Structures Informatiques et Logiques pour la Modélisation Linguistique (MPRI 2.27.1 - second part)

Philippe de Groote

Inria

2015-2016

Syntax/semantics Interface

1. Compositionality
2. Context-free grammars
   - Example
   - Abstract syntax as heterogeneous algebra
   - Homomorphism
3. Higher-order abstract syntax
   - Higher-order signature
   - Examples
   - Higher-order homomorphism
4. Abstract categorial grammars
   - Definition
   - Generated languages
   - Example
   - Language-theoretic example
   - Expressive power

Syntax/semantics Interface

Compositionality

Compositionality principle

- The meaning of a complex expression is determined by the meanings of its constituents and by the formation rules used to combine them.

Montague's homomorphism requirement

- Semantics must be obtained as a homomorphic image of syntax.

Contextuality principle

- The meaning of an expression is determined by the meanings of the complex expressions of which it is a constituent.

Syntax/semantics Interface

Syntax/semantics Interface

Rule to rule semantics

Context free grammar:

$$\begin{aligned}
\mathrm{S} &\to \mathrm{NP}\ \mathrm{VP}\\
\mathrm{VP} &\to \mathrm{tV}\ \mathrm{NP}\\
\mathrm{tV} &\to \text{loves}\\
\mathrm{NP} &\to \text{John}\\
\mathrm{NP} &\to \text{somebody}
\end{aligned}$$

Semantic rules:

$$\begin{aligned}
[\![\mathrm{S}]\!] &= [\![\mathrm{NP}]\!]\,[\![\mathrm{VP}]\!]\\
[\![\mathrm{VP}]\!] &= \lambda x.\,[\![\mathrm{NP}]\!]\,(\lambda y.\,[\![\mathrm{tV}]\!]\,y\,x)\\
[\![\mathrm{tV}]\!] &= \lambda y.\,\lambda x.\,\mathbf{love}\,x\,y\\
[\![\mathrm{NP}]\!] &= \lambda k.\,k\,\mathbf{j}\\
[\![\mathrm{NP}]\!] &= \lambda k.\,\exists y.\,k\,y
\end{aligned}$$

Syntax/semantics Interface

Signature associated to a CFG

Context free grammar:

$$\begin{array}{ll} \mathrm{S} \to \mathrm{NP\ VP} & (\mathrm{p}_1) \\ \mathrm{VP} \to \mathrm{tV\ NP} & (\mathrm{p}_2) \\ \mathrm{tV} \to \mathrm{loves} & (\mathrm{p}_3) \\ \mathrm{NP} \to \mathrm{John} & (\mathrm{p}_4) \\ \mathrm{NP} \to \mathrm{somebody} & (\mathrm{p}_5) \end{array}$$

Associate a *sort* to each non-terminal, and an *operator* to each production rule:

$$\begin{array}{l} \mathrm{p}_1 : \mathrm{NP} \times \mathrm{VP} \to \mathrm{S} \\ \mathrm{p}_2 : \mathrm{tV} \times \mathrm{NP} \to \mathrm{VP} \\ \mathrm{p}_3 : \mathrm{tV} \\ \mathrm{p}_4 : \mathrm{NP} \\ \mathrm{p}_5 : \mathrm{NP} \end{array}$$

Syntax/semantics Interface

Syntactic and semantic algebras

Syntactic algebra:

$$\begin{aligned} \mathrm{p}_1 &: \mathrm{NP} \times \mathrm{VP} \to \mathrm{S} \\ \mathrm{p}_2 &: \mathrm{tV} \times \mathrm{NP} \to \mathrm{VP} \\ \mathrm{p}_3 &: \mathrm{tV} \\ \mathrm{p}_4 &: \mathrm{NP} \\ \mathrm{p}_5 &: \mathrm{NP} \end{aligned}$$

