Computational Issues in the Parsing of Natural Language

Part 1: Selected Background Material (60%)

- The reason why generative parsing methods for programming languages are inadequate for natural languages.

- Modern techniques for parsing natural languages.

➢ This part is background; it is not about my research.

➢ It assumes knowledge of parsing in computer science (*i.e.,* CFG's), but no knowledge of computational linguistics.

Part 2: Current Research Topics (40%)

- Declarative specification of HPSG-style grammars.

- Computational management of partially specified type hierarchies.

Natural Language Processing and Computational Linguistics

A terminological distinction:

Natural Language Processing (NLP):

- Typically a component of a larger application.
- Restricted domain of linguistic discourse.
- Embodiment of linguistic principles secondary.

➢ Example: Natural-language interface to a help system for UNIX.

Computational Linguistics (CL):

- Study of formal models of language which are amenable to computation.
- Often involves stand-alone or demonstration implementation projects.
- Fairly general domain of linguistic discourse.
- Embodiment of linguistic principles essential.

Some major theories within computational linguistics:

- Lexical Functional Grammar (LFG).
- Head-Driven Phrase Structure Grammar (HPSG).
- Categorial Grammar.

➢ Example parsing systems : PATR, ALE, CUF, Troll, TFS, [Meurers'].

➢ Example projects within CL: Verbmobil.

Parsing

Parsing refers to the process of determining the syntactic and possibly semantic structure of a string.

In programming languages:

- We design the language (usually) to have certain properties, among them that of being amenable to parsing.
  - Notable exceptions: Ada, SQL

- Parsing base:
  - *Unambiguous* context-free grammar (BNF)
  - Semantic constraints to eliminate unwanted parses

- A given string has at most one parse (programs have unique derivations).

In natural language:

- We have to parse the language as it is.

- There are many strategies for parsing; some are based upon context-free grammars, some are not.

- A given sentence may have several parses.
  - Time flies like an arrow.
  - I saw the woman on the hill with a telescope.

- Correct parses may be strongly context dependent.
  - How do I print a file on the laser printer with the lines numbered?

- Parses may be rejected on purely semantic grounds.
  - John gave the book to Mary.
  - * John gave the book from Mary.
  - John gave the book from Mary to Marie.

Apparently similar constructions may be legal or illegal based upon fairly subtle rules.

- John hit the ball to Mary.
- John hit Mary the ball.
- John hit the ball to the wall.
- * John hit the wall the ball.

Pure Generative Parsing of NL

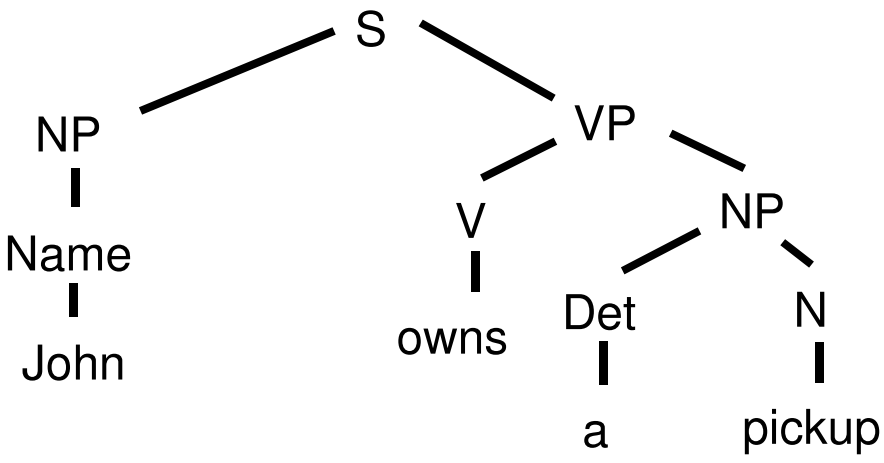
A context-free grammar (CFG) for a tiny fragment of English:

Syntactic Rules:
S → NP VP
VP → V | V NP
NP → Name | Det N

Lexical Rules:
Name → John | Mary | I | We | You
N → car | cars | truck | trucks | pickup | pickups
Det → a | an | the | two
V → own | owns | sleep | sleeps

A parse tree for the sentence:

"John owns a pickup."



The Problem with Pure Generative Parsing

Many ungrammatical sentences can also be generated.

- * John own an pickup.
- * We owns two pickup.
- * John owns.
- * Mary sleeps a truck.

These problems can be remedied by forcing number and verb-type distinction within the grammar.

To fix the verb-type distinction:

$VP \rightarrow V_I \mid V_T\ NP$
$V_I \rightarrow$ sleep $\mid$ sleeps
$V_T \rightarrow$ own $\mid$ owns

The number distinction on noun phrases:

$N_s \rightarrow$ car $\mid$truck $\mid$ pickup
$N_P \rightarrow$ cars $\mid$trucks $\mid$pickups
$Det_S \rightarrow$ a $\mid$ an $\mid$ the
$Det_P \rightarrow$ two
$NP \rightarrow Det_S\ N_s \mid Det_P\ N_P$

Why This Is Not a Good Solution

- These types of “fixes” result in a combinatorial explosion in the number of rules of the grammar.

