambda t \lambda T.T(\text{a} \bullet t)$
  $\lambda P' P \lambda i \exists x[P'(x)(i) \wedge P(x)(i)]$
- `man`, man, *man*
- `woman`, woman, *woman*
- `loves`
  $\lambda t_1 \lambda t_2.(t_2 \bullet (\text{loves} \bullet t_1))$
  $\lambda x \lambda y \lambda i.love(y, x, i)$

Two Generated Signs

- $(\texttt{a woman})\,\lambda\xi.(\texttt{every man})(\texttt{loves}\ \ \xi)$

  $((\text{every} \bullet \text{man}) \bullet (\text{loves} \bullet (\text{a} \bullet \text{woman})))$
  $\lambda i \exists y[\mathit{woman}(y, i) \wedge \forall x[\mathit{man}(x, i) \rightarrow \mathit{love}(x, y, i)]]$

- $(\texttt{every man})\,\lambda\xi'.(\texttt{a woman})\,\lambda\xi.\texttt{loves}\ \ \xi\xi'$

  $((\text{every} \bullet \text{man}) \bullet (\text{loves} \bullet (\text{a} \bullet \text{woman})))$
  $\lambda i \forall x[\mathit{man}(x, i) \rightarrow \exists y[\mathit{woman}(y, i) \wedge \mathit{love}(x, y, i)]]$

Phenogrammar Deals with Word Order

- The combinatorial part of our grammar is undirected. Word order is dealt with on the level of the syntactic terms.
- This idea can already be found in Curry (1961). Oehrle (1994, 1995) explicitly considers λ-terms over a syntactic domain. These are attached to proofs via the Curry-Howard isomorphism.
- If tectogrammar is language universal, as argued by Dowty (1982), word order should definitely go to phenogrammar.

Medial Gaps

- One obvious linguistic advantage of the approach is a straightforward treatment of medial gaps as in *the book that Sue gave to Bill*.
- Directed forms of categorial grammar can deal with *peripheral* gaps and in multimodal approaches, where movement can be simulated, gaps can be sent to the periphery to be dealt with there (Morrill, Moortgat).
- But movement of gaps to the periphery seems an artifice that arises because of the formal machinery that was chosen. It does not seem a solution to an inherently linguistic problem.

Medial Quantifiers

- In directed systems there are related problems with obtaining all readings for sentences with quantifiers in medial positions. Here is a Dutch example due to Gosse Bouma:
- (dat) elke man een vrouw kust
  '(that) every man kisses a woman'
- The $\exists\forall$ reading is not obtained. Here we see that treating word order on the level of types creates problems in a domain that has nothing to do with word order at all.
- Again there are clever solutions, but again one feels that the need for these solutions arises out of the choice of formal machinery, not out of some real property of natural language.

Modularity: Linguistic Perspectives

- Syntax and semantics treated in strictly parallel fashion. Textbook wisdom that semantics is dependent upon syntax is contradicted.
- Surface compositionality is an illusion. It can only be upheld as long as phenogrammar is sufficiently like tectogrammar.
- For many of the world's languages (e.g. Warlpiri) there is no internal evidence for a syntactic structure rich enough to support semantics (Simpson, Dalrymple et al.).
- Type flexibility needed in semantics no longer constrained to raising from peripheral positions.

Modularity: Logical Perspectives

- An n-dimensional λ-grammar/ACG consists of $n+1$ logical engines.
  - The basic combinatorial mechanism (linear λ-terms = the $\multimap$ fragment of linear logic = the combinators **B**, **C** and **I**).
  - A logic for each of the n dimensions. *Axioms* may play a role here.
- The various logics can be studied in isolation. Communication between various dimensions goes through the lexicon.
- Teasing apart logics in this way will hopefully lead to increased simplicity.
- We'll sketch possibilities for setting up the phrase structure component in the next section.

Syntax as a Modal Logic

- Lambda Grammars/ACGs are compatible with many approaches to setting up logics for the various dimensions.
- Our strategy: take classical type theory for each dimension and add axioms (postulates) for extra requirements on structure.
- We'll sketch how multimodal approaches to movement and restructuring (Oehrle, Morrill, Moortgat) can be modeled in this way.
- There will no special attempt to be linguistically innovative. Our point concerns architecture, not linguistic analysis.

