LING 473: Day 14

START THE RECORDING

Deep Processing

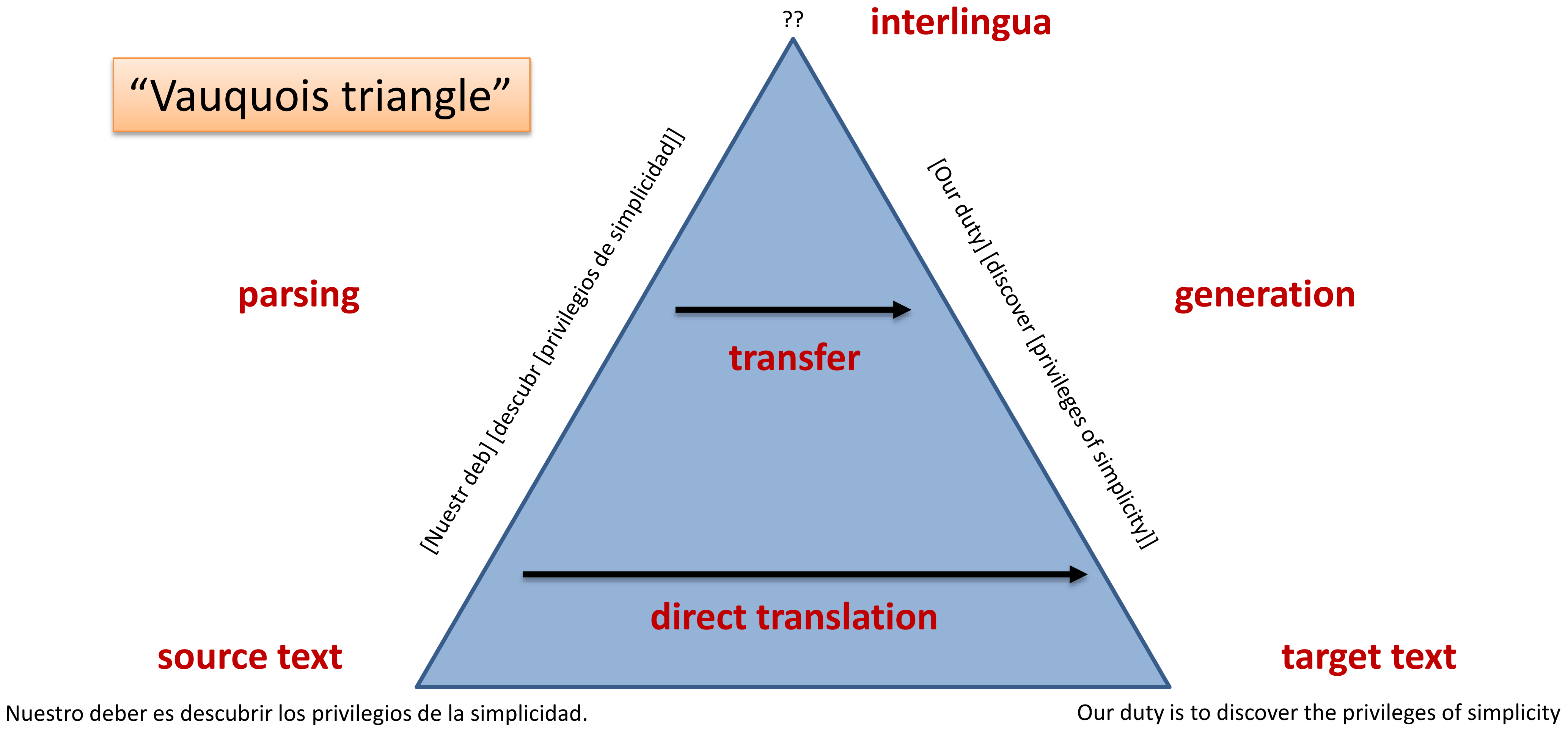
Reminders

- Course reviews available (check e-mail)
- Writing assignment is due today
- Project 5 is due Thursday.
- We’ll go over the self quiz solutions Thursday
- Course Evaluations

