L98: Introduction to Computational Semantics

Lecture 9: Modeling Syntactico-Semantic Composition

Weiwei Sun and Simone Teufel

Natural Language and Information Processing Research Group
Department of Computer Science and Technology
University of Cambridge

Lent 2021/22

Can you understand the sentences before watching the video? And why?

- Mama have you said, that you first your homework make must.
- When you this ready have, then you can to Julia go.
- Lara, can you me please out the bath a towel bring?

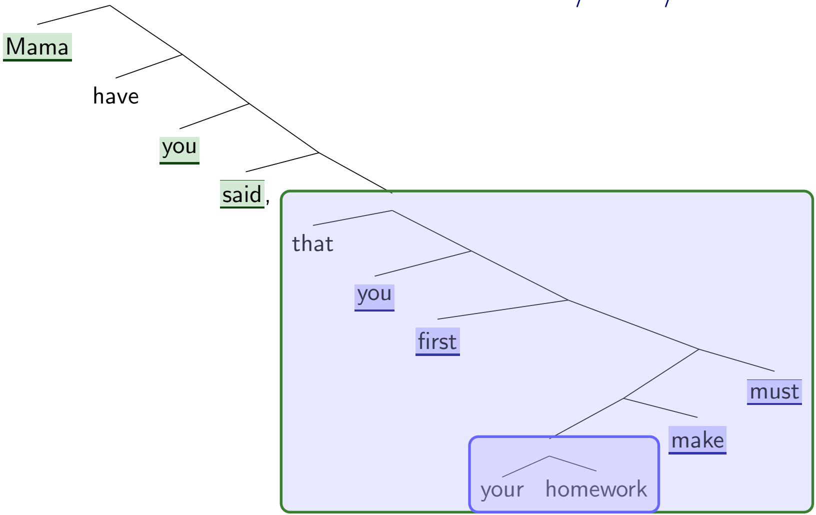
from Knallerfrauen

Lecture 9: Modeling Syntactico-Semantic Composition

1. Representational vs derivational
2. Locality
3. Context-free graph rewriting

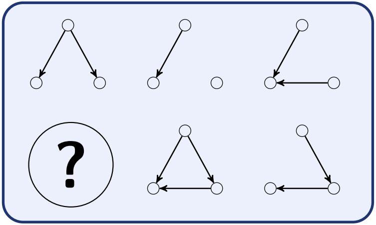
Representational vs Derivational

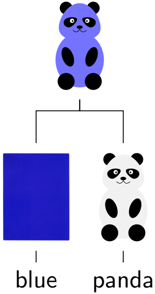
I can understand because of the sister/aunt/. . . relations



- Arguments/adjuncts should **c-command** a target verb.
- A node in a syntactic tree c-commands its sister node and all of its sister's descendants

Representational vs derivational





Modeling syntactico-semantic composition/derivation

- Assume the complicated structure is generated step-by-step.
- And assume that it is relatively easy to make a decision for a single step.
- An internal structure, e.g. tree, is used to represent the process.
- We don't directly evaluate the *goodness* of the target structure, which is the result of a derivation.
- We directly evaluate the *goodness* of the derivation structure, and get the derived structure (for free).
- Parsing and generation share a model, probably like human language processing.

Composition

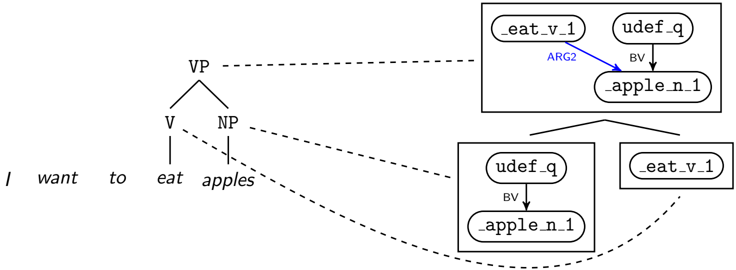
I want to eat apples

Composition

Compositional Parsing

semantic/meaning representation parsing: mapping a sentence to an MR, such as semantic graph.

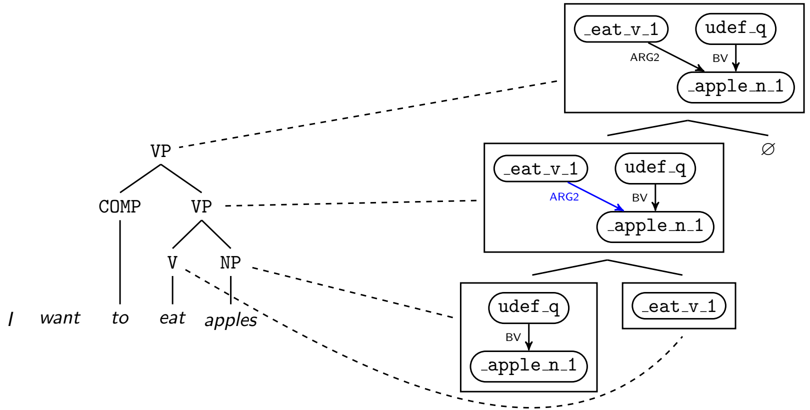
step 1: assign semantic interpretations to "words". The elementary MR for *apples* is graph with a single node. We mainly do "word sense disambiguation" in this step.
step 2: combine graphs according to syntactico-semantic rules. We merge the "eat" and "apples" graphs by augmenting VP→V NP. We add the blue edge to link the two graphs. We should view the blue edge as a third graph.