Node Properties

- Remember our choice $c^1(\text{S}) = c^1(\text{NP}) = c^1(\text{N}) = c^1(\text{INF}) = \nu t$. We will consider *properties of nodes* rather than nodes.
- This will give a boolean structure and we will employ $\sqsubseteq$, $\sqcap$, etc., with the obvious definitions.
- Suppose AX is a set of axioms. Then we can say, for Σ, Σ' of type νt, that Σ' follows from Σ if $\text{AX} \models \Sigma \sqsubseteq \Sigma'$.
- I.e. entailment is inclusion in all relevant models.
- Choosing νt as our central syntactic domain also provides us with a possibility to define modal operators.

Modal Operators

- We will assume that our language contains symbols for unary (type νt), binary (type $\nu(\nu t)$), and ternary (type $\nu(\nu(\nu t))$) accessibility relations.
- With the help of a binary accessibility relation R^m, we can define unary modal operators $\Diamond_m$ and $\Box_m$ (type $(\nu t)(\nu t)$).
- $\Diamond_m = \lambda t \lambda k.\exists k'[R^m(k,k') \wedge t(k')]$
  $\Box_m = \lambda t \lambda k.\exists k'[R^m(k',k) \rightarrow t(k')]$
- A ternary accessibility relation R^m will give rise to a binary modality $\bullet_m$ (type $(\nu t)((\nu t)(\nu t))$), for which we will use infix notation.
- $\bullet_m = \lambda t_1 t_2 \lambda k.\exists k_1 k_2[R^m(k,k_1,k_2) \wedge t_1(k_1) \wedge t_2(k_2)]$
- Viewed in this light type νt symbols can be interpreted as 0-place modal operators.

Interaction Postulates

- Interaction postulates such as $\forall k_1 k_2 k_3 k_4 [\exists k [R^c(k_1, k_2, k) \wedge R^\uparrow(k, k_3, k_4)] \rightarrow [\exists k [R^\uparrow(k_1, k, k_4) \wedge R^c(k, k_2, k_3)]]$
- will entail interaction principles such as $A \bullet_c (B \bullet_\uparrow C) \sqsubseteq (A \bullet_c B) \bullet_\uparrow C$
- Such principles allow restructuring of (multimodal) trees.

Monotonicity

The following principles hold:

$A \sqsubseteq A'$ entails $\Diamond_m A \sqsubseteq \Diamond_m A'$
$A \sqsubseteq A'$ entails $\Box_m A \sqsubseteq \Box_m A'$
$A \sqsubseteq A'$ entails $A \bullet_m B \sqsubseteq A' \bullet_m B$
$B \sqsubseteq B'$ entails $A \bullet_m B \sqsubseteq A \bullet_m B'$
$A \sqsubseteq A'$ entails $A \sqcap B \sqsubseteq A' \sqcap B$
$B \sqsubseteq B'$ entails $A \sqcap B \sqsubseteq A \sqcap B'$

This means we can rewrite in any of our modal contexts.

Strings

- Consider a modality $\circ$ with underlying accessibility relation R°, with $R^\circ kk_1k_2$ interpreted as 'k is the concatenation of k_1 and k_2'.
- Some type ν objects are interpreted as strings (with 1 as unity), others may be various kinds of trees, etc.
- Give some obvious axioms for R°, validating:

(1) a. $(A \circ B) \circ C = A \circ (B \circ C)$
b. $A \circ \mathrm{e} \sqsubseteq A$
c. $\mathrm{e} \circ A \sqsubseteq A$

- We ultimately want to connect strings with meanings.
- But may want to use intermediary structures, such as (multimodal) trees.

