Semantic Analysis

CMSC 35100

Natural Language Processing

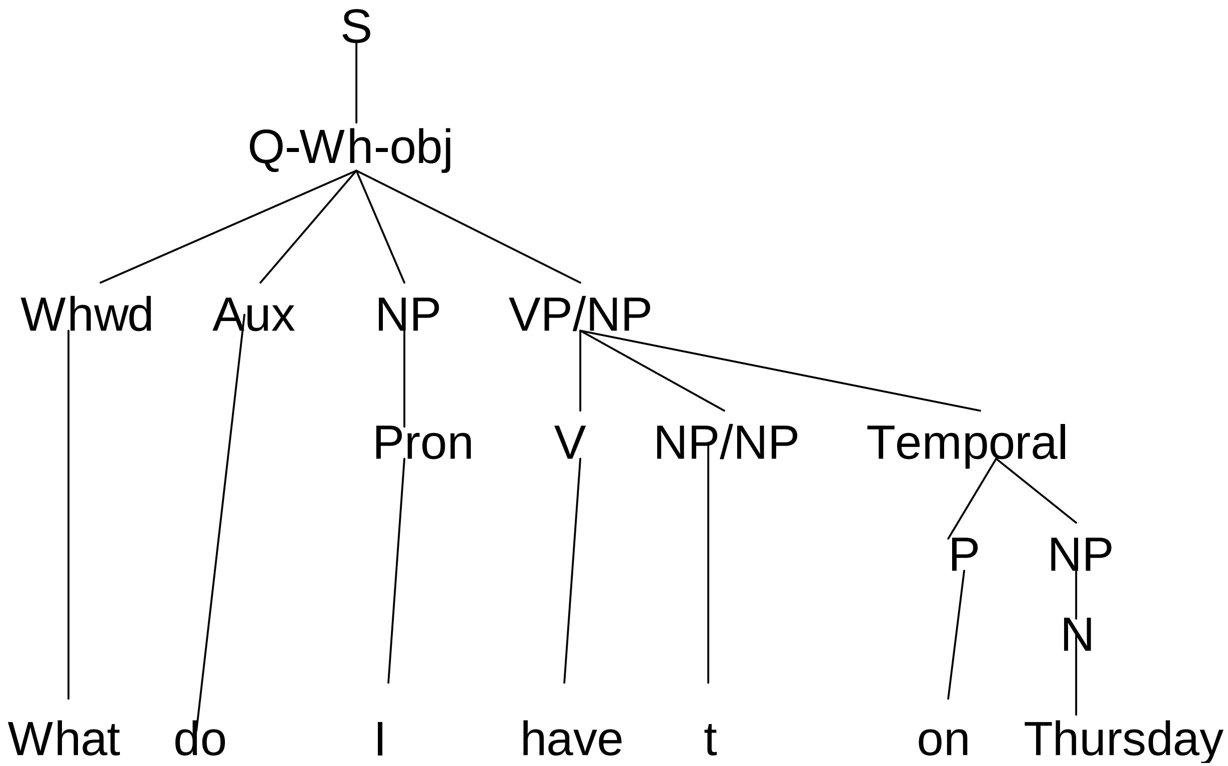
May 8, 2003

Roadmap

- Semantic Analysis
  - Motivation:
    - Understanding commands
  - Approach I: Syntax-driven semantic analysis
    - Augment productions with semantic component
      - Lambda calculus formulation
  - Approach II: Semantic Grammar
    - Augment with domain-specific semantics
  - Approach III: Information Extraction
    - Template-based semantics