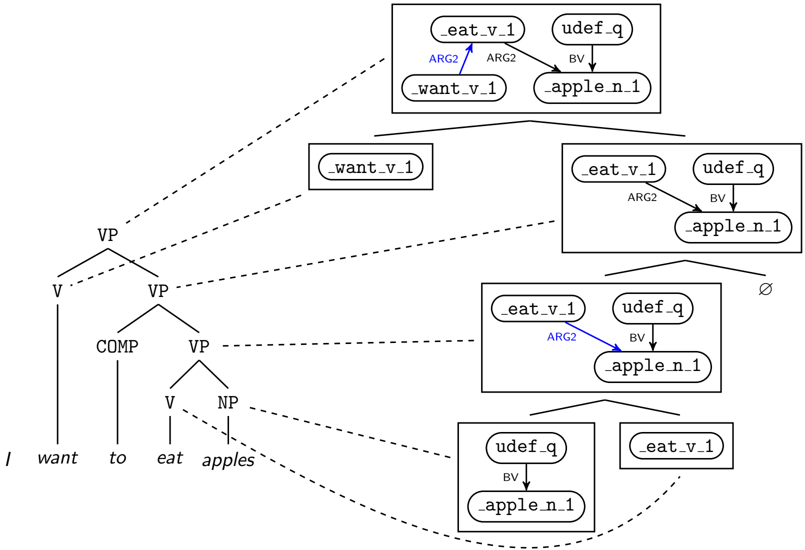
Composition

Compositional Parsing

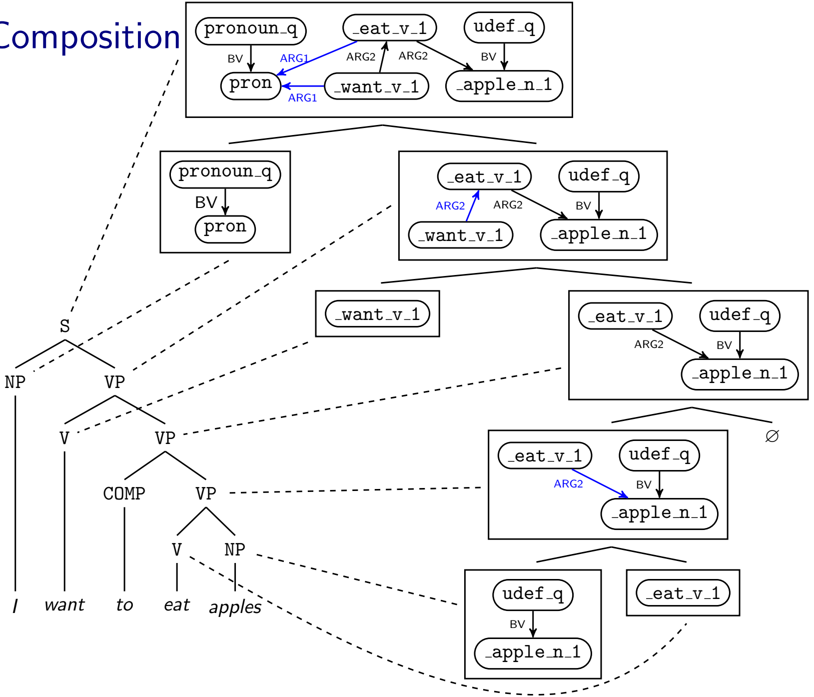
step 1: assign semantic interpretations to "to", which is an empty graph.
step 2: glue the "eat apples" graph with an empty graph.
continue: iterate the above process.