From Trees to Strings

- The modality $\bullet_c$, usually written as $\bullet$, will represent constituency. Read $\Diamond_y$ as 'the yield of'.
- The Tree-to-String package:

$$
\begin{array}{ll}
\Diamond_y(A \bullet B) \sqsubseteq \Diamond_y A \circ \Diamond_y B & \text{TS1} \\
\Diamond_y A \sqsubseteq A, \text{ if } A \in \text{Lex or } A = \text{e} & \text{TS2} \\
\Diamond_y \Diamond_y A \sqsubseteq \Diamond_y A & \text{TS3}
\end{array}
$$

-
$$
\begin{array}{ll}
\Diamond_y(\text{Aad} \bullet (\text{denkt} \bullet \Diamond_y(\text{dat} \bullet (\text{Marie} \bullet \text{slaapt})))) & \\
\Diamond_y\text{Aad} \circ \Diamond_y(\text{denkt} \bullet \Diamond_y(\text{dat} \bullet (\text{Marie} \bullet \text{slaapt}))) & \text{TS1} \\
\text{Aad} \circ \Diamond_y(\text{denkt} \bullet \Diamond_y(\text{dat} \bullet (\text{Marie} \bullet \text{slaapt}))) & \text{TS2} \\
\text{Aad} \circ \Diamond_y\text{denkt} \circ \Diamond_y\Diamond_y(\text{dat} \bullet (\text{Marie} \bullet \text{slaapt})) & \text{TS1} \\
\text{Aad} \circ \text{denkt} \circ \Diamond_y\Diamond_y(\text{dat} \bullet (\text{Marie} \bullet \text{slaapt})) & \text{TS2} \\
\text{Aad} \circ \text{denkt} \circ \Diamond_y(\text{dat} \bullet (\text{Marie} \bullet \text{slaapt})) & \text{TS3} \\
\text{Aad} \circ \text{denkt} \circ \text{dat} \circ \text{Marie} \circ \text{slaapt} & \text{etc.}
\end{array}
$$

Derivable String-Meaning Signs

- A νt term is a $\circ$-term if it is built from word labels (every, likes,...) using $\circ$ only.
- A sign $\langle \Sigma, \Delta \rangle$ (with Σ in the phrase structure dimension) is a string-meaning sign if Σ is a $\circ$-term.
- A sign $\langle \Sigma, \Delta \rangle$ is a derivable sign if there is a generated sign $\langle \Sigma', \Delta \rangle$ such that $\Sigma' \sqsubseteq \Sigma$ is valid, given our postulates. We are interested in the derivable string-meaning signs.
- In order to show that we can derive such a sign, we must get rid of all modalities other than $\circ$.
- This can be used to enforce movement, feature checking, etc. We have a way to drive derivations.

Dutch Word Order

- We'll sketch an analysis of some aspects of Dutch word order along the lines of Oehrle and Moortgat.
- Verb final order is default, V2 and V1 need something special.
- We'll assume that mood places an outer modality. Yes/no-questions get $\Diamond_1$; assertions $\Diamond_2$.

The Tectogrammar Part

- $?((\texttt{een}\,\texttt{docent})\lambda\zeta.(\texttt{een}\,\texttt{student})(\texttt{mag}(\texttt{plagen}\,\zeta)))$
- $\Diamond_y\Diamond_1((\text{een} \bullet \text{student}) \bullet (((\text{een} \bullet \text{docent}) \bullet \text{plagen}_{\text{vc}}) \bullet_0 \text{mag}_{\text{vc,fin}}))$
  $\lambda i.[\exists y[\mathit{teacher}\ yi \wedge \exists z[\mathit{student}\ zi \wedge \exists j[Mji \wedge \mathit{tease}\ zyj]]] \leftrightarrow \exists y[\mathit{teacher}\ yw_0 \wedge \exists z[\mathit{student}\ zw_0 \wedge \exists j[Mjw_0 \wedge \mathit{tease}\ zyj]]]]$
- May a student tease a teacher?
- Lexicon suppressed here.
- $\text{mag}_{\text{vc,fin}}$, for example, is short for $\text{mag} \sqcap \text{vc} \sqcap \text{fin}$
- The syntactic part must be rewritten:
  $\text{mag} \circ \text{een} \circ \text{student} \circ \text{een} \circ \text{docent} \circ \text{plagen}$