Deep *vs.* shallow

- Analytical methods:
  - "deep," rule-based
  - linguistically informed/motivated
  - traditionally rules have been hand-crafted
    - but see Poon and Domingos 2009
    http://www.cs.washington.edu/homes/pedrod/papers/emnlp09.pdf
  - system development feedback: direct
- Statistical methods
  - "shallow," automatically-extracted
  - development feedback from results: indirect

Semantic transfer machine translation



HPSG (Head-driven Phrase Structure Grammar)

- Highly consistent and powerful formalism
- Declarative, non-derivational, lexicalist, constraint-based
- Has been used to model many different languages
- Some large-scale implementations

HPSG foundations: Typed Feature Structures

- Typed Feature Structures
- High expressive power
- Parsing complexity: exponential (to the input length)
- Tractable with efficient parsing algorithms
- Efficiency can be improved with a well designed grammar

Feature Structures In Unification-Based Grammar Development

- A feature structure is a set of attribute-value pairs
  - Each attribute (or feature) is an atomic symbol
  - The value of each attribute can be either atomic, or complex (a feature structure, a list, or a set)

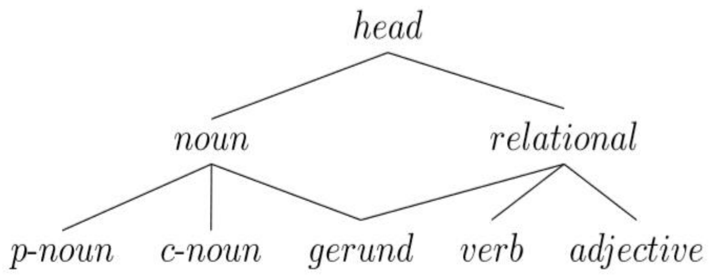
$$\begin{bmatrix} \text{CATEGORY} & \textit{noun-phrase} \\ \text{AGREEMENT} & \begin{bmatrix} \text{PERSON} & \textit{3rd} \\ \text{NUMBER} & \textit{sing} \end{bmatrix} \end{bmatrix}$$

Feature Structure Example

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{subord-with-verbal-comp-lex-item}} \\
\text{SYNSEM.LOCAL} & \left[\begin{array}{ll}
\text{CAT} & \left[\begin{array}{ll}
\text{HEAD} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{adp}} \\
\text{MC} & - \\
\text{MOD} & \left\langle \left[\begin{array}{ll}
\text{CAT.HEAD} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{verb}} \\
\text{VAL} & \left[\text{COMPS}\ \langle\ \rangle\right]
\end{array}\right] \\
\text{CONT.HOOK} & \left[\text{LTOP}\ \boxed{1}\right]
\end{array}\right] \right\rangle
\end{array}\right] \\
\text{VAL} & \left[\begin{array}{ll}
\text{COMPS} & \left\langle \left[\begin{array}{ll}
\text{CAT} & \left[\begin{array}{ll}
\text{MC} & - \\
\text{HEAD} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{verb}} \\
\text{VAL} & \left[\text{COMPS}\ \langle\ \rangle\right]
\end{array}\right]
\end{array}\right] \\
\text{CONT.HOOK} & \left[\begin{array}{ll}
\text{LTOP} & \boxed{2} \\
\text{INDEX} & \boxed{3}
\end{array}\right]
\end{array}\right] \right\rangle
\end{array}\right]
\end{array}\right] \\
\text{CONT} & \left[\text{HOOK.INDEX}\ \boxed{3}\right]
\end{array}\right] \\
\text{SYNSEM.LKEYS} & \left[\text{KEYREL} \left[\begin{array}{ll}
\text{ARG1} & \textit{qeq}\ (\boxed{1}) \\
\text{ARG2} & \textit{qeq}\ (\boxed{2})
\end{array}\right]\right]
\end{array}\right]
$$