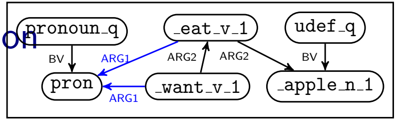
Composition



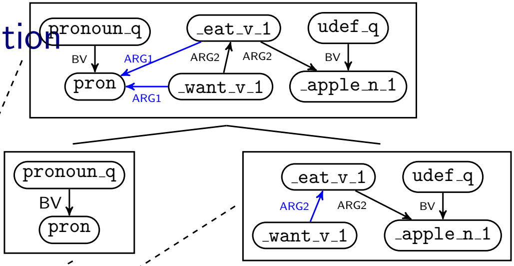
Composition

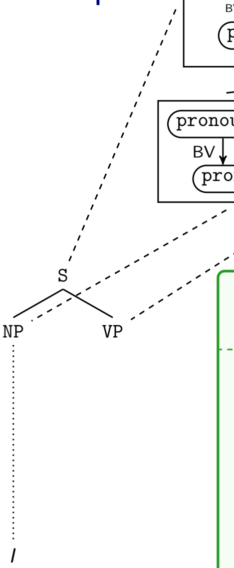


Decomposition



Decomposition



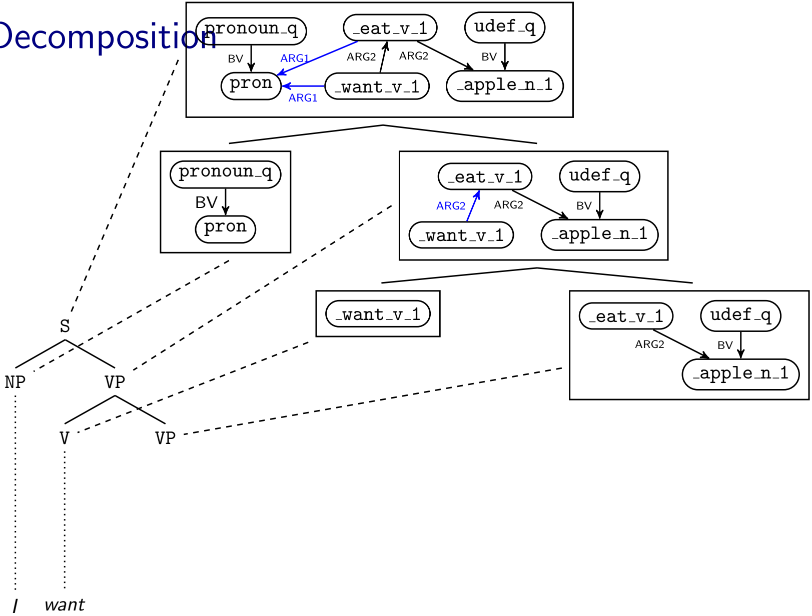


Compositional Surface Realisation

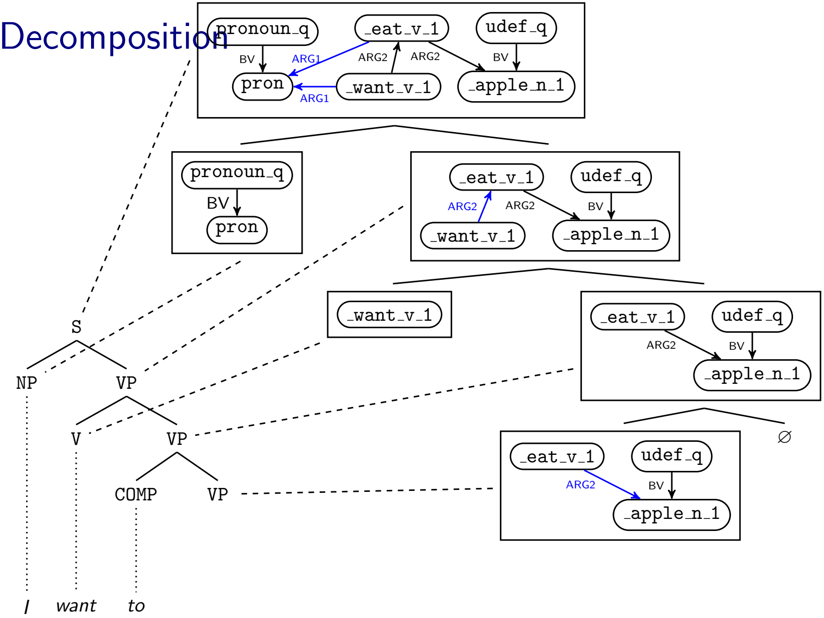
surface realisation: mapping an MR, such as semantic graph, to a string.

We want to recursively decompose the input graph until we meet elementary graphs that correspond to "words".
In each step, we cut a big graph into two smaller ones. In fact, we should view the blue edges as a third graph.
Our scissors are syntactico-semantic rules.
In the first step to handle current example, we get S→NP VP on the syntactic side, which puts the subject NP before the predicate VP.

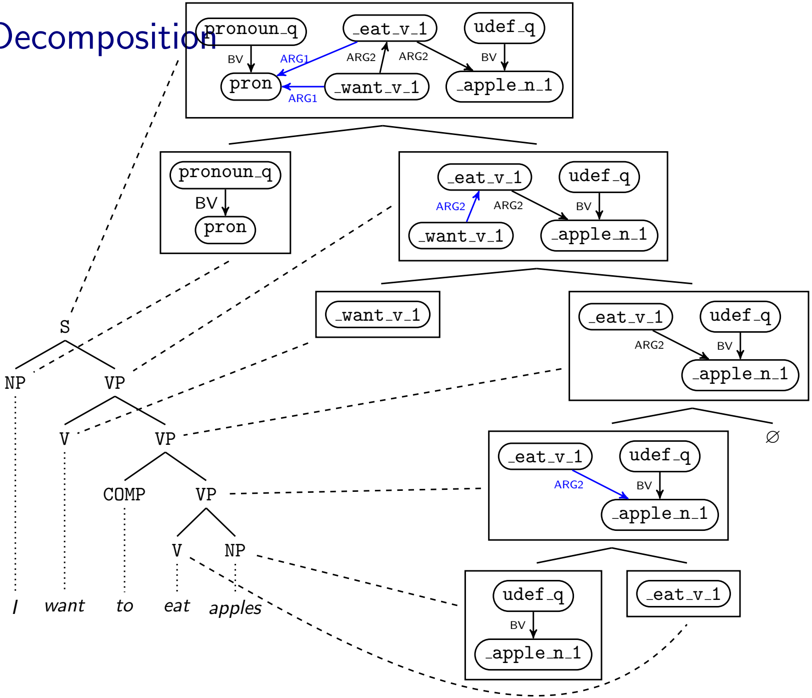
Decomposition



Decomposition



Decomposition



Locality

Can you understand?

- Mama have you said, that you first your homework make must.
- When you this ready have, then you can to Julia go.

(1) Mama first have you said, that you your homework make must.

! It is hard for me to understand (1), because it breaks the sister/aunt/... pattern, which is found in a majority of natural languages.

Is the distance between *you* and *make* long?

Mama have you said, that you first your homework, *which you should do but haven't done yet,* make must.

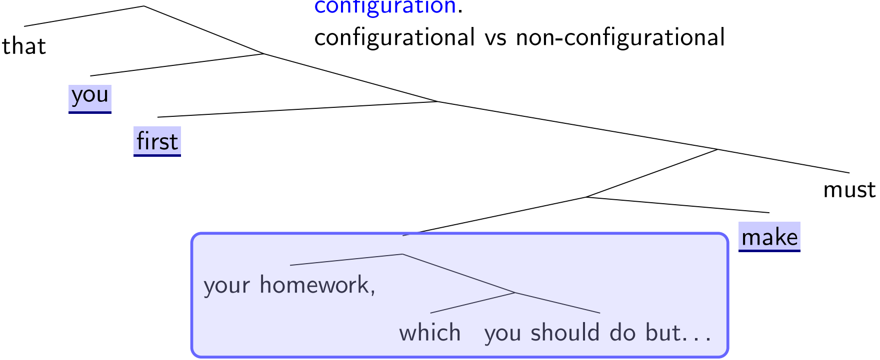
> ! 11 words in between; but still local wrt *tree*

Is the distance between *you* and *make* long?

