Computational Semantics

Section 18.1 – 18.2 in Textbook

hannes@ru.is

T-725-MALV

Semantic Analysis

- Process of **constructing meaning representations** from linguistic expressions
- Requires **knowledge**, e.g.
  - Meaning of words
  - Conventional meaning of grammar constructs
  - Structure of discourse
  - Common sense knowledge about topic
  - Knowledge about state of world
- ..and inference techniques

Let's Start Modestly

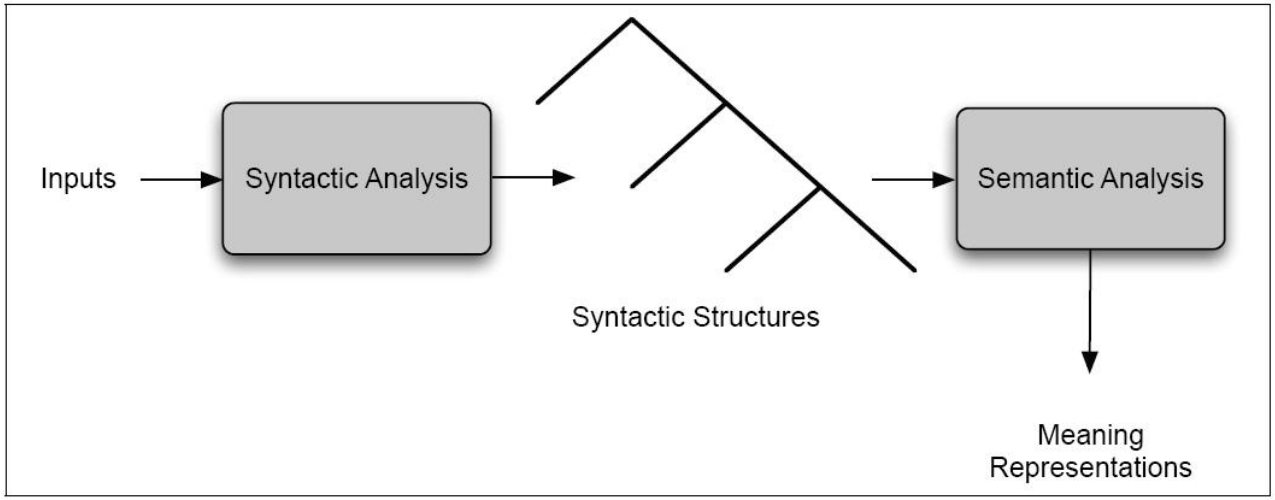
- We focus on **syntax-driven semantic analysis**
- The knowledge we will require is only gleaned from the **lexicon** and the **grammar**
- This may be enough for some use and may also serve as first step for further processing

Principle of Compositionality

- Approach is based on the principle of **compositionality**
- Meaning of a sentence can be constructed from the **meanings of its parts**
- But are **word** meanings **enough** then? What about "I see what I eat" vs. "I eat what I see"?
- The meaning is also based on **ordering** and **grouping** of words and **relations** among them in a sentence (i.e. syntactic structure)