Verb Clusters 1

- Following Oehrle (to appear) and Moortgat (1999), we introduce a modality $\bullet_0$ for head adjunction.
- The Verb Clustering package:

$$
\begin{array}{ll}
(A \bullet B) \bullet_0 C \sqsubseteq A \bullet (B \bullet_0 C) & \mathrm{VC1} \\
(A \bullet B) \bullet_0 C \sqsubseteq (A \bullet_0 C) \bullet B & \mathrm{VC2} \\
A_{\mathrm{vc}} \bullet_0 B_{\mathrm{vc}} \sqsubseteq (B \bullet A)_{\mathrm{vc}} & \mathrm{VC3}
\end{array}
$$

- `((wil((helpen(kussen marie))ben))aad)`
  $(\mathrm{Aad} \bullet ((\mathrm{Ben} \bullet ((\mathrm{Marie} \bullet \mathrm{kussen}_{\mathrm{vc}}) \bullet_0 \mathrm{helpen}_{\mathrm{vc}})) \bullet_0 \mathrm{wil}_{\mathrm{vc,fin}}))$

Verb Clusters 2

$(\text{Aad} \bullet ((\text{Ben} \bullet ((\text{Marie} \bullet \text{kussen}_{\text{vc}}) \bullet_0 \text{helpen}_{\text{vc}})) \bullet_0 \text{wil}_{\text{vc,fin}}))$	
$(\text{Aad} \bullet ((\text{Ben} \bullet (\text{Marie} \bullet (\text{kussen}_{\text{vc}} \bullet_0 \text{helpen}_{\text{vc}}))) \bullet_0 \text{wil}_{\text{vc,fin}}))$	VC1
$(\text{Aad} \bullet (\text{Ben} \bullet ((\text{Marie} \bullet (\text{kussen}_{\text{vc}} \bullet_0 \text{helpen}_{\text{vc}})) \bullet_0 \text{wil}_{\text{vc,fin}})))$	VC1
$(\text{Aad} \bullet (\text{Ben} \bullet (\text{Marie} \bullet ((\text{kussen}_{\text{vc}} \bullet_0 \text{helpen}_{\text{vc}}) \bullet_0 \text{wil}_{\text{vc,fin}}))))$	VC1
$(\text{Aad} \bullet (\text{Ben} \bullet (\text{Marie} \bullet ((\text{helpen} \bullet \text{kussen})_{\text{vc}} \bullet_0 \text{wil}_{\text{vc,fin}}))))$	VC3
$(\text{Aad} \bullet (\text{Ben} \bullet (\text{Marie} \bullet (\text{wil}_{\text{fin}} \bullet (\text{helpen} \bullet \text{kussen}))_{\text{vc}})))$	VC3
$(\text{Aad} \bullet (\text{Ben} \bullet (\text{Marie} \bullet (\text{wil}_{\text{fin}} \bullet (\text{helpen} \bullet \text{kussen})))))$	Boole

- Dependencies are cross-serial.
- (that) Aad wants to help Ben kiss Marie

Verb Initial and Verb Second 1

- The Rising package:

$$A_{\text{fin}} \sqsubseteq \text{e} \bullet_{\uparrow} A_{\text{fin}} \quad \uparrow 1$$
$$A \bullet (B \bullet_{\uparrow} C) \sqsubseteq (A \bullet B) \bullet_{\uparrow} C \quad \uparrow 2$$
$$(A \bullet_{\uparrow} B) \bullet C \sqsubseteq (A \bullet C) \bullet_{\uparrow} B \quad \uparrow 3$$

- The V package:

$$\Diamond_1(A \bullet_{\uparrow} B_{\text{fin}}) \sqsubseteq B \bullet A \quad \text{V1}$$
$$\Diamond_2((A \bullet B) \bullet_{\uparrow} C_{\text{fin}}) \sqsubseteq A \bullet (C \bullet B) \quad \text{V2}$$

- Finite verbs can go into ‘rise mode’, travel upwards until they meet $\Diamond_1$ or $\Diamond_2$ and are then placed into first ($\Diamond_1$) or second ($\Diamond_2$) position.