Mama have you said, that you first your homework, *which you should do but haven't done yet,* make must.

! 11 words in between; but still local wrt *tree*

the calculation of distance is based on the syntactic configuration.
configurational vs non-configurational



Can you understand?

- Mama have you said, that you first your homework make must.

(2) Your homework, Mama have you said, that you first make must.

- I can understand this because this is a very frequent phenomenon in Mandarin.
- Li and Thompson distinguish topic-prominent languages, such as Mandarin and Japanese, from subject-prominent languages, such as English.
- A topic-prominent language its morphology/syntax to emphasize the topic-comment structure of the sentence.

Unbounded dependency constructions (UDC)

Topicalisation

(3) a. Kim, Sandy loves.

b. Your homework, Mama have you said, that you first make must.

Wh-movement

(4) a. Who do you think Bob saw?

b. Who do you think Bob said he saw?

c. Who do you think Bob said he imagined that he saw?

- Some sentences exhibit phrases that appear "out of place" based on simple head–argument or head–modifier constraints.
- The distance from the position of the "dislocated" phrase to its "natural home" can be quite far (in the limit, unbounded).

More UDCs

- The "dislocated" phrase is called "trace", denoted as $\_{i}$ or t_i.
- We use subscript $_i$, $_j$, etc. to indicate a discourse referent.

Relative clause

(5) a. This is the man [who$_i$ Sandy loves $\_{i}$]. ▷*Wh*-**relative clause**

b. This is [the man]$_i$ [Sandy loves $\_{i}$]. ▷**Reduced relative clause**

Clefts

(6) a. It is Kim$_i$ [who$_i$ Sandy loves $\_{i}$]. ▷*It*-**clefts**

b. It is Kim$_i$ [Sandy loves $\_{i}$].

(7) [What$_i$ Sandy loves $\_{i}$] is Kim$_i$. ▷**Pseudoclefts**

And more...

A syntactic link is needed

(8) a. Kim_i, Sandy trusts $\_{i}$.

b. $[\text{On Kim}]_i$, Sandy depends $\_{i}$.

(9) a. *$[\text{On Kim}]_i$, Sandy trusts $\_{i}$.

b. *Kim_i, Sandy depends $\_{i}$.

(10) a. Kim_i, Ada believes Bob knows Sandy trusts $\_{i}$.

b. $[\text{On Kim}]_i$, Ada believes Bob knows Sandy depends $\_{i}$.

(11) a. *$[\text{On Kim}]_i$, Ada believes Bob knows Sandy trusts $\_{i}$.

b. *Kim_i, Ada believes Bob knows Sandy depends $\_{i}$.

! This link has to be established for an unbounded length.

Bounded vs unbounded

(12) a. Sandy$_i$ is hard to love $\_{i}$. ▷ *Tough* **construction**

b. [This question]$_i$ is tough to answer $\_{i}$.

c. Kim$_i$ is easy (for John) to please $\_{i}$.

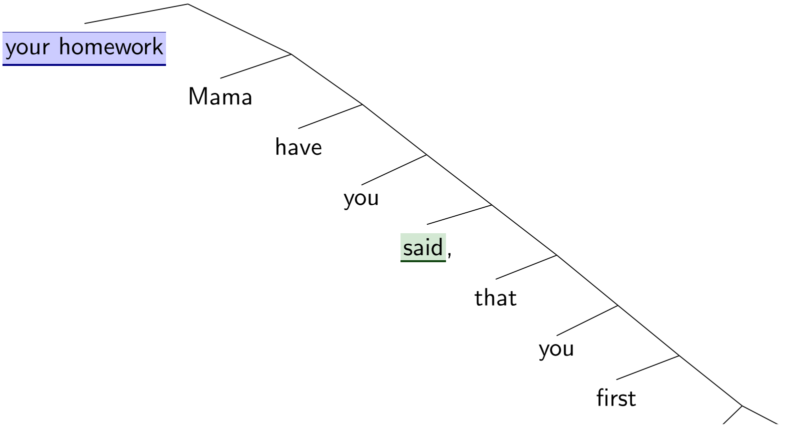
d. Kim$_i$ is easy to prove that Mary asked Paul to bribe $\_{i}$.

(13) a. Kim seems to love Sandy. ▷**raising**

b. Kim wants to prove that. ▷**control**

! What is the difference to the raising/control construction?
The corresponding dependencies in the raising/control construction is bounded.

Challenge 1: long distance



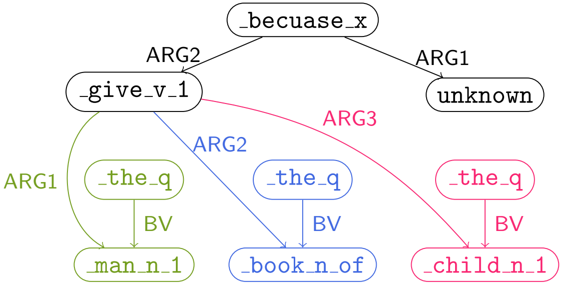
Modeling syntactico-semantic composition/derivation

- In each step, the rule should be “local”.
- The non-local/long-distance dependency is derived by combining all “local” rules.
- We can achieve this by augmenting phrase-structure rules.