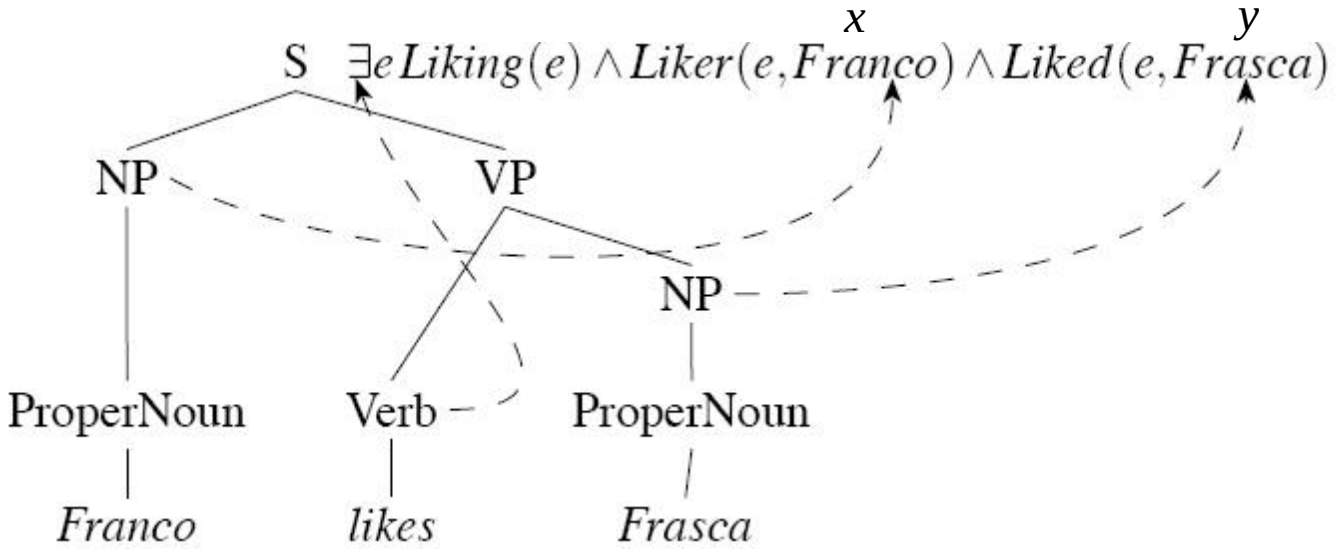
Pipeline

- A **Semantic Analyzer** processes the output of a syntax analyzer to produce meaning representation



Simplified Idea

- Grab nouns and use **template** based on verb



- **Problem:** Need knowledge about this particular example and parse tree

Solving the Problem

- The right place for semantic knowledge in a syntax-directed approach is with the **finite set of devices** that generate the trees in the first place:
  - Grammar rules
  - Lexical entries
- We augment context-free grammars with **semantic attachments**

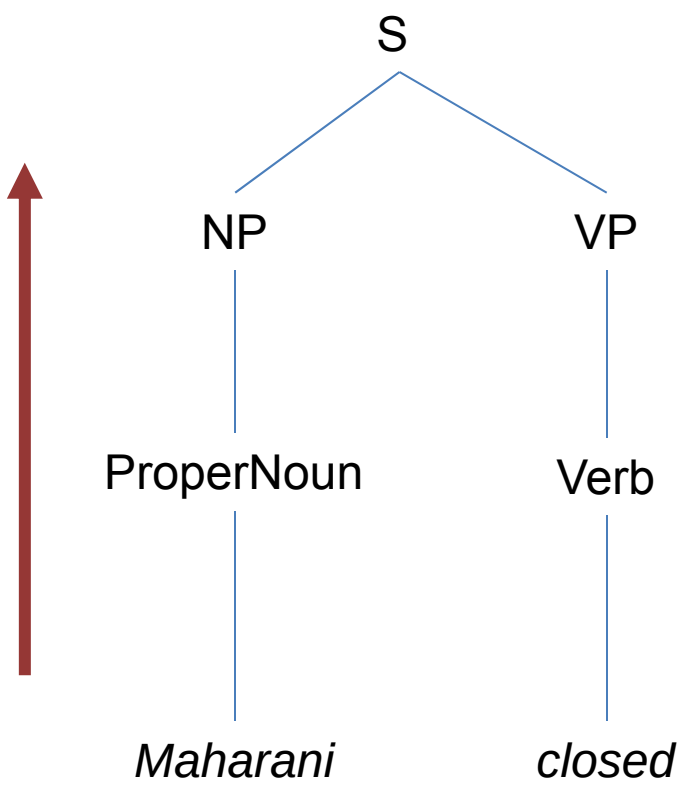
Semantic Attachments

- **Semantic attachments** are **instructions** specifying how to compute **meaning** representation of a construction **from** meanings of constituent **parts**

$$A \rightarrow \alpha_1 \,..\, \alpha_n \quad \{ f(\alpha_j.sem, ..., \alpha_k.sem) \}$$

Basic Example

FOL: Closed(Maharani)



NL: "Maharani closed."