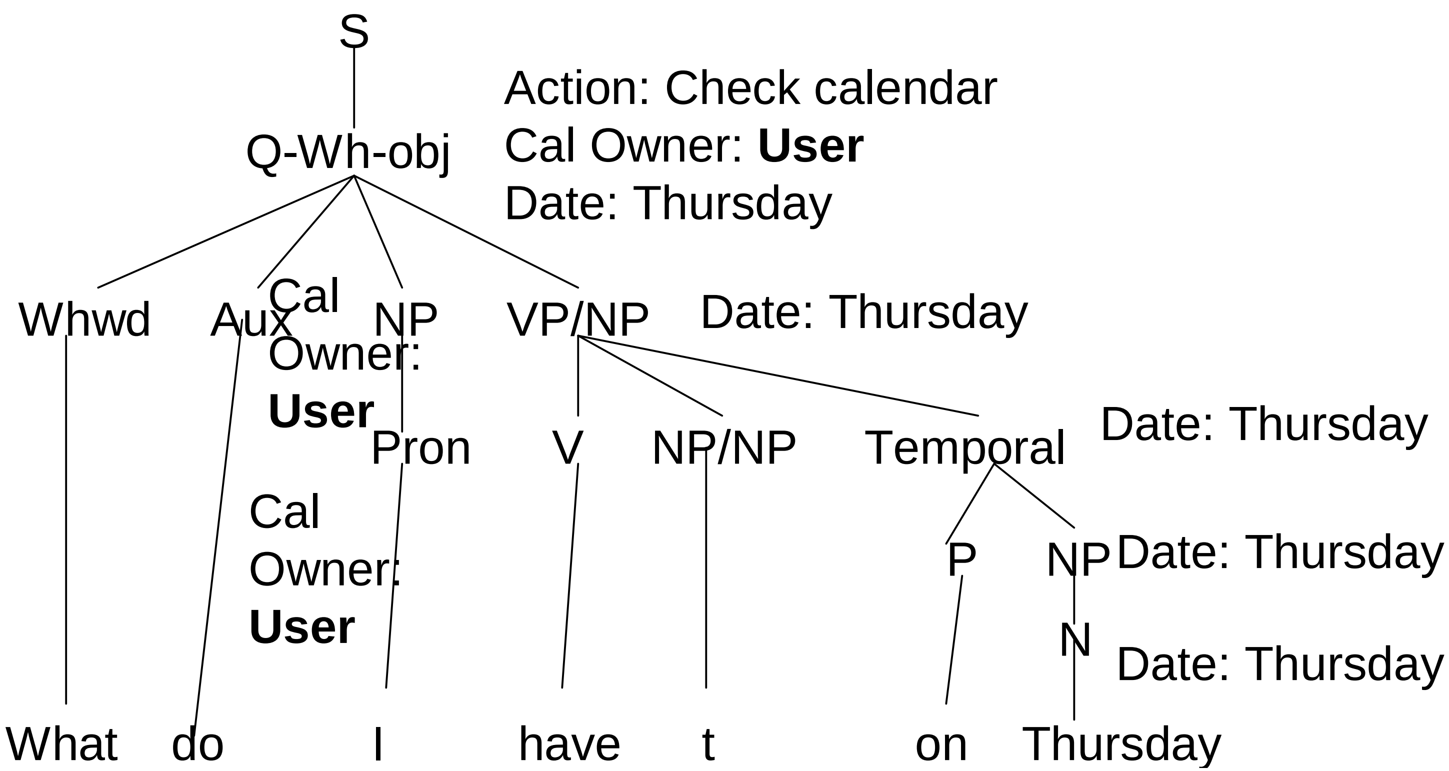
Understanding Commands

- “What do I have on Thursday?”
- Parse:



Understanding Commands

- Parser:
  - Yes, it's a sentence & here's the structure
- System: Great! But what do I do?



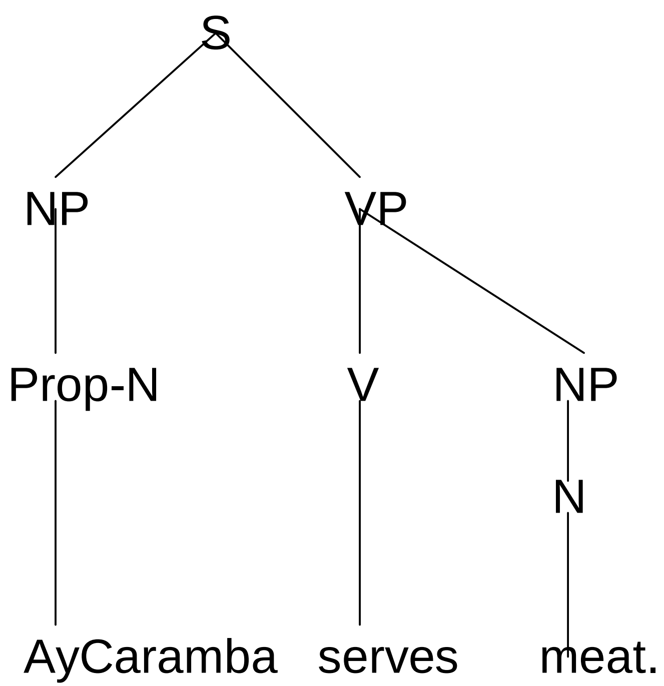
Syntax-driven Semantic Analysis

- Key: Principle of Compositionality
  - Meaning of sentence from meanings of parts
    - E.g. groupings and relations from syntax
- Question: Integration?
- Solution 1: Pipeline
  - Feed parse tree and sentence to semantic unit
  - Sub-Q: Ambiguity:
    - Approach: Keep all analyses, later stages will select