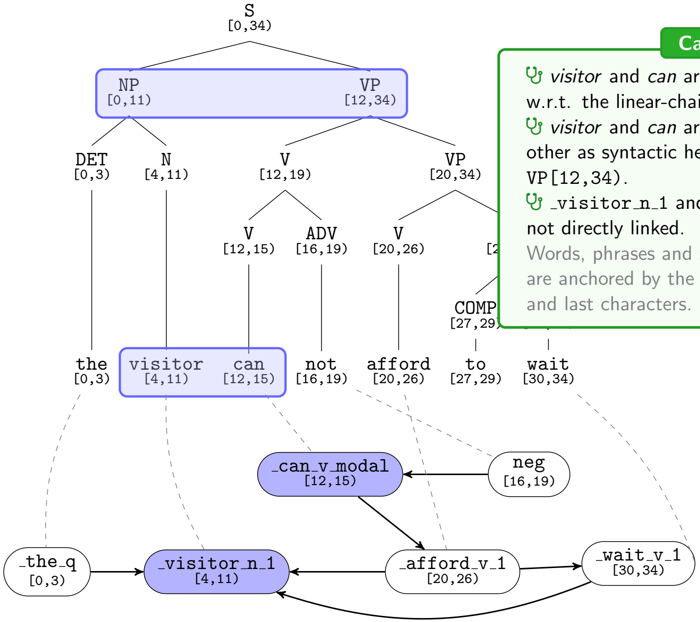
Case and word order in German

because the man gives the book to the child							
weil	*der*	*Mann*	*dem*	*Kind*	*das*	*Buch*	*gibt*
weil	*der*	*Mann*	*das*	*Buch*	*dem*	*Kind*	*gibt*
weil	*das*	*Buch*	*der*	*Mann*	*dem*	*Kind*	*gibt*
weil	*das*	*Buch*	*dem*	*Kind*	*der*	*Mann*	*gibt*
weil	*dem*	*Kind*	*der*	*Mann*	*das*	*Buch*	*gibt*
weil	*dem*	*Kind*	*das*	*Buch*	*der*	*Mann*	*gibt*

from S. Müller's course



Challenge 2: syntax–semantics mismatch?

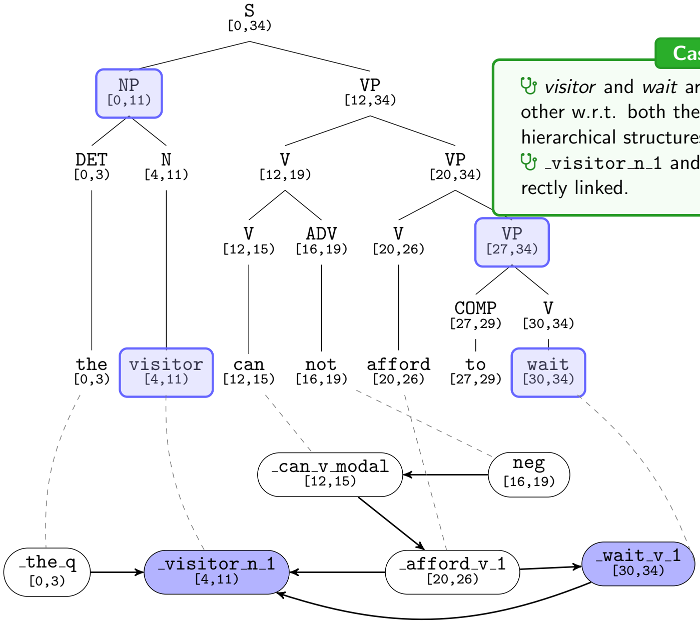


Case 1

- *visitor* and *can* are next to each other w.r.t. the linear-chain structure.
- *visitor* and *can* are also next to each other as syntactic heads of NP[0,11) and VP[12,34).
- _visitor_n_1 and _can_v_modal are not directly linked.

Words, phrases and semantic predicates are anchored by the positions of their first and last characters.

Challenge 2: syntax–semantics mismatch?

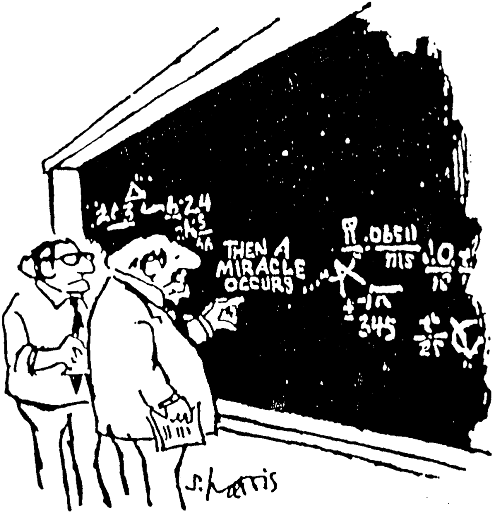


Case 2

- *visitor* and *wait* are distant from each other w.r.t. both the linear-chain and the hierarchical structures.
- _visitor_n_1 and _wait_v_1 are directly linked.