Semantic algebra:

$$\begin{aligned} \mathrm{f}_1(a,b) &= a\,b && : \mathrm{NP}^* \times \mathrm{VP}^* \to \mathrm{S}^* \\ \mathrm{f}_2(a,b) &= \lambda x.\, b\,(\lambda y.\, a\,y\,x) && : \mathrm{tV}^* \times \mathrm{NP}^* \to \mathrm{VP}^* \\ \mathrm{f}_3 &= \lambda y.\, \lambda x.\, \mathbf{love}\,x\,y && : \mathrm{tV}^* \\ \mathrm{f}_4 &= \lambda k.\, k\,\mathbf{j} && : \mathrm{NP}^* \\ \mathrm{f}_5 &= \lambda k.\, \exists y.\, k\,y && : \mathrm{NP}^* \end{aligned}$$

Where:

$$\begin{aligned} \mathrm{S}^* &= o \\ \mathrm{VP}^* &= \iota \to o \\ \mathrm{tV}^* &= \iota \to \iota \to o \\ \mathrm{NP}^* &= (\iota \to o) \to o \end{aligned}$$

Syntax/semantics Interface

Syntax/semantics Interface

Definition

Let $\mathcal{T}(A)$ be the set of functional types built on the set of atomic types A, i.e.:

$$\mathcal{T}(A) \quad ::= \quad A \mid (\mathcal{T}(A) \to \mathcal{T}(A))$$

A higher-order signature is a triple $\Sigma = \langle A, C, \tau \rangle$, where:

- A is a finite set of atomic types;
- C is a finite set of constants;
- $\tau : C \to \mathcal{T}(A)$ is a function that assigns each constant in C with a simple type built on A.

We use $\Lambda(\Sigma)$ to denote the set of simply typed λ-terms built upon a higher-order linear signature Σ.
We use $\Lambda^0(\Sigma)$ to denote the set of linear λ-terms built upon a higher-order signature Σ.

Syntax/semantics Interface

Trees

$$
\begin{aligned}
p_1 &: \mathrm{NP} \to \mathrm{VP} \to \mathrm{S} \\
p_2 &: \mathrm{tV} \to \mathrm{NP} \to \mathrm{VP} \\
p_3 &: \mathrm{tV} \\
p_4 &: \mathrm{NP} \\
p_5 &: \mathrm{NP}
\end{aligned}
$$

Strings

A canonical way of representing strings as λ-terms consists of representing them as function compositions:

$$\text{`}abbac\text{'} = \lambda x.\, a\,(b\,(b\,(a\,(c\,x))))$$

In this setting:

$$\begin{array}{rcl} \epsilon & \triangleq & \lambda x.\, x \\ \alpha + \beta & \triangleq & \lambda \alpha.\, \lambda \beta.\, \lambda x.\, \alpha\,(\beta\,x) \end{array}$$

First-order logic

$$
\begin{aligned}
\mathrm{zero} &: \mathrm{term} \\
\mathrm{succ} &: \mathrm{term} \to \mathrm{term} \\
\mathrm{add} &: \mathrm{term} \to \mathrm{term} \to \mathrm{term} \\
&\vdots \\
\mathrm{eq} &: \mathrm{term} \to \mathrm{term} \to \mathrm{prop} \\
\mathrm{not} &: \mathrm{prop} \to \mathrm{prop} \\
\mathrm{and} &: \mathrm{prop} \to \mathrm{prop} \to \mathrm{prop} \\
\mathrm{forall} &: (\mathrm{term} \to \mathrm{prop}) \to \mathrm{prop}
\end{aligned}
$$

linguistic example

$$
\begin{array}{rl}
 & \vdots \\
\mathrm{a} & : \mathrm{N} \to \mathrm{NP} \\
\mathrm{wise} & : \mathrm{N} \to \mathrm{N} \\
\mathrm{man} & : \mathrm{N} \\
\mathrm{who} & : (\mathrm{NP} \to \mathrm{S}) \to \mathrm{N} \to \mathrm{N} \\
\mathrm{loves} & : \mathrm{NP} \to \mathrm{NP} \to \mathrm{S} \\
\mathrm{himself} & : (\mathrm{NP} \to \mathrm{NP} \to \mathrm{S}) \to \mathrm{NP} \to \mathrm{S} \\
 & \vdots
\end{array}
$$