semantic attachments

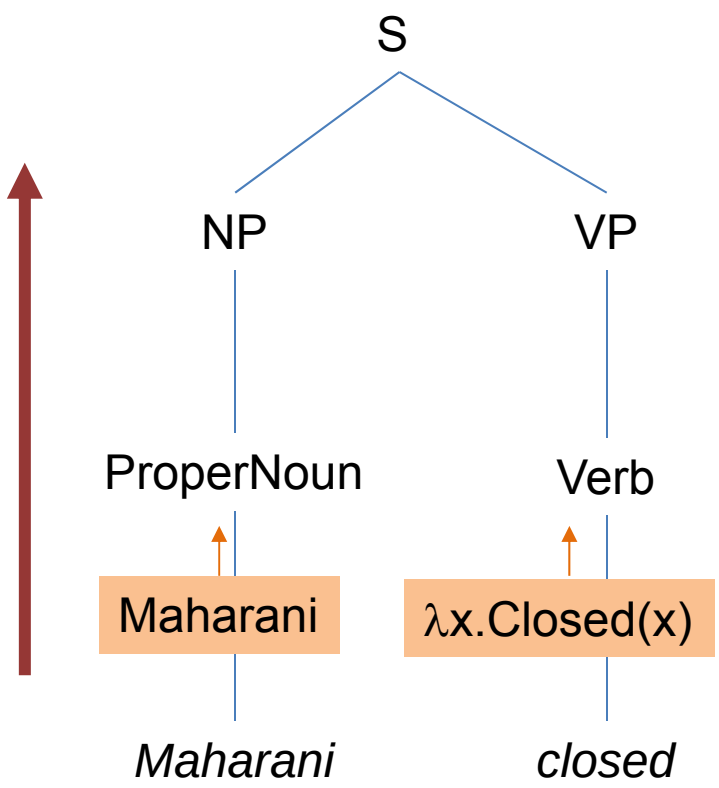
Grammar

S→ NP VP	{ VP.sem(NP.sem) }
NP → ProperNoun	{ ProperNoun.sem }
VP → Verb	{ Verb.sem }

Lexicon

ProperNoun → *Maharani*	{ Maharani }
Verb → *closed*	{ λx.Closed(x) }

Basic Example



NL: "Maharani closed."

semantic attachments

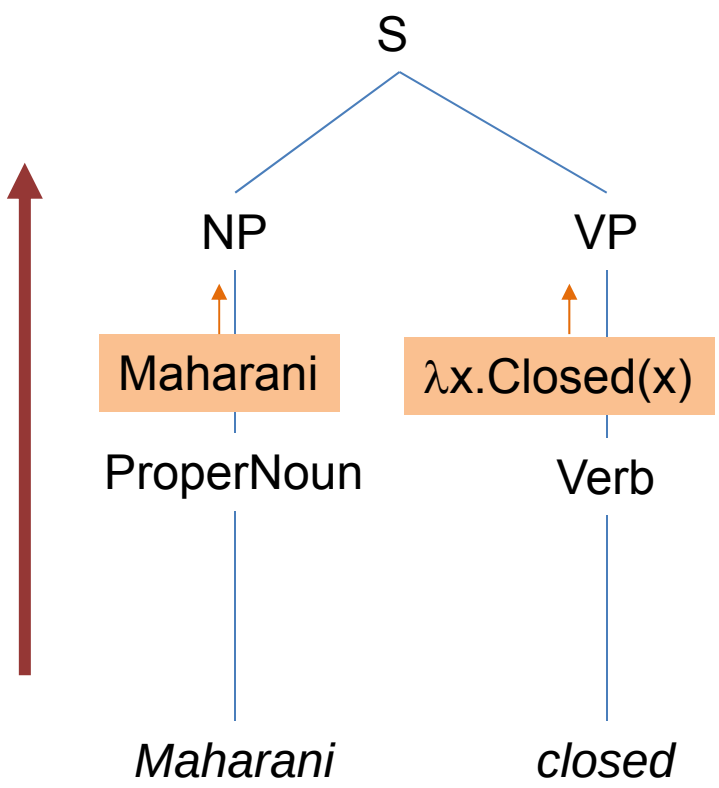
Grammar

S→ NP VP	{ VP.sem(NP.sem) }
NP → ProperNoun	{ ProperNoun.sem }
VP → Verb	{ Verb.sem }

Lexicon

ProperNoun → *Maharani*	{ Maharani }
Verb → *closed*	{ λx.Closed(x) }

Basic Example



NL: "Maharani closed."