Verb Initial and Verb Second 2

- `(assert((wil((helpen(kussen marie))ben))aad))`
  $\Diamond_y \Diamond_2(\mathsf{Aad} \bullet ((\mathsf{Ben} \bullet ((\mathsf{Marie} \bullet \mathsf{kussen}_{\mathsf{vc}}) \bullet_0 \mathsf{helpen}_{\mathsf{vc}})) \bullet_0 \mathsf{wil}_{\mathsf{vc,fin}}))$

$\Diamond_y \Diamond_2(\mathsf{Aad} \bullet ((\mathsf{Ben} \bullet ((\mathsf{Marie} \bullet \mathsf{kussen}_{\mathsf{vc}}) \bullet_0 \mathsf{helpen}_{\mathsf{vc}})) \bullet_0 \mathsf{wil}_{\mathsf{vc,fin}}))$	
$\Diamond_y \Diamond_2(\mathsf{Aad} \bullet (\mathsf{Ben} \bullet (\mathsf{Marie} \bullet (\mathsf{wil}_{\mathsf{fin}} \bullet (\mathsf{helpen} \bullet \mathsf{kussen})))))$	as before
$\Diamond_y \Diamond_2(\mathsf{Aad} \bullet (\mathsf{Ben} \bullet (\mathsf{Marie} \bullet ((e \bullet_\uparrow \mathsf{wil}_{\mathsf{fin}}) \bullet (\mathsf{helpen} \bullet \mathsf{kussen})))))$	$\uparrow 1$
$\Diamond_y \Diamond_2(\mathsf{Aad} \bullet (\mathsf{Ben} \bullet (\mathsf{Marie} \bullet ((e \bullet (\mathsf{helpen} \bullet \mathsf{kussen})) \bullet_\uparrow \mathsf{wil}_{\mathsf{fin}}))))$	$\uparrow 3$
$\Diamond_y \Diamond_2(\mathsf{Aad} \bullet (\mathsf{Ben} \bullet ((\mathsf{Marie} \bullet (e \bullet (\mathsf{helpen} \bullet \mathsf{kussen}))) \bullet_\uparrow \mathsf{wil}_{\mathsf{fin}})))$	$\uparrow 2$
$\Diamond_y \Diamond_2(\mathsf{Aad} \bullet ((\mathsf{Ben} \bullet (\mathsf{Marie} \bullet (e \bullet (\mathsf{helpen} \bullet \mathsf{kussen})))) \bullet_\uparrow \mathsf{wil}_{\mathsf{fin}}))$	$\uparrow 2$
$\Diamond_y \Diamond_2((\mathsf{Aad} \bullet (\mathsf{Ben} \bullet (\mathsf{Marie} \bullet (e \bullet (\mathsf{helpen} \bullet \mathsf{kussen}))))) \bullet_\uparrow \mathsf{wil}_{\mathsf{fin}})$	$\uparrow 2$
$\Diamond_y (\mathsf{Aad} \bullet (\mathsf{wil} \bullet (\mathsf{Ben} \bullet (\mathsf{Marie} \bullet (e \bullet (\mathsf{helpen} \bullet \mathsf{kussen}))))))$	V2
$\vdots$	
$\mathsf{Aad} \circ \mathsf{wil} \circ \mathsf{Ben} \circ \mathsf{Marie} \circ \mathsf{helpen} \circ \mathsf{kussen}$	TS

Conclusion

- Lambda Grammars/ACGs offer a highly modular approach to formal linguistics.
- Syntax and semantics are derived from the same underlying tectogrammar, but are relatively autonomous.
- One can use a multimodal logic as the logic of the phenogrammar dimension and let derivations be driven by the need to get string-meaning signs.
- The usual set-up of Multimodal Categorial Grammar is then modularized and simplified.
- But it might be a good idea to reign in the generative capacity of the multimodal component.

Conclusion (continued)

- In a multimodal approach, derivations are akin to derivations in generative grammar. The resemblance can be made stronger by choosing appropriate postulates.
- But the single mother-daughter relation one finds in generative grammar is replaced by a whole gamut of modes of composition.

Paper

Reinhard Muskens.
Separating Syntax and Combinatorics in Categorial Grammar.
Manuscript.
Available from the author (r.a.muskens@uvt.nl).