Syntax/semantics Interface

Definition

Given two higher-order signatures $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ and $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$, a higher-order homomorphism $\mathcal{H} = \langle \eta, \theta \rangle$ from Σ_1 to Σ_2 is generated by two functions:

- $\eta : A_1 \to \mathcal{T}(A_2)$,
- $\theta : C_1 \to \Lambda(\Sigma_2)$,

such that

$$\vdash_{\Sigma_2} \theta(c) : \hat{\eta}(\tau_1(c)).$$

where $\hat{\eta}$ is the homomorphic extension of η, i.e.:

- $\hat{\eta}(a) = \eta(a)$, for $a \in A_1$.
- $\hat{\eta}(\alpha \to \beta) = \hat{\eta}(\alpha) \to \hat{\eta}(\beta)$.

Syntax/semantics Interface

Syntax/semantics Interface

Vocabularies and Lexicon

A vocabulary is defined to be a higher-order signature.

Given two vocabularies Σ_1 and Σ_2, a lexicon $\mathcal{L}$ from Σ_1 to Σ_2 is defined to be a linear higher-order homomorphism $\mathcal{L} : \Sigma_1 \to \Sigma_2$.

Definition

An abstract categorial grammar is a quadruple

$$\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathcal{L}, s \rangle$$

where :

- $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ and $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$ are two higher-order linear signatures; Σ_1 is called the abstract vocabulary and Σ_2 is called the object vocabulary;
- $\mathcal{L} : \Sigma_1 \to \Sigma_2$ is a lexicon from the abstract vocabulary to the object vocabulary;
- $s \in \mathcal{T}(A_1)$ is a type of the abstract vocabulary; it is called the distinguished type of the grammar.

Syntax/semantics Interface

Abstract and object language

Let $\Lambda^0(\Sigma)$ be the set of linear λ-terms built upon a higher-order signature Σ. The abstract language generated by $\mathcal{G}$ ($\mathcal{A}(\mathcal{G})$) is defined as follows:

$$\mathcal{A}(\mathcal{G}) = \{t \in \Lambda^0(\Sigma_1) \mid \vdash_{\Sigma_1} t : s \text{ is derivable}\}$$

The object language generated by $\mathcal{G}$ ($\mathcal{O}(\mathcal{G})$) is defined to be the image of the abstract language by the term homomorphism induced by the lexicon $\mathcal{L}$:

$$\mathcal{O}(\mathcal{G}) = \{t \in \Lambda^0(\Sigma_2) \mid \exists u \in \mathcal{A}(\mathcal{G}).\ t = \mathcal{L}(u)\}$$

Syntax/semantics Interface

Signatures

$$
\begin{array}{lrcl}
\Sigma_0\text{:} & N, NP, S & : & \text{type} \\
& \mathrm{J} & : & NP \\
& \mathrm{U} & : & N \\
& \mathrm{A} & : & N \multimap ((NP \multimap S) \multimap S) \\
& \mathrm{S} & : & ((NP \multimap S) \multimap S) \multimap (NP \multimap S) \\
\Sigma_1\text{:} & /\mathsf{a}/,\ /\mathsf{John}/,\ /\mathsf{seeks}/,\ /\mathsf{unicorn}/ & : & STRING \\
\Sigma_2\text{:} & \iota, o & : & \text{type} \\
& \wedge & : & o \multimap (o \multimap o) \\
& \exists & : & (\iota \to o) \multimap o \\
& \mathbf{j} & : & \iota \\
& \mathbf{unicorn} & : & \iota \multimap o \\
& \mathbf{find} & : & \iota \multimap (\iota \multimap o) \\
& \mathbf{try} & : & \iota \multimap ((\iota \multimap o) \multimap o)
\end{array}
$$