- Representation of other languages will result in much more serious explosions in size:
  - Multiple genders on nouns which require agreement with determiners and adjectives:
    French, Swedish: 2; German, Norwegian: 3
  - Complex verb conjugation depending upon both number and person:
    German: 4; French: 5 or 6

Conclusion: CFG, by itself, is not an adequate tool for describing natural language.

Remarks:

- It is a debatable theoretical question as to whether languages such as English can be represented by CFG, in the *weak sense* of being able to generate exactly the legal sentences strings.

- It is easy to show that English cannot be generated by a CFG in the *strong sense*, in which the parse trees reflect the grammatical structure.

The Hybrid
Generative + Constraint Solution

- Approach:
  - CFG (the *context-free skeleton / backbone)* +
  - constraint set
- LFG (Lexical-Functional Grammar) uses this approach.

A simple example: The objects in the lexicon are augmented with types:

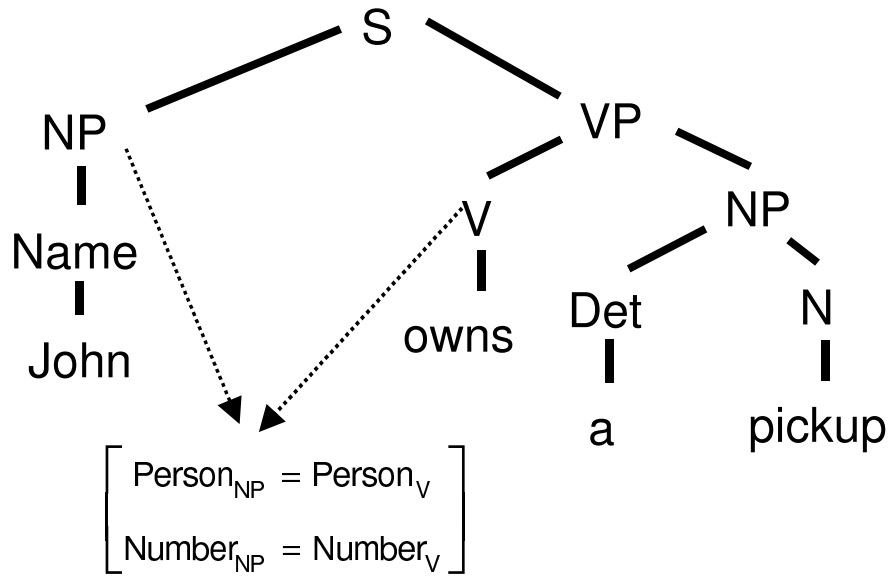
Some Names:

$$\begin{bmatrix} \text{Name} : & \text{John} \\ \text{Person} : & 3 \\ \text{Number} : & \text{Singular} \end{bmatrix} \qquad \begin{bmatrix} \text{Name} : & \text{We} \\ \text{Person} : & 1 \\ \text{Number} : & \text{Plural} \end{bmatrix}$$

A verb:

$$\begin{bmatrix} \text{Name} : & \text{owns} \\ \text{Person} : & 3 \\ \text{Number} : & \text{sing} \end{bmatrix} \qquad \begin{bmatrix} \text{Name} : & \text{own} \\ \text{Person} : \\ \text{Number} : \end{bmatrix} \text{any but sing} + 3 \text{ pers}$$

- The grammatical productions now have these constraints attached to them.
- Similar in idea to *indexed grammars* for programming languages.



The main idea of this approach thus consists of two steps:

- **Generate**: Find the candidate parses for the underlying CFG.
- **Constrain**: Use the feature constraints to eliminate undesired parses.

Decision properties:

- In general, the problem of deciding whether a sentence has a parse tree which satisfies the given constraints is undedicable.
  - ... because there can be infinitely many parse trees for a given string.

- However, it is decidable if we can find all parse trees algorithmically *(off-line parsing condition).*
  - ... in which case the number of parse trees is provably finite and identifiable.

Pure Constraint-Based Parsing

- In pure constraint based parsing (as exemplified by HPSG), there is no underlying context-free grammar.

- The entire parsing process is one of finding *models* of constraints, which are expressed in a *typed feature logic.*

- The main operation is *unification*; hence the techniques is also called unification-based parsing.

The following examples illustrate the idea of typed feature logic, but do **not** represent the HPSG model.

Feature Structures

Typed Feature structures are simply record-like entities.

Example:

Pascal Notation	Feature-Structure Notation

type Person = record
  Name: Name_type;
  ID_number: ID_type;
  Income: Integer
end record; {Person}
	$\tau_{\text{Person}} \begin{bmatrix} \text{Name:} & \tau_{\text{Name-type}} \\ \text{ID_number:} & \tau_{\text{ID_type}} \\ \text{Income:} & \tau_{\text{Integer}} \end{bmatrix}$