semantic attachments

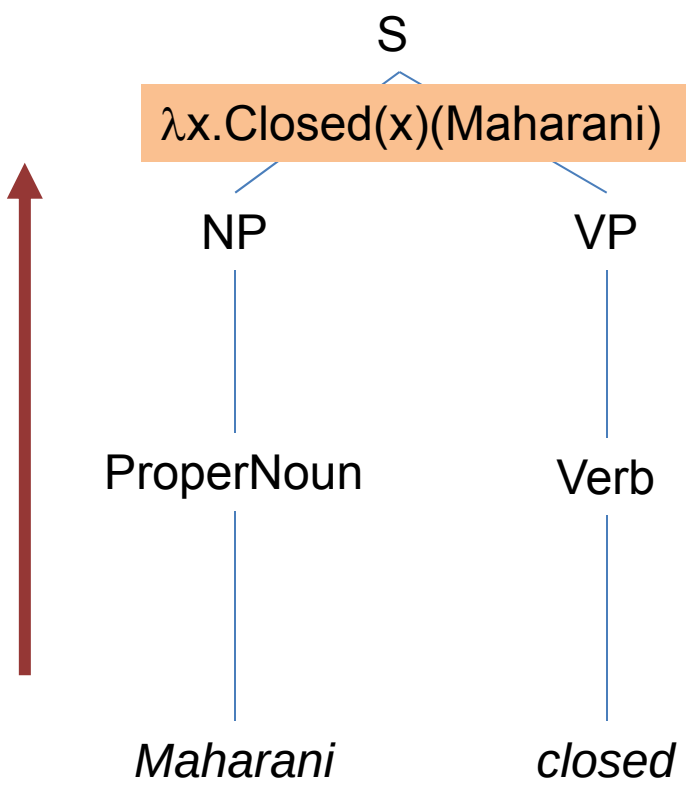
Grammar

S→ NP VP	{ VP.sem(NP.sem) }
NP → ProperNoun	{ ProperNoun.sem }
VP → Verb	{ Verb.sem }

Lexicon

ProperNoun → *Maharani*	{ Maharani }
Verb → *closed*	{ λx.Closed(x) }

Basic Example



NL: "Maharani closed."

semantic attachments

Grammar

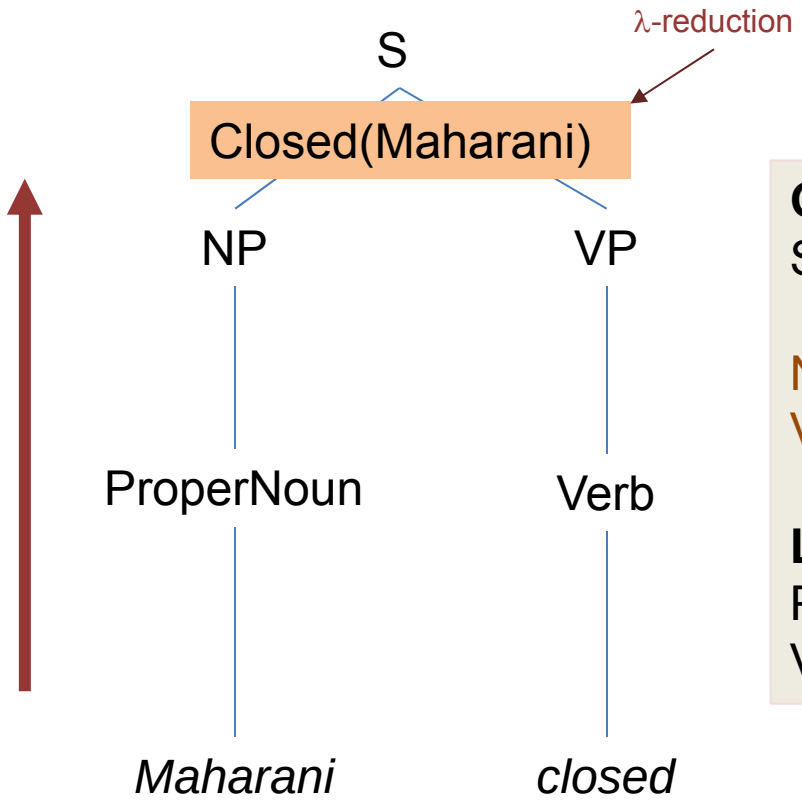
S→ NP VP	{ VP.sem(NP.sem) }
NP → ProperNoun	{ ProperNoun.sem }
VP → Verb	{ Verb.sem }