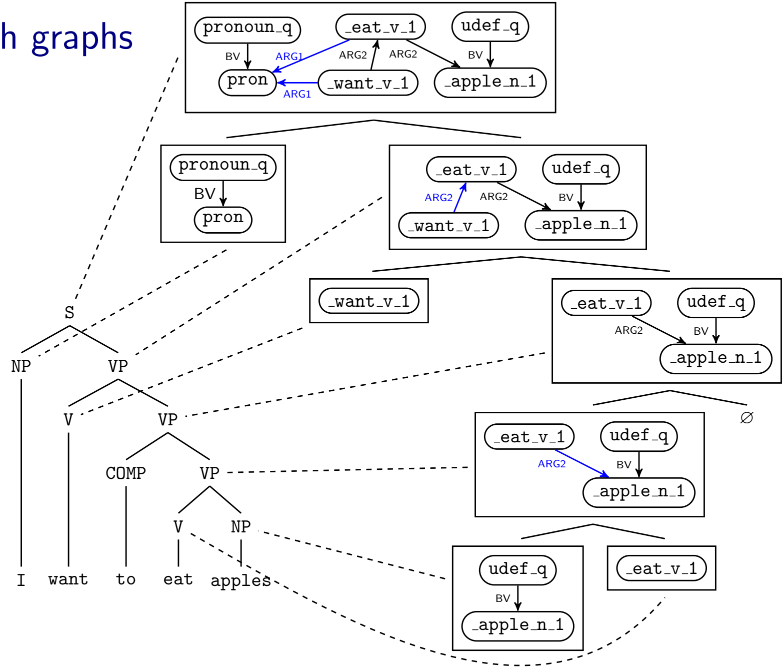
Context-Free Graph Rewriting

I think you should be more explicit in syntactico-semantic composition



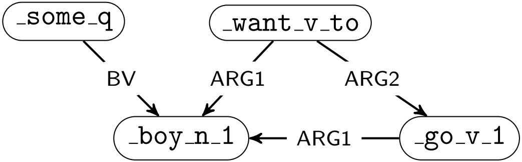
"I think you should be more explicit here in step two"

With graphs



pronoun_q
_eat_v_1
udef_q
BV
ARG1
ARG2
ARG2
BV
pron
_want_v_1
_apple_n_1
ARG1
S
NP
VP
V
VP
COMP
VP
V
NP
I
want
to
eat
apples
∅

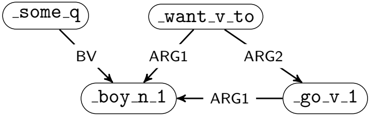
Hypergraph

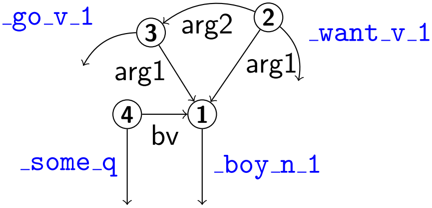


A graph consists of:

- A set of nodes.
- A set of edges connecting two nodes.

Hypergraph

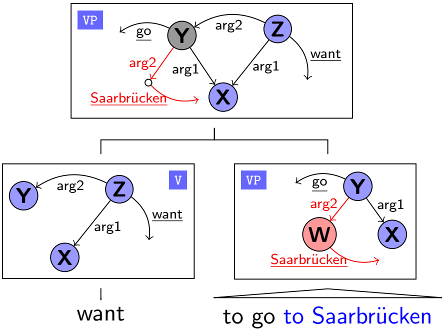




A hypergraph adds:

- Hyperedges connecting any number of nodes.
- A single node can be treated as an edge.

Hyperedge Replacement Grammar [1]



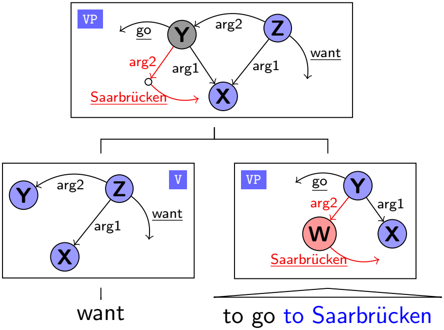
λ ∼ external node

When we combine two graphs, we don't need they know every detail of each other.

Only very few nodes of each graphs should be accessable. All other nodes are internal and won't participate in further composition. E.g. the "Saarbrüecken" node is invisible outside the phrase "to go to Saarbüecken".

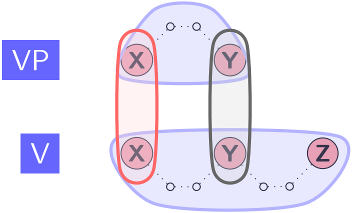
The few accessable nodes are called "external nodes", and they together make up an hyperedge.

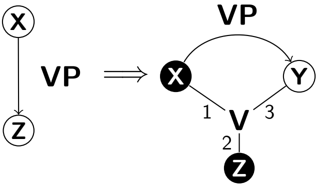
Hyperedge Replacement Grammar [1]



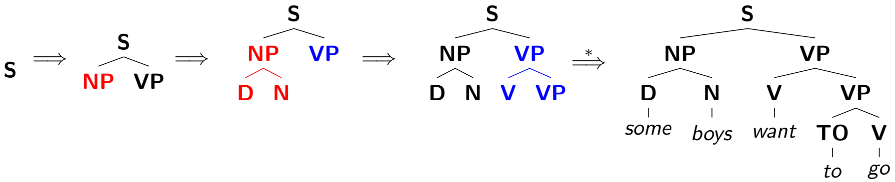
λ ∼ external node

- Graphs are glued at their corresponding external nodes.
- After a new bigger graph is created, some nodes no longer participate in further composition. Then, we "forget" them.