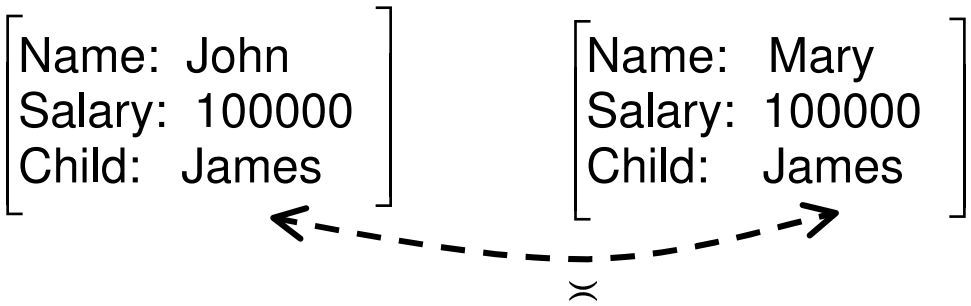
var P Person; P.Name := Smith; P.ID_number := 123456; P.Income := 100000;	$\begin{bmatrix} \text{Name:} & \text{Smith} \\ \text{ID_number:} & 123456 \\ \text{Income:} & 100000 \end{bmatrix}$

- Feature structures also admit direct and indirect recursion on types.

Pseudo Pascal	Feature Structure

type Node = record
  Value: Integer;
  Next: Node
end record {Node}
	$\tau_{\text{Node}} \begin{bmatrix} \text{Value:} & \tau_{\text{Integer}} \\ \text{Next:} & \tau_{\text{Node}} \end{bmatrix}$

Feature structures also admit *coalescing* of values.



Here James is the same person in each case.

This type of construction is critical in modelling linguistic situations in which the same object is referenced in two ways.

Gap-filler constructions:

- Jan$_j$ is easy to talk to ___$_j$ about problems of this sort.

- [Problems of this sort]$_i$, Jan$_j$ is easy to talk to ___$_j$ about ___$_i$.

Example from Norwegian:

- Han vasket sin bil.
- Han vasket hans bil.

Unification-Based Parsing

Recapture: $S \rightarrow NP\ VP$

$$\tau_S \begin{bmatrix} \text{Subj:} & \tau_{NP} \\ \text{Verb_P:} & \tau_{VP} \end{bmatrix}$$

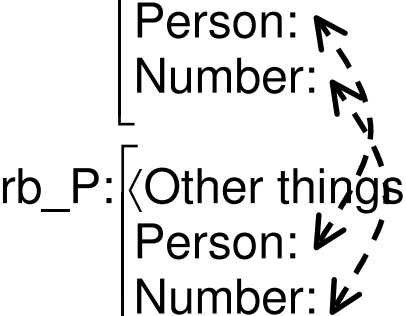
The corresponding logical constraint:

$$(\forall x)(\exists y)(\exists z)$$
$$(\tau_S(x) \Rightarrow x{:}\text{Subj}{:}\tau_{NP}(y) \wedge x{:}\text{Verb_P}{:}\tau_{VP}(z))$$

To recapture agreement:

$$\tau_{NP} \begin{bmatrix} \langle \text{Other things} \rangle .. \\ \text{Person:} & \tau_{Pers} \\ \text{Number:} & \tau_{Num} \end{bmatrix} \qquad \tau_{VP} \begin{bmatrix} \langle \text{Other things} \rangle .. \\ \text{Person:} & \tau_{Pers} \\ \text{Number:} & \tau_{Num} \end{bmatrix}$$

$$\begin{bmatrix} \text{Subj:} & \begin{bmatrix} \langle \text{Other things} \rangle .. \\ \text{Person:} \\ \text{Number:} \end{bmatrix} \\ \text{Verb_P:} & \begin{bmatrix} \langle \text{Other things} \rangle .. \\ \text{Person:} \\ \text{Number:} \end{bmatrix} \end{bmatrix}$$



$$(\forall x)\ (\tau_S(x) \Rightarrow (x{:}\text{Subj}{\cdot}\text{Person} \asymp x.\text{Verb_P}{\cdot}\text{Person} \wedge$$
$$x{:}\text{Subj}{\cdot}\text{Number} \asymp x.\text{Verb_P}{\cdot}\text{Number}))$$

$VP \rightarrow V \mid V\ NP$

$$(\forall x)(\tau_{VP}(x) \Leftrightarrow \tau_{VP1}(x) \vee \tau_{VP2}(x))$$

Schema for V:

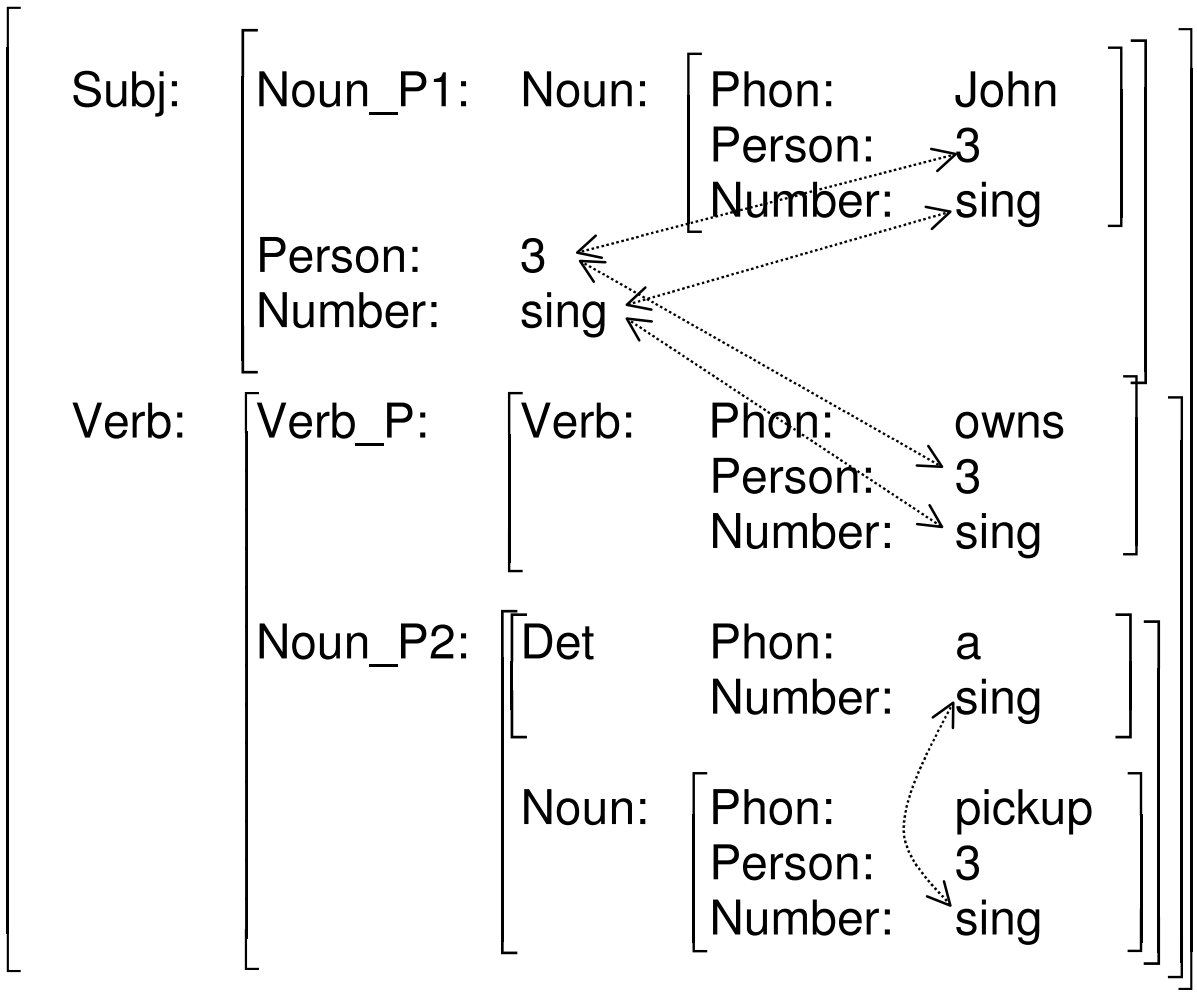
$$\tau_{VP1}\begin{bmatrix}\text{Phon:} & \tau_{V_Lex}\\ \text{Person:} & \tau_{Pers}\\ \text{Number:} & \tau_{Num}\end{bmatrix}$$

Instances from the lexicon would contain the following information:

$$\begin{bmatrix}\text{Phon:} & \text{own}\\ \text{Person:} & \text{any but}\\ \text{Number:} & \text{3p+sing}\end{bmatrix} \vee \begin{bmatrix}\text{Phon:} & \text{owns}\\ \text{Person:} & 3\\ \text{Number:} & \text{sing}\end{bmatrix} \vee ..$$

This process goes on and on..

This is what the final parse might look like:



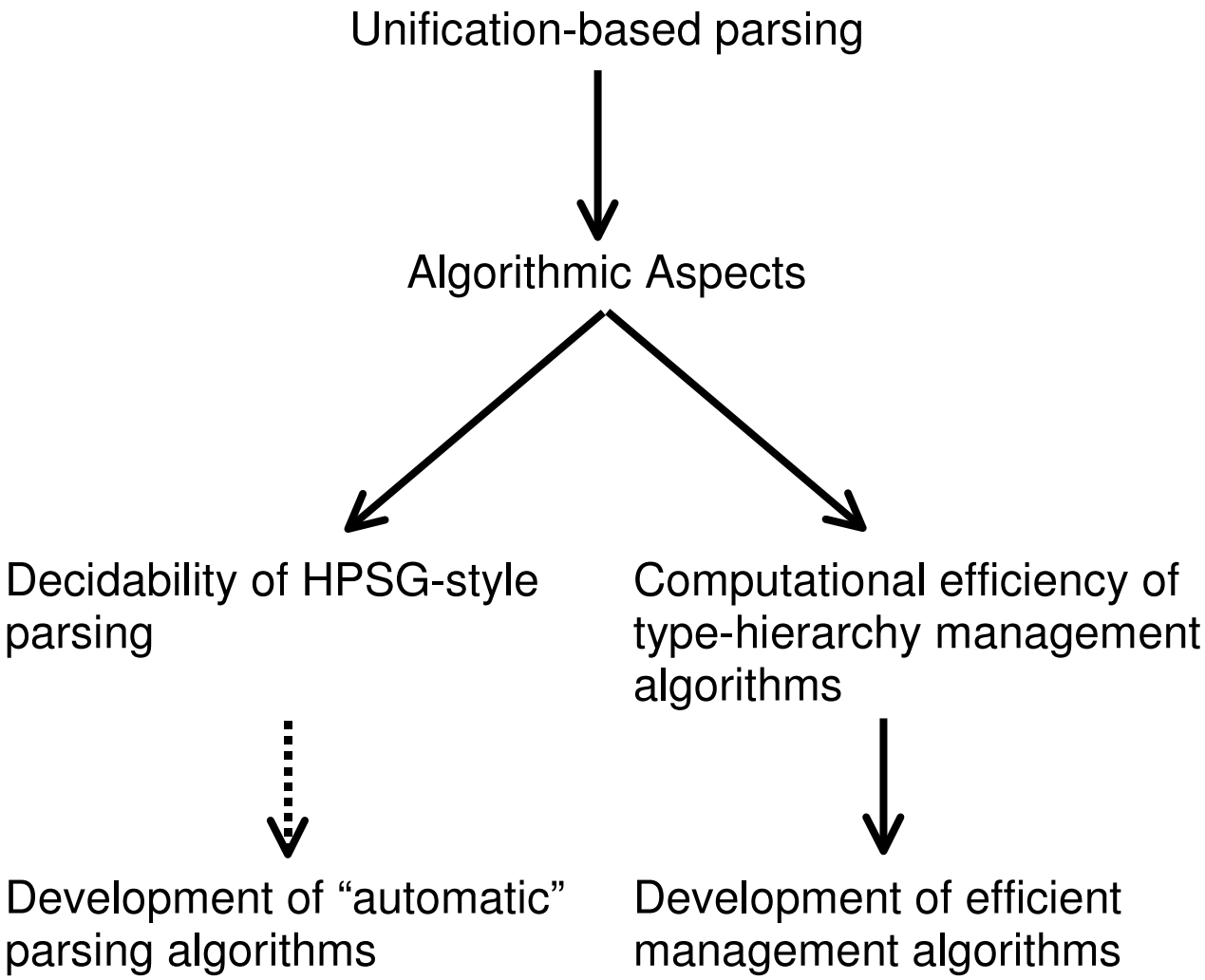
Notes:

- Such a structure is a model of the appropriate constraints in the underlying logic:
  - Framework constraints;
  - Input specific constraints.

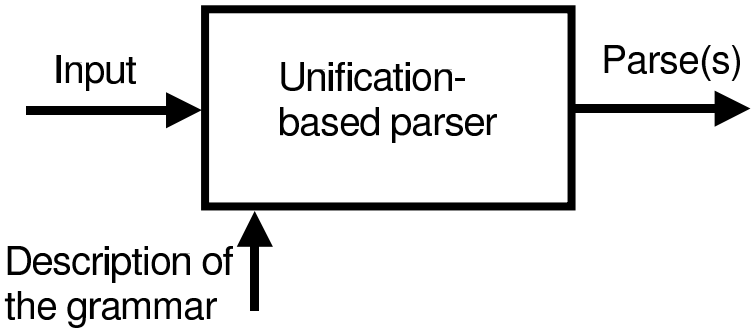
- If there were more than one legal parse, each such parse would be represented by a structure of this form.

- This is **not** HPSG.

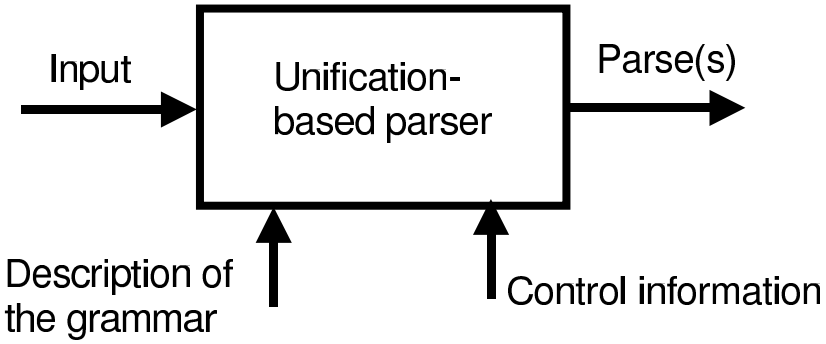
Current Research Directions



Research Problem 1: The Decidability of Parsing within HPSG-style Formalisms



The *ideal* world of constraint-based parsing
(Similar to the *real* world of PL parsing.)



The *real* world of constraint-based parsing

- Question: Is it *possible* to render the process of constraint-based parsing decidable, so that the grammar writer / user need not supply control information?

- This is a nontrivial question...

Motivation:

- The process of understanding natural language appears to be decidable. (Humans do it all the time.)
- Generative frameworks such as LFG render large fragments of natural languages decidable. The user need not supply control information.
- Existing tools for unification-based grammatical formalisms (such as HPSG) require the user to supply control information.

General approach:

- Formalize to an appropriate logic.
- Show the logic, or significant fragments thereof, to be decidable.

My approach:

- Develop a custom logic, and then embedded this logic into first-order predicate logic.
- Show that this fragment is decidable.
- Rationale:
  - A great deal is known about decidable and undecidable classes of formulas within first order logic. This approach makes use of that knowledge.