Lexicons

$\mathcal{L}_1 : \Sigma_0 \to \Sigma_1$

$$
\begin{array}{rcl}
N, NP, S & := & STRING \\
\mathrm{J} & := & /\mathrm{John}/ \\
\mathrm{U} & := & /\mathrm{unicorn}/ \\
\mathrm{A} & := & \lambda x.\, \lambda p.\, p\,(/\mathrm{a}/ + x) \\
\mathrm{S} & := & \lambda p.\, \lambda x.\, p\,(\lambda y.\, x + /\mathrm{seeks}/ + y)
\end{array}
$$

$\mathcal{L}_2 : \Sigma_0 \to \Sigma_2$

$$
\begin{array}{rcl}
N & := & \iota \to o \\
NP & := & \iota \\
S & := & o \\
\mathrm{J} & := & \mathbf{j} \\
\mathrm{U} & := & \lambda x.\, \mathbf{unicorn}\, x \\
\mathrm{A} & := & \lambda p.\, \lambda q.\, \exists x.\, p\, x \wedge q\, x \\
\mathrm{S} & := & \lambda p.\, \lambda x.\, \mathbf{try}\, x\, (\lambda y.\, p\, (\lambda z.\, \mathbf{find}\, y\, z))
\end{array}
$$

Syntax/semantics transfer

We have that:

$$\mathcal{L}_1(\mathrm{S}\,(\mathrm{A}\,\mathrm{U})\,\mathrm{J}) = /\mathrm{John}/ + /\mathrm{seeks}/ + /\mathrm{a}/ + /\mathrm{unicorn}/$$

$$\mathcal{L}_2(\mathrm{S}\,(\mathrm{A}\,\mathrm{U})\,\mathrm{J}) = \mathbf{try}\,\mathbf{j}\,(\lambda x.\,\exists y.\,\mathbf{unicorn}\,y \wedge \mathbf{find}\,x\,y)$$

$$\mathcal{L}_1(\mathrm{A}\,\mathrm{U}\,(\lambda x.\,\mathrm{S}\,(\lambda k.\,k\,x)\,\mathrm{J})) = /\mathrm{John}/ + /\mathrm{seeks}/ + /\mathrm{a}/ + /\mathrm{unicorn}/$$

$$\mathcal{L}_2(\mathrm{A}\,\mathrm{U}\,(\lambda x.\,\mathrm{S}\,(\lambda k.\,k\,x)\,\mathrm{J})) = \exists y.\,\mathbf{unicorn}\,y \wedge \mathbf{try}\,\mathbf{j}\,(\lambda x.\,\mathbf{find}\,x\,y)$$

Syntax/semantics Interface

Context-free grammars

$$S \to \epsilon$$
$$S \to aSb$$

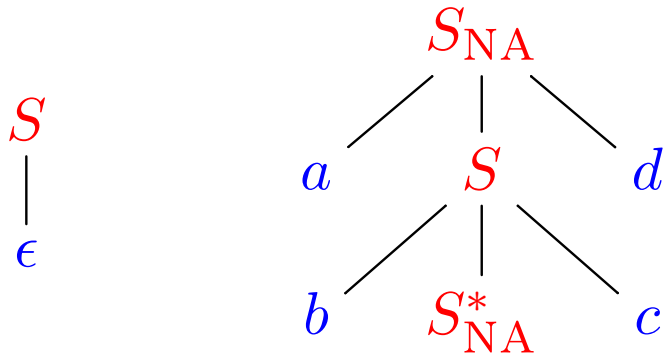
Abstract vocabulary :

$$S : \text{type}$$
$$A : S$$
$$B : S \multimap S$$

Lexicon :