Simple Example

- AyCaramba serves meat.

$$\exists e\ Isa(e, Serving) \wedge Server(e, AyCaramba) \wedge Served(e, Meat)$$



Rule-to-Rule

- Issue:
  - Need detailed information about sentence, parse tree
    - Infinitely many sentences & parse trees
- Solution:
  - Tie semantics to finite components of grammar
    - E.g. rules & lexicon
  - Augment grammar rules with semantic info
    - Aka “attachments”
      - Specify how RHS elements compose to LHS

Semantic Attachments

- Basic structure:
  - A-> a1….an {f(aj.sem,…ak.sem)}
- Language for semantic attachments
  - Lambda calculus
    - Extends First Order Predicate Calculus (FOPC) with function application
- Example (continued):
  - Nouns represented by constants
    - Prop-n -> AyCaramba {AyCaramba}
    - N -> meat {meat}

Semantic Attachment Example

- Phrase semantics is function of SA of children
  - E.g. NP -> Prop-n {Prop-n.sem}
  - NP -> N {N.sem}
- More complex functions are parameterized
  - E.g. Verb -> serves

    $\{\lambda x \lambda y \exists e\ Isa(e, Serving) \wedge Server(e, y) \wedge Served(e, x)\}$
  - VP -> Verb NP {V.sem(NP.sem)}
    - Application= $\lambda y \exists e\ Isa(e, Serving) \wedge Server(e, y) \wedge Served(e, Meat)$
  - S -> NP VP
    - Application=

      $\exists e\ Isa(e, Serving) \wedge Server(e, AyCaramba) \wedge Served(e, Meat)$

Complex Attachments

- Complex terms:
  - Allow FOPC expressions to appear in otherwise illegal positions
    - E.g. Server(e, $\exists$ x Isa(x,Restaurant))
    - Embed in angle brackets
    - Translates as $\exists$ x Isa(x,Restaurant) $\wedge$ Server(e,x)
      - Connective depends on quantifier
- Quantifier Scoping
  - Ambiguity: Every restaurant has a menu
    - Readings: all have a menu; all have same menu
    - Potentially O(n!) scopings (n=# quanifiers)
  - Solve ad-hoc fashion

Inventory of Attachments

- S -> NP VP {DCL(VP.sem(NP.sem))}
- S -> VP {IMP(VP.sem(DummyYou)}
- S -> Aux NP VP {YNQ(VP.sem(NP.sem))}
- S -> WhWord NP VP
  - {WHQ(NP.sem.var,VP.sem(NP.sem))}
- Nom -> Noun Nom {λx Nom.sem(x) ^ NN(Noun.sem)}
- PP -> P NP {P.sem(NP.sem)} ;; NP mod
- PP -> P NP {NP.sem} ;; V arg PP
- P -> on {λyλx On(x,y)}
- Det -> a {∃}
- Nom -> N {λx Isa(x,N.sem)}

Earley Parsing with Semantics

- Implement semantic analysis
  - In parallel with syntactic parsing
    - Enabled by compositional approach
- Required modifications
  - Augment grammar rules with semantic field
  - Augment chart states with meaning expression
  - Completer computes semantics – e.g. unifies
    - Can also fail to unify
      - Blocks semantically invalid parses
    - Can impose extra work

Sidelight: Idioms

- Not purely compositional
  - E.g. kick the bucket = die
  - tip of the iceberg = beginning
- Handling:
  - Mix lexical items with constituents (word nps)
  - Create idiom-specific const. for productivity
  - Allow non-compositional semantic attachments
- Extremely complex: e.g. metaphor

Approach II: Semantic Grammars

- Issue:
  - Grammatical overkill
    - Constituents with little (no) contribution to meaning
    - Constituents so general that semantics are vacuous
  - Mismatch of locality
    - Components scattered around tree
- Solution: Semantic Grammars
  - Developed for dialogue systems
    - Tied to domain
    - Exclude unnecessary elements

Semantic Grammar Example

- What do I have on Thursday?
  - CalQ -> What Aux UserP have {on} DateP
    - Cal action:=find; CalOwner:= head UserP; Date:=head DateP;
  - UserP-> Pron
    - Head:=Head Pron
  - Pron-> I
    - Head:= USER
  - DateP -> Dayof Week
    - Head:= sem DayofWeek

Semantic Grammar Pros & Cons

- Useful with ellipsis & anaphora
  - Restrict input by semantic class: e.g. DataP

- Issues:
  - Limited reuse
    - Tied to application domain
  - Simple rules may overgenerate