Other approaches:

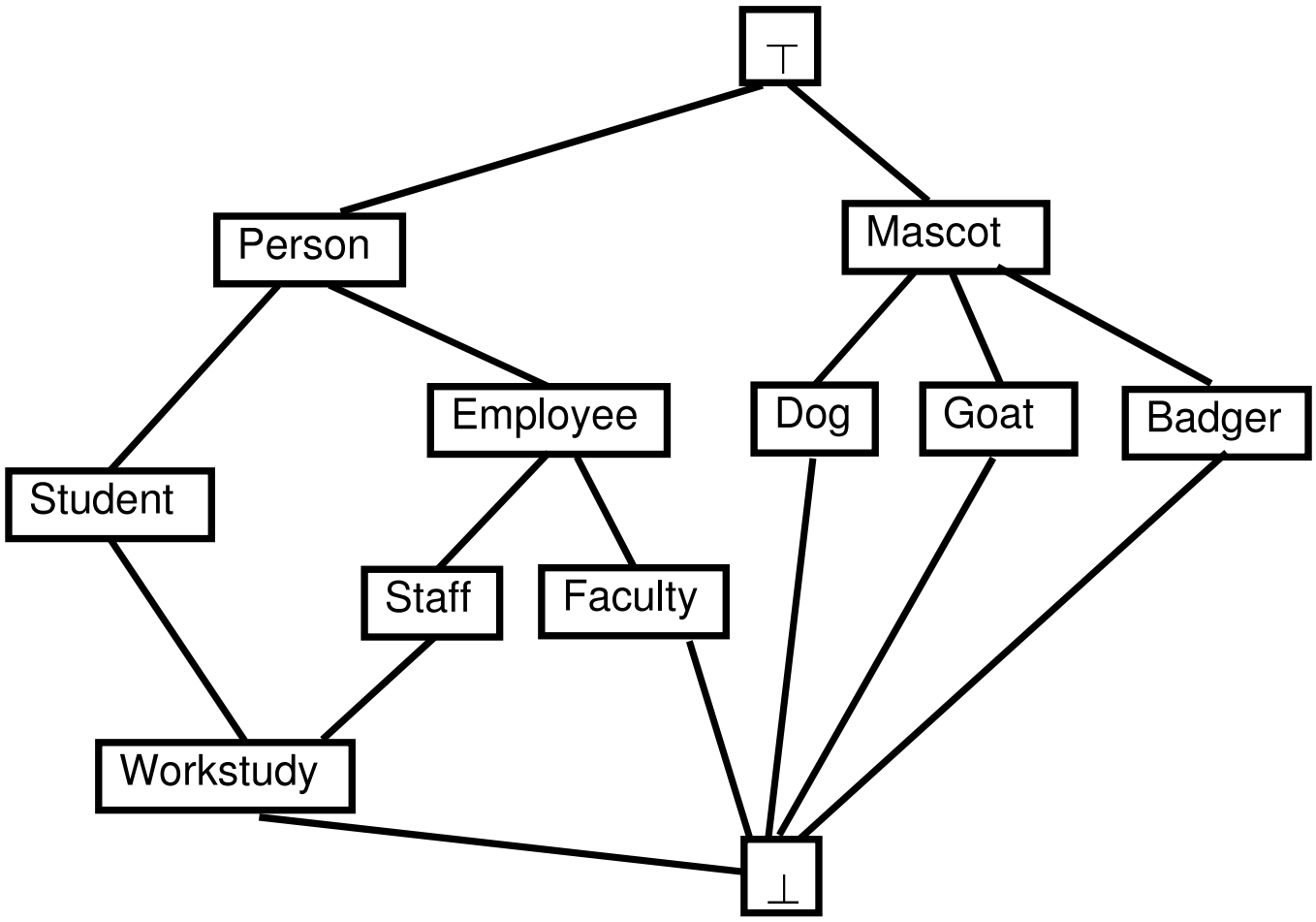
- Custom logics:
  - Tübingen (King, Kepser)
  - Edinburgh (Manandhar)

Research Problem 2: Efficient Computational Management of Partially Specified Type Hierarchies

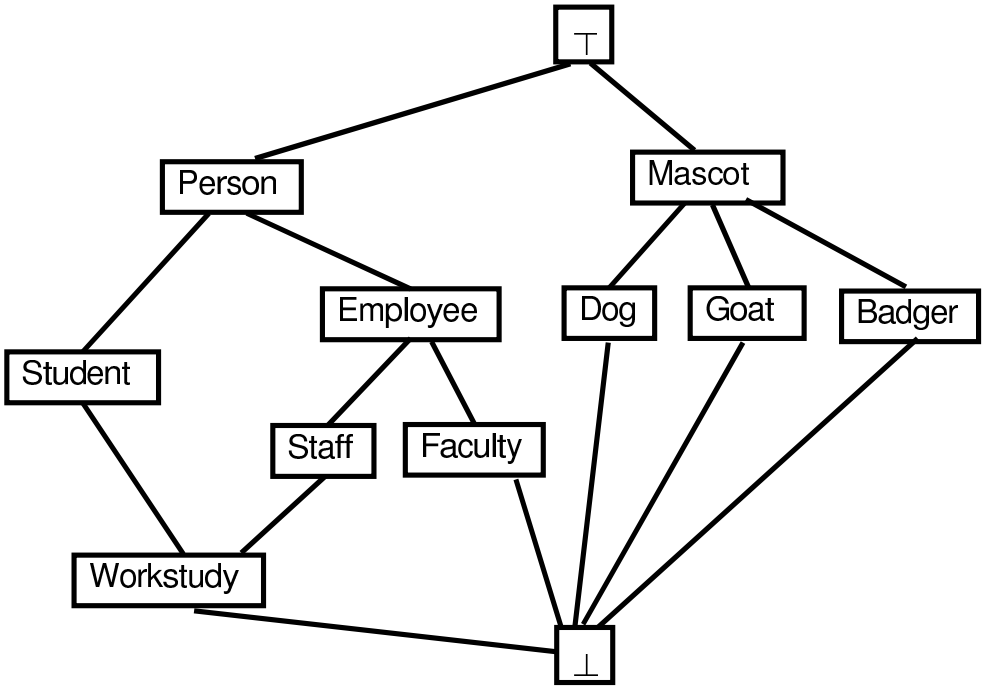
- Feature logic, as applied to formalisms such as HPSG, depends critically upon the concept of a type hierarchy.