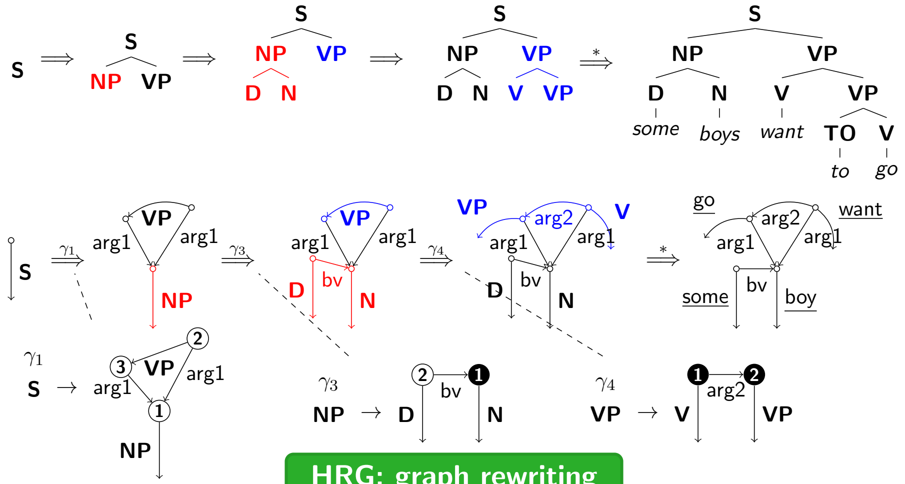
Symbol rewriting and graph rewriting



CFG: symbol rewriting

When we derive according to a CFG, we iteratively rewrite non-terminal symbols.
E.g. `S` is rewritten to `NP VP`.

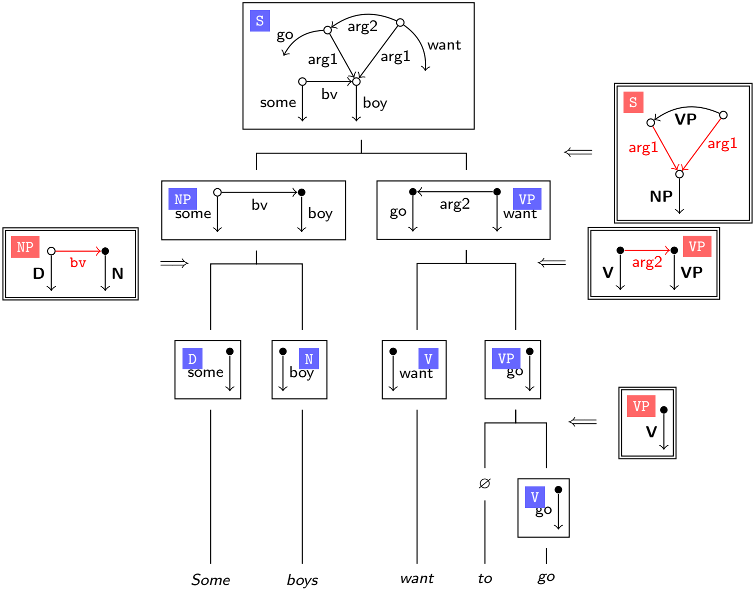
Symbol rewriting and graph rewriting



HRG: graph rewriting

We iteratively rewrite non-terminal hyperedges, i.e. hyperedges with non-terminal labels. Each hyperedge is replaced by a hypergraph. Rules should and could be linguistically-informed. γ_1: control; γ_3: quantification; γ_4: verb–object

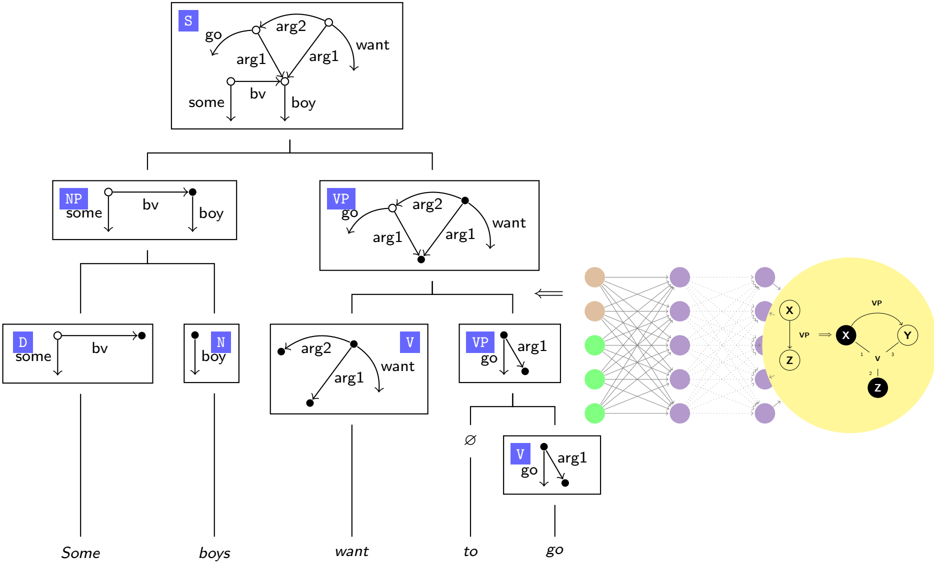
Derivation structure is tree; derived structure is graph



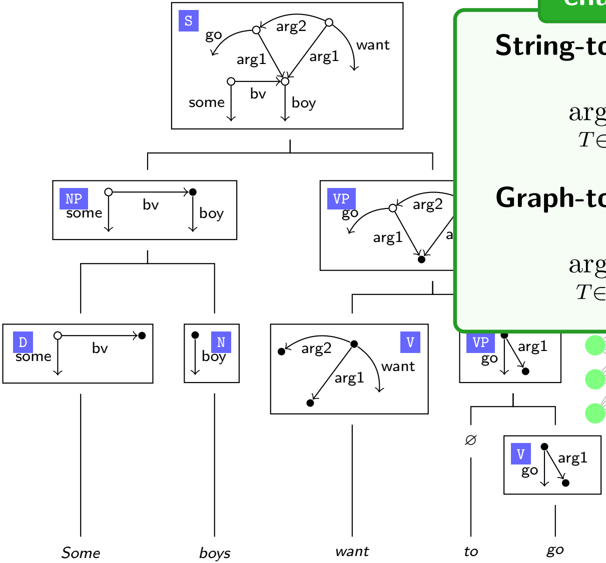
Derivation structure is tree

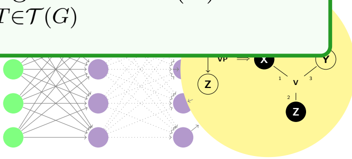
- The derived structure is a complicated graph, while the derivation structure is a seemly simpler tree.
- A very general approach to understand complex structures.
- For programming languages, compilers build abstract syntactic trees.
- For categorial grammars, categories like "S\NP" are merged along with a tree.

