Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

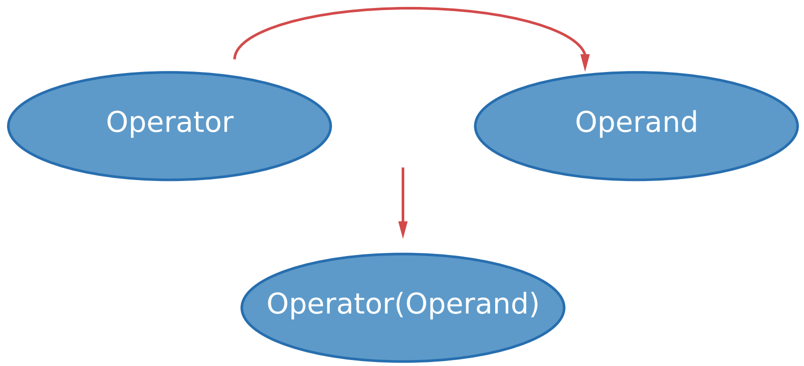
Autumn 2013

Outline

Applicative system

- CL (Curry & Feys, 1958, 1972) as an applicative system

  CL is an applicative system because the basic unique operation in CL is the application of an **operator** to an **operand**



Combinators

CL defines general operators, called Combinators.

- Each combinator composes between them the elementary combinators and defines the complexe combinators.
- Certains combinators are considered as the basic combinators to define the other combinators.

Elementary combinators

I	$=_{\text{def}}$	$\lambda x.x$	*(identificator)*
K	$=_{\text{def}}$	$\lambda x.\lambda y.x$	*(cancellator)*
W	$=_{\text{def}}$	$\lambda x.\lambda y.xyy$	*(duplicator)*
C	$=_{\text{def}}$	$\lambda x.\lambda y.\lambda z.xzy$	*(permutator)*
B	$=_{\text{def}}$	$\lambda x.\lambda y.\lambda z.x(yz)$	*(compositor)*
S	$=_{\text{def}}$	$\lambda x.\lambda y.\lambda z.xz(yz)$	*(substitution)*
Φ	$=_{\text{def}}$	$\lambda x.\lambda y.\lambda z.\lambda u.x(yu)(zu)$	*(distribution)*
Ψ	$=_{\text{def}}$	$\lambda x.\lambda y.\lambda z.\lambda u.x(yz)(yu)$	*(distribution)*

β-reductions

The combinators are associated with the **β-reductions** in a canonical form:

β-reduction relation between X and Y

$$X \geq_\beta Y$$

Y was obtained from X by a β-reduction

β-reductions

Ix	$\geq_\beta$	x
Kxy	$\geq_\beta$	x
Wxy	$\geq_\beta$	xyy
$Cxyz$	$\geq_\beta$	xzy
$Bxyz$	$\geq_\beta$	$x(yz)$
$Sxyz$	$\geq_\beta$	$xz(yz)$
$\Phi xyzu$	$\geq_\beta$	$x(yu)(zu)$
$\Psi xyzu$	$\geq_\beta$	$x(yz)(yu)$

Each combinator is an operator which has a certain number of arguments (operands); sequences of the arguments which follow the comnator are called “the scope of combinator”.

β-reductions

Intuitive interpretations of the elementary combinators are given by the associated **β-reductions**.

- The combinator I expresses the identity.
- The combinator K expresses the constant function.
- The combinator W expresses the diagonalisation or the duplication of an argument.
- The combinator C expresses the conversion, that is, the permutation of two arguments of an binary operator.
- The combinator B expresses the functional composition of two operators.
- The combinator S expresses the functional composition and the duplication of argument.
- The combinator Φ expresses the composition in parallel of operators acting on the common data.
- The combinator Ψ expresses the composition by distribution.

Introduction and elimination rules of combinators

Introduction and elimination rules of combinators can be presented in the style of Gentzen (*natural deduction*).

Elim. Rules	**Intro. Rules**
If --- [e-**I**] f	f --- [i-**I**] If
Kfx ----- [e-**K**] f	f ---- [i-**K**] Kfx

Lecture 7

Introduction and elimination rules of combinators

Elim. Rules	**Intro. Rules**
Cfx --- [e-**C**] xf	xf --- [i-**C**] **C**fx
Bfxy ----- [e-**B**] f(xy)	f(xy) ---- [i-**B**] **B**fxy
Φfxyz ----- [e-Φ] f(xz)(yz)	f(xz)(yz) ---- [i-Φ] Φfxyz

Lecture 7

Combinators vs. λ -expressions

The most important difference between the CL and λ-calculus is the use of the bounded variables.