What is a type hierarchy?

- The concept of inheritance via a hierarchy should be familiar from object-oriented languages.



The Semantics of Type Hierarchies:



- Every class has a set of instances:
  - Inst(Student) = the class of students, etc.
  - Inst($\top$) = universe; Inst($\bot$) = $\varnothing$.

ISA semantics:

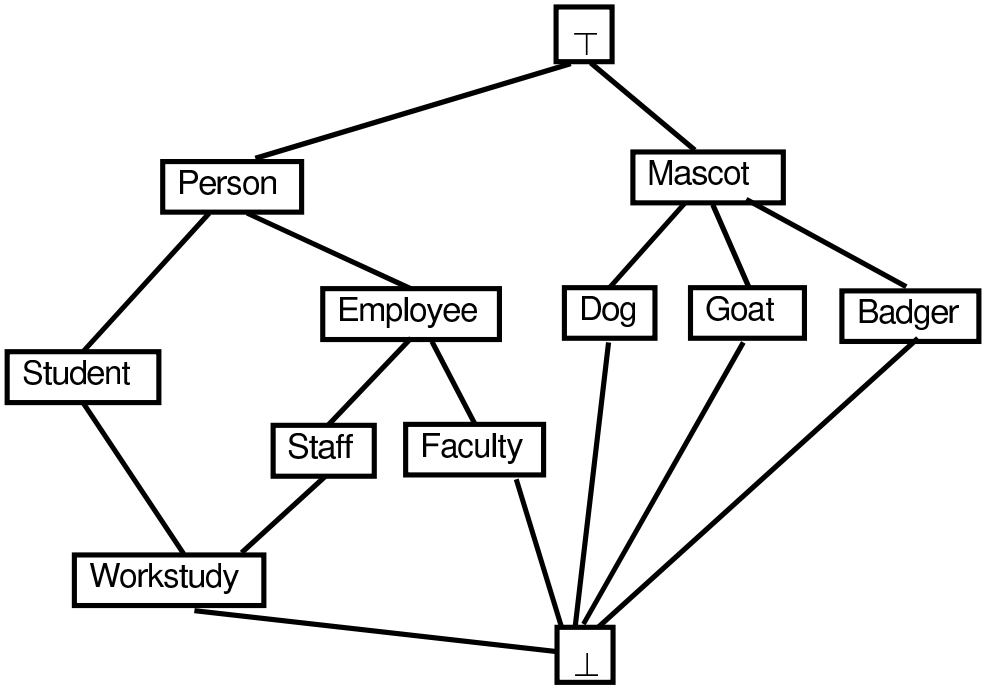
- $X \leq Y \Rightarrow \text{Inst}(Y) \subseteq \text{Inst}(X)$.
  - Workstudy(x) $\Rightarrow$ Student(x)

Natural semantics:

($\wedge$ = infimum ; $\vee$ = supremum)

- $Z = X \wedge Y \Rightarrow \text{Inst}(Z) = \text{Inst}(X) \cap \text{Inst}(Y)$
- $Z = X \vee Y \Rightarrow \text{Inst}(Z) = \text{Inst}(X) \cup \text{Inst}(Y)$
  - Student(x) $\wedge$ Staff(x) $\Leftrightarrow$ Workstudy(x)
  - Student(x) $\vee$ Staff(x) $\Leftrightarrow$ Person(x)

Are these semantics always consistent?



The natural semantics is valid if and only if the hierarchy does not contain a *pentagon* or a *diamond*.