Scoring a derivation tree step-by-step



Scoring a derivation tree step-by-step





enumerating trees

String-to-graph parsing:

$$\arg\max_{T \in \mathcal{T}(x)} \text{SCORE}(T)$$

Graph-to-string Parsing:

$$\arg\max_{T \in \mathcal{T}(G)} \text{SCORE}(T)$$

Graph parsing and graph parsing

- Task 1: graph parsing (=string-to-graph parsing).
- Task 2: graph parsing (=graph-to-string parsing).
- To solve both tasks, we need to score a derivation tree, but search for the best tree in different spaces.
- For task 1, the search space is denoted as $\mathcal{T}(x)$, where x is the input string. We enumerate all trees that are compatble to x, or say all trees licensed by our grammar.

$$\arg\max_{T \in \mathcal{T}(x)} \textsc{score}(T)$$

- For task 2, the search space is denoted as $\mathcal{T}(G)$, where G is the input meaning representation. We enumerate all trees that are compatble to G, again, according to our grammar.

$$\arg\max_{T \in \mathcal{T}(G)} \textsc{score}(T)$$

- It is relatively straightforward to score a tree by summation over all rules applied.

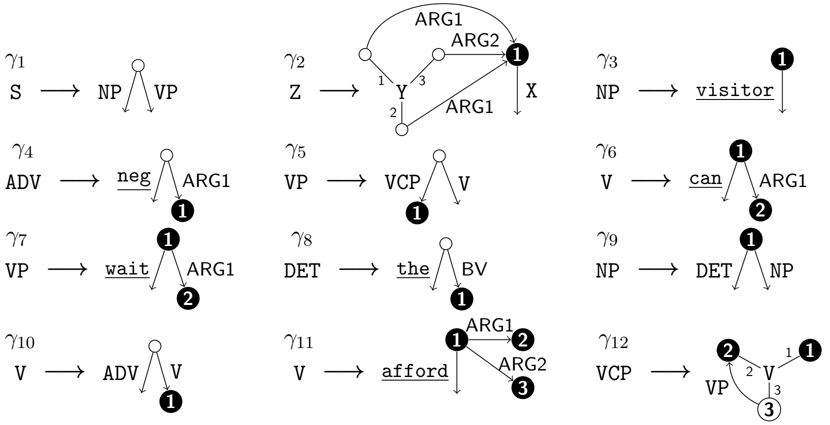
Exercise:

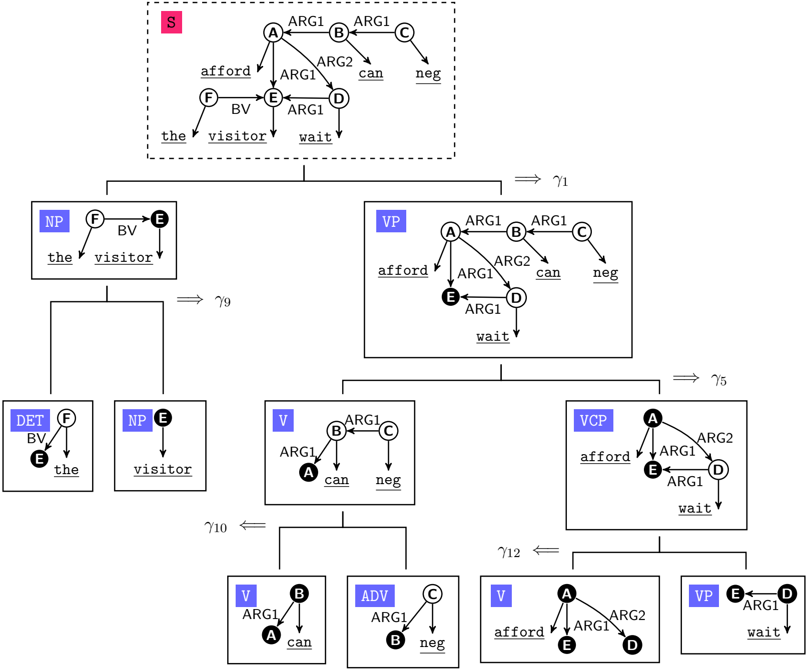
- Pre-lecture: word order video from Knallerfrauen
- Post-lecture:
  - read the two slides behind to understand how some syntax–semantics mismatches are handled.
  - analyze a UDC example with HRG.

Readings:

- Y. Chen and W. Sun. Parsing into Variable-in-situ Logico-Semantic Graphs.
- Y. Ye and W. Sun. Exact yet Efficient Graph Parsing, Bi-directional Locality and the Constructivist Hypothesis.

Example: more rules




S
ARG1
ARG2
BV
afford
can
neg
the
visitor
wait
⟹ γ1
NP
VP
⟹ γ9
DET
V
VCP
⟹ γ5
γ10 ⟸
γ12 ⟸
ADV

References I

F. Drewes, H.-J. Kreowski, and A. Habel.
Hyperedge Replacement Graph Grammars.
In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, pages 95–162. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1997.