Typed Feature Structure (TFS)

- A typed feature structure is composed of a (possibly empty) set of attribute-value pairs with each value being a TFS
- Each value is typed in some hierarchy (like types in programming languages)



(Malouf, 2000)

Properties of TFSes

- Finiteness

  a typed feature structure has a finite number of nodes

- Unique root and connectedness

  a typed feature structure has a unique root node; apart from the root, all nodes have at least one parent

- No cycles

  no node has an arc that points back to the root node or to another node that intervenes between the node itself and the root

- Unique features

  no node has two features with the same name and different values

- Typing

  each node has single type which is defined in the hierarchy

Type hierarchy

- In the DELPH-IN joint reference formalism:
  - A unique most general type: *top* T
  - Each non-top type has one or more parent type(s)
  - Two types are compatible if they share at least one offspring type
  - Each non-top type is associated with optional constraints
    - Constraints specified in ancestor types are monotonically inherited
    - Constraints (either inherited, or newly introduced) must be compatible

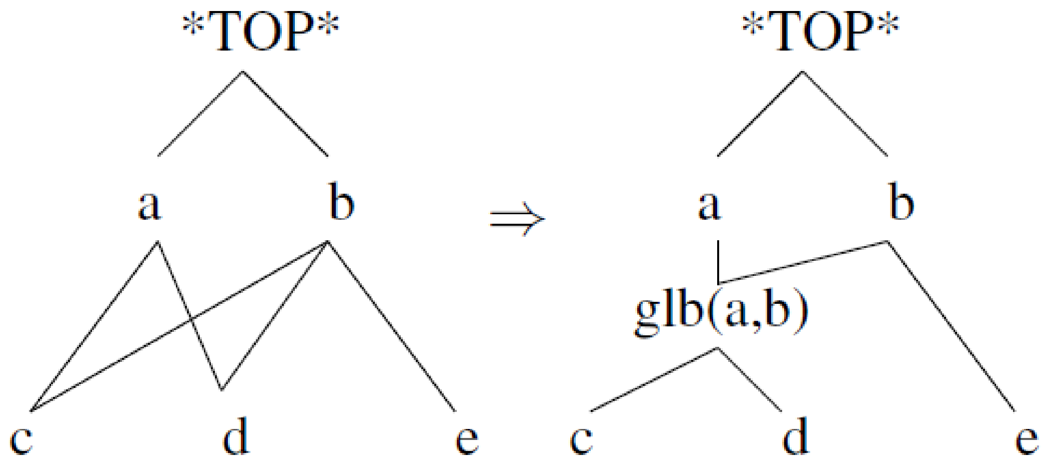
Unification

The unification result on two TFSes TFS_a and TFS_b is:

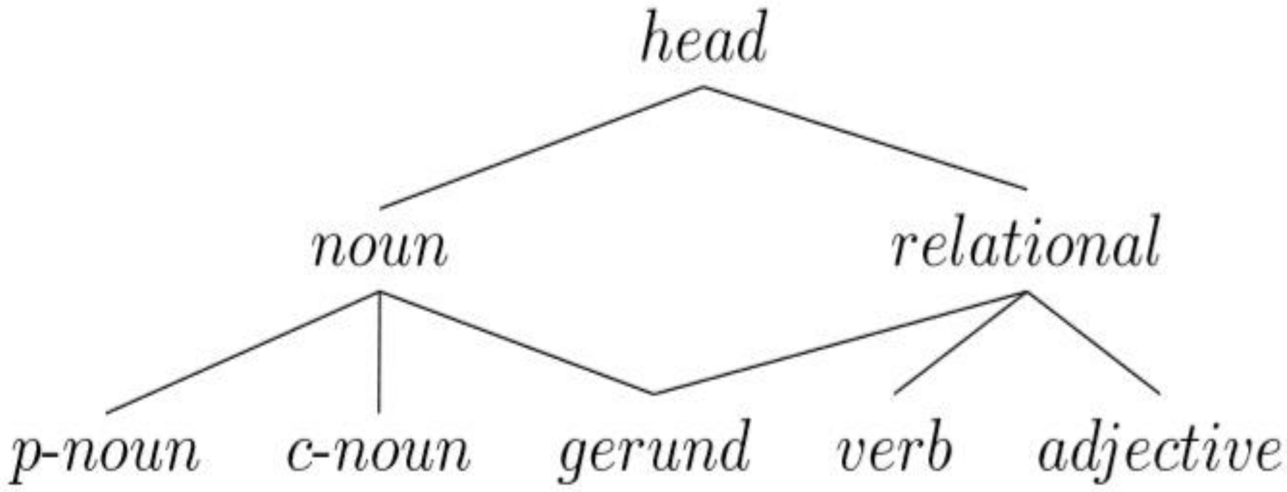
- $\perp$, if either one of the following:
  - type a and b are incompatible
  - unification of values for attribute X in TFS_a and TFS_b returns $\perp$
- a new TFS, with:
  - the most general shared subtype of a and b
  - a set of attribute-value pairs being the results of unifications on sub-TFSes of TFS_a and TFS_b

GLB Types

- In case of multiple inheritance, two types can have more than one shared subtype that neither is more general than the others
- Non-deterministic unification results
- Type hierarchy can be automatically modified to avoid this



Type Hierarchy Examples



(Malouf, 2000)

Semantics desiderata

- For each sentence admitted by the grammar, we want to produce a meaning representation suitable for applying rules of inference.

"This fierce dog chased that angry cat."

$$\mathrm{this}(x) \wedge \mathrm{fierce}(x) \wedge \mathrm{dog}(x) \wedge \\ \mathrm{chased}(e, x, y) \wedge \\ \mathrm{that}(y) \wedge \mathrm{angry}(y) \wedge \mathrm{cat}(y)$$

Semantics desiderata

- Compositionality
  - The meaning of a phrase is composed of the meanings of its parts.
    - (Where is this wrong?)
- Existing machinery
  - Unification is the only mechanism we use for constructing semantics in the grammar.

Semantics in feature structures

- Semantic content in the CONT attribute of every word and phrase

$$
\begin{bmatrix}
\textit{sign} & \\
\text{SYNSEM} & \begin{bmatrix}
\text{LOCAL} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix}
\text{HEAD} & \begin{bmatrix} \textit{head} \\ \begin{bmatrix} \text{MOD} & \langle \ldots \rangle \end{bmatrix} \end{bmatrix} \\
\text{VAL} & \begin{bmatrix} \text{SUBJ} & \langle \ldots \rangle \\ \text{SPR} & \langle \ldots \rangle \\ \text{COMPS} & \langle \ldots \rangle \end{bmatrix}
\end{bmatrix} \\
\textbf{CONT} & \begin{bmatrix} \text{HOOK} & [\,] \\ \text{RELS} & \langle \ldots \rangle \\ \text{HCONS} & \langle \ldots \rangle \end{bmatrix}
\end{bmatrix} \\
\text{NON-LOCAL} & \begin{bmatrix} \text{SLASH} \ \langle \ldots \rangle \end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Semantics formalism: MRS

- Minimal Recursion Semantics

  Copestake, A., Flickinger, D., Pollard, C. J., and Sag, I. A. (2005). *Minimal recursion semantics: an introduction*. Research on Language and Computation, 3(4):281–332.

- Used across DELPH-IN projects
- The value of CONT for a sentence is essentially a list of relations in the attribute RELS, with the arguments in those relations appropriately linked:
  - Semantic relations are introduced by lexical entries
  - Relations are appended when words are combined with other words or phrases.