Lexicon

ProperNoun → *Maharani*	{ Maharani }
Verb → *closed*	{ λx.Closed(x) }

Basic Example

FOL: Closed(Maharani)



semantic attachments

Grammar

S→ NP VP	{ VP.sem(NP.sem) }
NP → ProperNoun	{ ProperNoun.sem }
VP → Verb	{ Verb.sem }

Lexicon

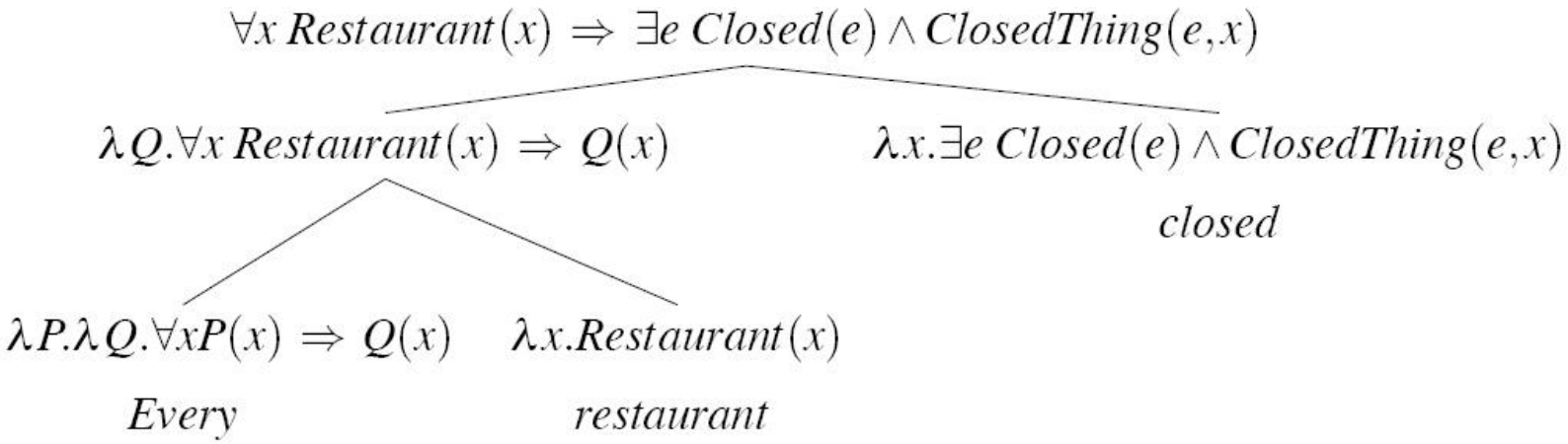
ProperNoun → *Maharani*	{ Maharani }
Verb → *closed*	{ λx.Closed(x) }

NL: "Maharani closed."

General Pattern

- Semantic attachments to **grammar rules** consists primarily of **λ-reductions**
  - One element serving as functor
  - Other elements serving as arguments
- Bulk of the **meaning** representations introduced **in lexicon**

More Complex Example



Grammar Rule	Semantic Attachment
$S \rightarrow NP\ VP$	$\{NP.sem(VP.sem)\}$
$NP \rightarrow Det\ Nominal$	$\{Det.sem(Nominal.sem)\}$
$NP \rightarrow ProperNoun$	$\{ProperNoun.sem\}$
$Nominal \rightarrow Noun$	$\{Noun.sem\}$
$VP \rightarrow Verb$	$\{Verb.sem\}$
$VP \rightarrow Verb\ NP$	$\{Verb.sem(NP.sem)\}$
$Det \rightarrow every$	$\{\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)\}$
$Det \rightarrow a$	$\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)\}$
$Noun \rightarrow restaurant$	$\{\lambda r.Restaurant(r)\}$
$ProperNoun \rightarrow Matthew$	$\{\lambda m.m(Matthew)\}$
$ProperNoun \rightarrow Franco$	$\{\lambda f.f(Franco)\}$
$ProperNoun \rightarrow Frasca$	$\{\lambda f.f(Frasca)\}$
$Verb \rightarrow closed$	$\{\lambda x.\exists e Closed(e) \wedge ClosedThing(e,x)\}$
$Verb \rightarrow opened$	$\{\lambda w.\lambda z.w(\lambda x.\exists e Opened(e) \wedge Opener(e,z) \wedge Opened(e,x))\}$