Every combinator is an λ -expression.

$$\mathbf{B}fg \equiv \lambda x.f(gx)$$
$$\mathbf{T}x \equiv \lambda f.fx$$
$$\mathbf{S}fg \equiv \lambda x.fx(gx)$$

Lecture 7

Application to natural language parsing

John is brilliant

- The predicate *is brilliant* is an operator which operate on the operand John to construct the final proposition.
- The applicative representation associated to this analysis is the following:

(is-brillant)John

- We define the operator **John*** as being constructed from the lexicon *John* by

[John* = **C*** John].

1. John* (is-brillant)
2. [John* = **C*** John]
3. **C***John (is-brillant)
4. is-brillant (John)

Application to natural language parsing

John is brilliant in λ-term

Operator John* by λ-expression

$$[John^* = \lambda x.x\ (John')]$$

1. John*(λx.is-brilliant'(x))
2. [John* = λx.x (John')]
3. (λx.x(John'))(λx.is-brilliant'(x))
4. (λx.is-brilliant'(x))(John')
5. is-brillinat'(John')

Passivisation

Consider the following sentences

a. The man has been killed.

b. One has killed him.

→ Invariant of meaning
→ Relation between two sentences

:a. unary passive predicate (*has-been-killed*)

:b. active transitive predicate (*have-killed*)

Definition of the operator of passivisation 'PASS'

$$[PASS = B \sum C = \sum \circ C]$$

where B and C are the combinator of composition and of conversion and where $\sum$ is the existential quantificator which, by applying to a binary predicate, transforms it into the unary predicate.

Definition of the operator of passivisation 'PASS'

$$[PASS = B \sum C = \sum \circ C]$$

1/	has-been-killed (the-man)	*hypothesis*
2/	[has-been-killed=PASS(has killed)]	*passive lexical predicate*
3/	PASS (has-killed)(the-man)	*repl.2.,1.*
4/	[PASS = ***B*** $\sum$ ***C***]	*definition of 'PASS'*
5/	***B*** $\sum$ ***C*** (has-killed)(the-man)	*repl.4.,3.*
6/	$\sum$ (***C***(has-killed))(the-man)	[e-***B***]
7/	(***C***(has-killed)) x (the-man)	[e-$\sum$]
8/	(has-killed)(the-main) x	[e-***C***]
9/	[x in the agentive subject position = *one*]	*definition of 'one'*
10/	(has-killed)(the-man)*one*	*repl.9.,8., normal form*

Definition of the operator of passivisation 'PASS'

We establish the paraphrastic relation between the passive sentence with expressed agent and its active counterpart:

The man has been killed by the enemy

↓

The enemy has killed the man

Definition of the operator of passivisation 'PASS'

Relation between give-to and receive-from

z *gives* y *to* x

↕

x *receives* y *from* x

The lexical predicate "*give-to*" has a predicate converse associated to "*receive-from*";

[receive-from z y x = give-to x y z]

Definition of the operator of passivisation 'PASS'

1/ **(receive-from) z y x**

2/ **C**((receive-from) z) x y

3/ **BC**(receive-from) z x y

4/ **C(BC**(receive-from)) z x y

5/ **C(C(BC**(receive-from)) x) y z

6/ **BC(C(BC**(receive-from))) x y z

7/ [give-to=**BC(C(BC**(receive-from)))]

8/ **give-to x y z**

Combinators used in CCG

Motivation of applying the combinators to natural language parsing

- Linguistic: complex phenomena of natural language applicable to the various languages
- Informatics: left to right parsing (LR)
  ex: reduce the spurious-ambiguity

Parsing a sentence in CCG

Step 1: tokenization

Step 2: tagging the concatenated lexicon

Step 3: calculate on types attributed to the concatenated lexicons by applying the adequate combinatorial rules

Step 4: eliminate the applied combinators (we will see how to do on next week)

Step 5: finding the parsing results presented in the form of an operator/operand structure (predicate -argument structure)