MRS: example

RELS {

_the_q⟨0:3⟩	
LBL	*h4*
ARG0	*x6*
RSTR	*h7*
BODY	*h5*

_quick_a_1⟨4:9⟩	
LBL	*h8*
ARG0	*e9*
ARG1	*x6*

_brown_a_1⟨10:15⟩	
LBL	*h8*
ARG0	*e10*
ARG1	*x6*

_fox_n_1⟨16:19⟩	
LBL	*h8*
ARG0	*x6*

_jump_v_1⟨20:26⟩	
LBL	*h2*
ARG0	*e3*
ARG1	*x6*

_over_p_dir⟨27:31⟩	
LBL	*h2*
ARG0	*e11*
ARG1	*e3*
ARG2	*x12*

_the_q⟨32:35⟩	
LBL	*h13*
ARG0	*x12*
RSTR	*h15*
BODY	*h14*

_lazy_a_1⟨36:40⟩	
LBL	*h16*
ARG0	*e17*
ARG1	*x12*

_dog_n_1⟨41:44⟩	
LBL	*h16*
ARG0	*x12*

}

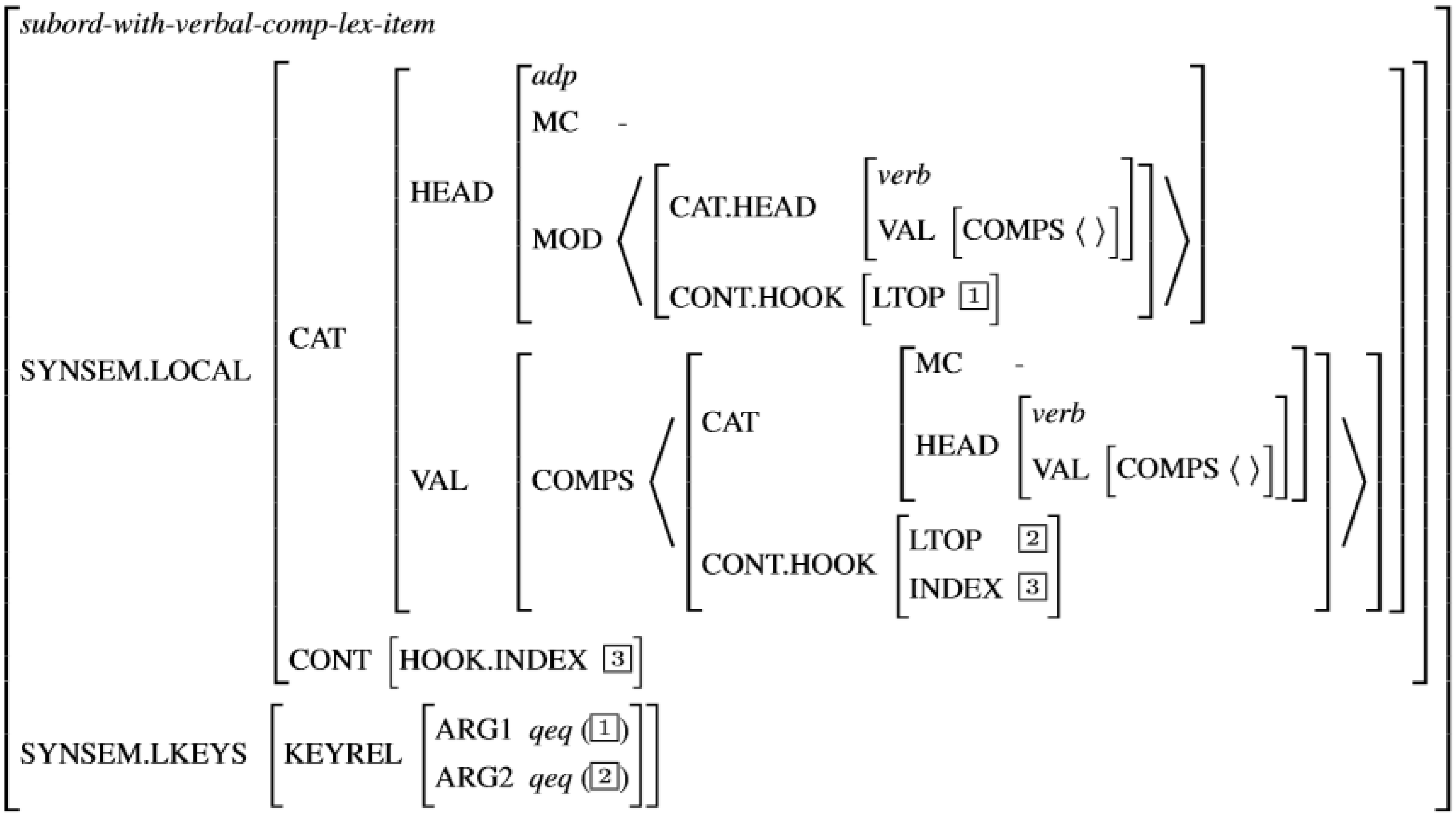
Linking semantic arguments

- Each word or phrase also ‘publishes’ an INDEX attribute in CONT.HOOK
- When heads select a complement or specifier, they constrain its INDEX value – an instance variable for nouns, an event variable for verbs.
- Each lexeme also specifies a KEY relation (to allow complex semantics)

Semantics of phrases

- Every phrase identifies its RELS attribute with the concatenation of its daughters' RELS lists
- Every phrase identifies its semantic INDEX value with the INDEX value of one of its daughters (the semantic head).
- Since we unify the *synsem* of a complement or specifier with the constraints in the head-daughter, unification also takes care of semantic linking.
- Head-modifier structures: the modifier constrains the semantic index of the modified head-daughter, and then rules unify the *synsem* of the head-daughter with the MOD value in the modifier.