This is equivalent to satisfaction of either of the *distributive laws* (Birkhoff"s representation):

- $(X \wedge Y) \vee Z = (X \vee Z) \wedge (Y \vee Z)$
- $(X \vee Y) \wedge Z = (X \wedge Z) \vee (Y \wedge Z)$

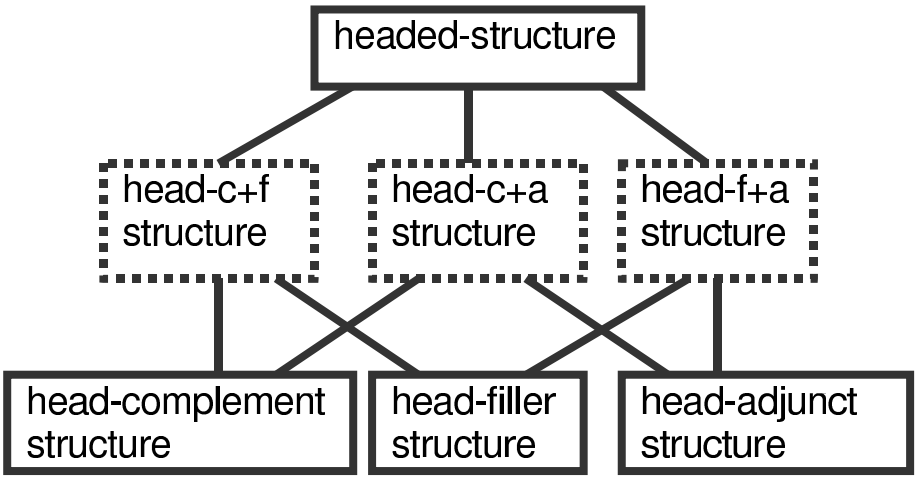
- These tests may be performed in time $O(n^3)$, with n the number of classes in the hierarchy.

Partial Specification

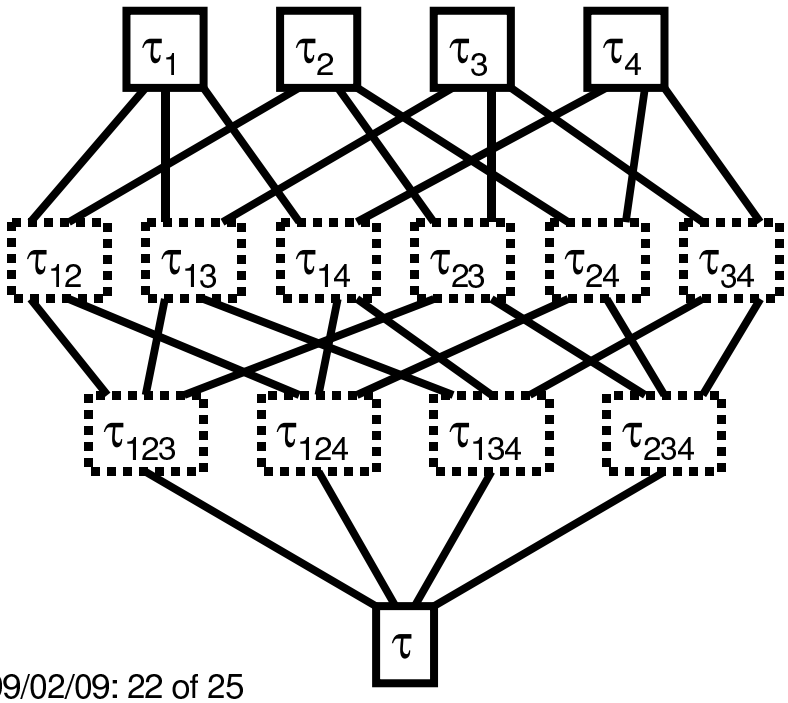
The need:

- In large systems, complete explicit specification of the type hierarchy is impractical.

headed-structure = head-complement-structure
$\vee$ head-filler-structure
$\vee$ head-adjunct-structure



$\tau = \tau_1 \wedge \tau_2 \wedge \tau_3 \wedge \tau_4$



Formal Open Specification

In a formal *open specification*, declarations of the following forms are allowed.

Order constraints:

(a) $\tau_1 \leq \tau_2$

(b) $\tau = \tau_1 \vee \tau_2 \vee .. \vee \tau_n$

(c) $\tau = \tau_1 \wedge \tau_2 \wedge .. \wedge \tau_n$

Note: Supremum is not $\vee$, by default.
Infimum is not $\wedge$, by default.

Position constraints:

(d) $\tau_1 \neq \tau_2$

(e) $\mathit{Atom}(\tau)$ $(\tau \notin \{\top,\bot\}.)$

- The interpretation of these constraints is in accordance with the natural semantics, and is inherently partial:
  - A collection of such rules characterizes not a single hierarchy; such a collection is a set of constraints defining a set of hierarchies.

Goal:

- Find effective algorithms for characterizing the distributive hierarchies which are consistent with such a collection of rules.

Result:

- Does a given open specification have an extension to a complete distributive type hierarchy?
  - This problem is *NP-complete* [Hegner, 1995].
    $\Rightarrow$ The best known algorithm is $O(2^n)$.

Special aspects making this problem unique:

- NP-complete problems must be dealt with, and are dealt with, all the time.

- Existing techniques (approximation) apply to optimization problems, and are not applicable to this problem.

- This problem belongs to a class including satisfiability problems in logic, in which approximation makes no sense.

- In addition, *all* solutions are sought, rather than just a single solution.

Current Directions:

- Development of techniques which make use of the special properties of the solution space (it forms a complete lattice) to reduce greatly the amount of searching which needs to be done.

- Experimental measurement of algorithm performance.

- Incremental modification:

  - Linguistic databases, particularly lexicons, undergo frequent modification.

  - Rather than rebuild the entire hierarchy every time a modification is to be performed, it would be simpler to modify the existing hierarchy.

  - Current research includes investigation of determining whether an update of a legal formal specification is also legal.