$$S := string$$
$$A := \epsilon$$
$$B := \lambda x.\, a + x + b$$

Tree-adjoining grammars



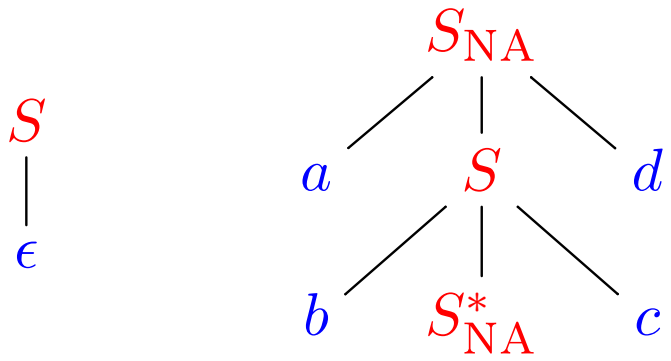
Abstract vocabulary :

$$
\begin{aligned}
&S, S', S'' : \text{type}\\
&A : (S'' \multimap S') \multimap S\\
&B : S'' \multimap (S'' \multimap S') \multimap S'\\
&C : S'' \multimap S'
\end{aligned}
$$

Lexicon :

$$
\begin{aligned}
&S, S', S'' := \mathit{string}\\
&A := \lambda f.\, f\, \epsilon\\
&B := \lambda x.\, \lambda g.\, a + g\,(b + x + c) + d\\
&C := \lambda x.\, x
\end{aligned}
$$

Tree-adjoining grammars revisited



Abstract vocabulary :

$$T, S : \text{type}$$
$$A : T \multimap S$$
$$B : T \multimap T$$
$$C : T$$

Lexicon :

$$S := string$$
$$T := string \multimap string$$
$$A := \lambda f.\, f\, \epsilon$$
$$B := \lambda g.\, \lambda x.\, a + g\,(b + x + c) + d$$
$$C := \lambda x.\, x$$

Syntax/semantics Interface

Abstract categorial hierarchy

Let $\mathcal{G} = \langle \Sigma, \Xi, \mathcal{L}, s \rangle$. Define the *order* and the *complexity* of $\mathcal{G}$:

- $\mathsf{ord}(\mathcal{G}) = \max\{\mathsf{ord}(\tau_\Sigma(c)) \,|\, c \in C_\Sigma\}$.
- $\mathsf{comp}(\mathcal{G}) = \max\{\mathsf{ord}(\mathcal{L}(a)) \,|\, a \in A_\Sigma\}$.

Define:

- $\mathbf{G}(m,n) = \{\mathcal{G} \,|\, \mathsf{ord}(\mathcal{G}) \leq m \text{ and } \mathsf{comp}(\mathcal{G}) \leq n\}$
- $\mathcal{L}(m,n) = \{\mathcal{O}(\mathcal{G}) \,|\, \mathcal{G} \in \mathbf{G}(m,n)\}$

For all $m, n \geq 1$, $\mathcal{L}(m, n+1) \subset \mathcal{L}(m+1, n)$.

For all $m, n \geq 1$, $\mathcal{L}(m+3, n) \subset \mathcal{L}(m+2, n+1)$.

Second-order hierarchy of string languages

$\mathcal{L}(2,1)$	regular languages
$\mathcal{L}(2,2)$	context-free languages
$\mathcal{L}(2,3)$	well-nested mildly context-sensitive languages
$\mathcal{L}(2,4)$	mildly context-sensitive languages
$\mathcal{L}(2,4+n)$	$\mathcal{L}(2,4)$

Membership

General case

- (Universal) membership is decidable if and only if the multiplicative-exponential fragment of linear logic is decidable.

Lexicalized case

- (Universal) membership is NP-complete.

Second-order case

- Universal membership is NP-complete, and membership is polynomial.