Semantics in the Feature Structure



Semantic Representation

Kim left when Pat arrived

$$
\begin{array}{l}
\langle h_1, e_3, \\
\left| \begin{array}{l}
h_4\text{:proper\_q}(x_6,\ h_5,\ h_7), \\
h_8\text{:named}(x_6,\ Kim), \\
h_9\text{:\_leave\_v\_1}(e_3,\ x_6,\ p_{10}), \\
h_2\text{:\_when\_x\_subord}(e_{11},\ h_{12},\ h_{13}), \\
h_{14}\text{:proper\_q}(x_{16},\ h_{15},\ h_{17}), \\
h_{18}\text{:named}(x_{16},\ Pat), \\
h_{19}\text{:\_arrive\_v\_1}(e_{20},\ x_{16})
\end{array} \right| \\
\{ h_{15}\ qeq\ h_{18},\ h_{13}\ qeq\ h_{19},\ h_{12}\ qeq\ h_9,\ h_5\ qeq\ h_8,\ h_1\ qeq\ h_2 \} \rangle
\end{array}
$$

Parsing and Generation

- DELPH-IN computational grammars are bi-directional:

the quick brown fox jumped over the lazy dog

Parsing

Generation

RELS {

_the_q⟨0:3⟩		_quick_a_1⟨4:9⟩		_brown_a_1⟨10:15⟩		_fox_n_1⟨16:19⟩		_jump_v_1⟨20:26⟩	
LBL	h4	LBL	h8	LBL	h8	LBL	h8	LBL	h2
ARG0	x6	ARG0	e9	ARG0	e10	ARG0	x6	ARG0	e3
RSTR	h7	ARG1	x6	ARG1	x6			ARG1	x6
BODY	h5								

_over_p_dir⟨27:31⟩		_the_q⟨32:35⟩		_lazy_a_1⟨36:40⟩		_dog_n_1⟨41:44⟩	
LBL	h2	LBL	h13	LBL	h16	LBL	h16
ARG0	e11	ARG0	x12	ARG0	e17	ARG0	x12
ARG1	e3	RSTR	h15	ARG1	x12		
ARG2	x12	BODY	h14				

}

ERG Demo

- http://erg.delph-in.net/logon

Next Time

- Review & wrap-up