Parsing a sentence in CCG

Example: *I requested and would prefer musicals*

STEP 1 : tokenization/lemmatization → *ex) POS Tagger, tokenizer, lemmatizer*

a. *I-requested-and-would-prefer-musicals*

b. *I-request-ed-and-would-prefer-musical-s*

STEP 2 : tagging the concatenated expressions → *ex) Supertagger, Inventory of typed words*

I	NP
Requested	$(S\backslash NP)/NP$
And	$CONJ$
Would	$(S\backslash NP)/VP$
Prefer	VP/NP
musicals	NP

Parsing a sentence in CCG

STEP 3 : categorial calculus

a. apply the type-raising rules → *Subject Type-raising* $(>T)$
$NP : a \Rightarrow T/(T\backslash NP) : Ta$

b. apply the functional composition rules → *Forward Composition:* $(>B)$
$X/Y : f \quad Y/Z : g \Rightarrow X/Z : Bfg$

c. apply the coordination rules → *Coordination:* $(<\&>)$
$X \; conj \; X \Rightarrow X$

	I-	requested-	and-	would-	prefer-	musicals	
1/	NP	$(S\backslash NP)/NP$	CONJ	$(S\backslash NP)/VP$	VP/NP	NP	
2/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$	CONJ	$(S\backslash NP)/VP$	VP/NP	NP	(>T)
3/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$	CONJ	$(S\backslash NP)/NP$		NP	(>B)
4/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$				NP	$(>\Phi)$
5/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$				NP	(>B)
6/	S/NP					NP	(>)
7/			**S**				

Parsing a sentence in CCG

STEP 4 : semantic representation (predicate-argument structure)

	I	requested	and	would	prefer	musicals
1/	:i’	:request’	:and’	: will’	:prefer’	: musicals’

2/ :λf.f I’

3/ : λx.λy.will’(prefer’x)y

4/ : λxλy.and’(will’(prefer’x)y)(request’xy)

5/ : λxλy.and’(will’(prefer’x)y)(request’xy)

6/ :λy.and’(would’(prefer’ musicals’)y)(request’ musicals’ y)

7/S: and’(will’(prefer’ musicals’) i’)(request’ musicals’ i’)

Semantic representation in term of the *combinators*

	I-	requested	and-	would-	prefer	musicals
1/	NP	(S\NP)/NP	CONJ	(S\NP)/VP	VP/NP	NP
2/	S/(S\NP)	(S\NP)/NP	CONJ	(S\NP)/VP	VP/NP	NP (>T)
	C*I	requested	and	would	prefer	musicals
3/	S/(S\NP)	(S\NP)/NP	CONJ	(S\NP)/NP		NP (>B)
	C*I	requested and		B would prefer		musicals
4/	S/(S\NP)	(S\NP)/NP				NP (> Φ)
	C*I	Φ and requested (B would prefer)				musicals
5/	S/NP					NP (>B)
	B((C*I)(Φ and requested (B would prefer)))					musicals
6/	S					(>)
	B((C*I)(Φ and requested (B would prefer))) musicals					

Semantic representation in term of the *combinators*

I requested and would prefer musicals

S: B((C*I)(Φ and requested (B would prefer))) musicals

1/	**B((C*I)(Φ and requested (B would prefer))) musicals**	
2/	**(C*I)((Φ and requested (B would prefer))) musicals)**	**[e-B]**
3/	**((Φ and requested (B would prefer))) musicals) I**	**[e-C*]**
4/	**(and (requested musicals) ((B would prefer) musicals)) I**	**[e-Φ]**
5/	**((and (requested musicals) (would (prefer musicals))) I)**	**[e-B]**

Normal form

A normal form is a combinatory expression which is irreducible in the sense that it contain any occurrence of a redex.

If a combinatory expression X reduce to a combinatory expression N which is in normal form, so N is called the normal form of X.

Example

Bxyz is reducible to x(yz).
x(yz) is a normal form of the combinatory expression **B**xyz.

Normal form

Example

Prove xyz is the normal form of **BBC**xyz.

$$\mathbf{BBC}xyz \rightarrow_\beta xyz$$

1/	**BBC**xyz	
2/	**C**(**C**x)yz	[e-**B**]
3/	**C**xzy	[e-**C**]
4/	xyz	[e-**C**]

Classwork

Give the semantic representation in term of combinators.
Please refer to the given paper on last lecture on CCG Parsing.